

ON-LINE TIME WARPING OF HUMAN MOTION SEQUENCES

MATHEW RANDALL



DMT LAB

SCHOOL OF COMPUTING AND DIGITAL TECHNOLOGY
FACULTY OF COMPUTING, ENGINEERING AND THE BUILT ENVIRONMENT
BIRMINGHAM CITY UNIVERSITY, BIRMINGHAM, UK

A REPORT SUBMITTED AS PART OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

AUGUST 2023

Abstract

Some application areas require motions to be time warped on-line as a motion is captured, aligning a partially captured motion to a complete prerecorded motion. For example movement training applications for dance and medical procedures, require on-line time warping for analysing and visually feeding back the accuracy of human motions as they are being performed. Additionally, real-time production techniques such as virtual production, in camera visual effects and the use of avatars in live stage performances, require on-line time warping to align virtual character performances to a live performer.

The work in this thesis first addresses a research gap in the measurement of the alignment of two motions, proposing approaches based on rank correlation and evaluating them against existing distance based approaches to measuring motion similarity. The thesis then goes onto propose and evaluate novel methods for on-line time warping, which plot alignments in a forward direction and utilise forecasting and local continuity constraint techniques.

Current studies into measuring the similarity of motions focus on distance based metrics for measuring the similarity of the motions to support motion recognition applications, leaving a research gap regarding the effectiveness of similarity metrics bases on correlation and the optimal metrics for measuring the alignment of two motions. This thesis addresses this research gap by comparing the performance of variety of similarity metrics based on distance and correlation, including novel combinations of joint parameterisation and correlation methods. The ability of each metric to measure both the similarity and alignment of two motions is independently

assessed.

This work provides a detailed evaluation of a variety of different approaches to using correlation within a similarity metric, testing their performance to determine which approach is optimal and comparing their performance against established distance based metrics. The results show that a correlation based metric, in which joints are parameterised using displacement vectors and correlation is measured using Kendall Tau rank correlation, is the optimal approach for measuring the alignment between two motions. The study also showed that similarity metrics based on correlation are better at measuring the alignment of two motions, which is important in motion blending and style transfer applications as well as evaluating the performance of time warping algorithms. It also showed that metrics based on distance are better at measuring the similarity of two motions, which is more relevant to motion recognition and classification applications.

A number of approaches to on-line time warping have been proposed within existing research, that are based on plotting an alignment path backwards from a selected end-point within the complete motion. While these approaches work for discrete applications, such as recognising a motion, their lack of monotonic constraint between alignment of each frame, means these approaches do not support applications that require an alignment to be maintained continuously over a number of frames. For example applications involving continuous real-time visualisation, feedback or interaction.

To solve this problem, a number of novel on-line time warping algorithms, based on forward plotting, motion forecasting and local continuity constraints are proposed and evaluated by applying them to human motions. Two benchmark standards for evaluating the performance of on-line time warping algorithms are established, based on UTW time warping and comparing the resulting alignment path with that produced by DTW. This work also proposes a novel approach to adapting existing local continuity constraints to a forward plotting approach.

The studies within this thesis demonstrates that these time warping approaches are able to produce alignments of sufficient quality to support applications that require

an alignment to be maintained continuously. The on-line time warping algorithms proposed in this study can align a previously recorded motion to a user in real-time, as they are performing the same action or an opposing action recorded at the same time as the motion being align. This solution has a variety of potential application areas including: visualisation applications, such as aligning a motion to a live performer to facilitate in camera visual effects or a live stage performance with a virtual avatar; motion feedback applications such as dance training or medical rehabilitation; and interaction applications such as working with Cobots.

Acknowledgements

Without the help, support and encouragement of many people this thesis would not have been possible.

I would like thank my supervisors, Prof Ian Williams, Dr Carlo Harvey and Prof. Cham Athwal, for regularly giving up their time for many years to support and direct me as a PhD student. Thank-you to all the members of the DMT Lab for contributing to the research environment and eco system in which this work took place.

A special thank-you to my Wife, Bryony Randall, for being a constant source of support and encouragement. Bryony has made significant sacrifices to allow me to complete this work and always had faith that I could complete it. I would like to thank my children, Reece and Jack Randall for encouragement and enduring faith in me.

Declaration

I confirm that the work contained in this BSc project report has been composed solely by myself and has not been accepted in any previous application for a degree. All sources of information have been specifically acknowledged and all verbatim extracts are distinguished by quotation marks.

Signed

Date

Mathew Randall

Contents

Abstract	2
Acknowledgements	5
Declaration	6
1 Introduction	34
1.1 Motivation	35
1.1.1 On-line Time Warping of Opposing Motions	35
1.1.2 Interaction Between Performers and Virtual Characters	36
1.1.3 Additional Application Areas	38
1.1.4 Additional Capabilities	38
1.2 Requirements	39
1.2.1 Motion Adaptation	39
1.2.2 Artistic Control	40
1.3 Challenges	40
1.3.1 On-Line Time Warping	40
1.3.2 Measuring Alignment	41
1.4 Research Aim	41
1.5 Research Objectives	41
1.6 Methodology	42
1.7 Approach	42
1.8 Thesis Structure	44
1.9 Contributions	45

1.10	Publications	46
2	Interpretation and Parameterization of Motion Capture Data	47
2.1	Introduction	47
2.2	Motion Capture Solutions	48
2.2.1	Optical Systems	48
2.2.2	Image Processing Systems	49
2.2.3	Inertial Systems	50
2.2.4	Motion Capture Solutions Summary	50
2.3	Working with Motion Capture Data	51
2.3.1	Forward Kinematics	52
2.3.2	Joint Axis	54
2.3.3	Re-sampling	55
2.4	Parameterising Joint Angles	56
2.4.1	Euler	56
2.4.2	Quaternions	56
2.4.3	Rotation Matrices	59
2.4.4	Displacement Vector	60
2.4.5	Logarithmic Map	60
2.4.6	Rotation Vector	61
2.4.7	Overview of Parametrization Methods	61
2.5	Parameterising Motions	62
2.5.1	Sensor Parameters	62
2.5.2	Feature Parameters	64
2.6	Motion Editing	67
2.6.1	Editing Motion Signals	68
2.6.2	Applying Constraints	69
2.6.3	Editing via Abstract Models	70
2.6.4	Blending and Synthesising Motions	71
2.6.5	Impact on Perceived Realism	72
2.7	Summary	72

3	Measuring Similarity and Time Warping of Human Motions	73
3.1	Measuring Similarity	73
3.1.1	Introduction	73
3.1.2	Types of Similarity Metric	74
3.1.3	Preparing Data	74
3.1.4	Conforming Motions	75
3.1.5	Joint Weightings	76
3.1.6	Distance Base Metrics	77
3.1.7	Correlation Base Metrics	83
3.1.8	Feature Based Metrics	84
3.2	Motion Recognition	85
3.2.1	Motion Recognition Using a Similarity Metric	85
3.2.2	Motion Recognition Using Clustering Algorithms	85
3.2.3	Motion Recognition Using Statistical Probability	86
3.2.4	Motion Recognition and Time Warping	87
3.3	Time Warping	87
3.3.1	Introduction to Time Warping	87
3.3.2	Uniform Time Warping (UTW)	87
3.3.3	Dynamic Time Warping (DTW)	88
3.3.4	Correlation Time Warping	95
3.3.5	Multidimensional Time Warping	95
3.3.6	Guided Time Warping	96
3.3.7	On-line Time Warping	97
3.4	Summary	99
4	Measuring Alignment and Similarity of Human Motion	101
4.1	Introduction	101
4.2	Background	103
4.2.1	Comparing Time Warping Algorithms	103
4.2.2	Correlation Vs Distance For Measuring Motion Similarity	104
4.2.3	Linear and Rank Correlation Methods	106
4.2.4	Correlation as a Motion Analysis and Manipulation Tool	108

4.2.5	Evaluating Similarity Metrics	109
4.3	Methodology	110
4.3.1	Overview	110
4.3.2	Sourcing the Data-Sets	110
4.3.3	Preparing and Assembling Data-sets	112
4.3.4	Measuring Motion Similarity	116
4.3.5	Joint Weightings	121
4.3.6	Evaluating Similarity Metrics	121
4.4	Results	125
4.4.1	Interpreting Results	125
4.4.2	Measuring Motion Alignment	126
4.4.3	Measuring Motion Similarity	129
4.4.4	Review of Performance Tests	132
4.5	Discussion	133
4.5.1	Combining Similarity Metrics	133
4.5.2	Impact of Motion Characteristics	134
4.6	Conclusions	138
5	Online Warping of Human Motion Using Windowing	140
5.1	Introduction	140
5.2	Background	143
5.2.1	Forward Plotting	143
5.2.2	Smoothing Methods	145
5.2.3	Measuring Alignment	146
5.3	Methodology	149
5.3.1	Overview	149
5.3.2	Sourcing and preparing the data-set	149
5.3.3	On-line time-warping algorithms	155
5.3.4	Evaluating the performance of time-warping algorithms	168
5.4	Results	172
5.4.1	Interpreting the results	172
5.4.2	Confidence in Results	172

5.4.3	Alignment and Distortion Results	174
5.4.4	Accuracy Results	176
5.4.5	Computational Performance	179
5.4.6	Visualising Aggregated Plots	180
5.4.7	Visualising Individual Alignments	182
5.5	Discussion	187
5.5.1	Impact of Sample Rate on Window Size	187
5.5.2	Joint Weighting	189
5.5.3	Jittery Alignments	190
5.5.4	Optimizing Alignment Accuracy	190
5.5.5	Optimizing Computational Performance	192
5.5.6	Frame Dropping	193
5.5.7	Performance Tests	194
5.6	Conclusions	194
6	Constraints and Penalties	196
6.1	Introduction	196
6.2	Background	197
6.2.1	Constraints	197
6.2.2	Penalties	202
6.3	Constraints Methodology	204
6.3.1	Adapting Constraints to Support Forward Plotting	204
6.3.2	Implementing Constraints	207
6.4	Results of Type V Constraint	208
6.4.1	Test Implementation and Interpreting Results	208
6.4.2	Alignment and Distortion Results	209
6.4.3	Accuracy Results	212
6.4.4	Computational Performance	215
6.4.5	Aggregate Plots	216
6.5	Discussion of Constraint Findings	219
6.5.1	Key Findings	219
6.5.2	Sticking on Input Frames	220

6.5.3	Impact of Global Constraints	221
6.5.4	Appraisal of Constraint Study	222
6.6	Penalties Methodology	223
6.6.1	Motivation and Overview	223
6.6.2	Determining the Penalty Factor	223
6.6.3	Determining the Normalisation Coefficient	226
6.6.4	Implementation of Penalties	227
6.7	Results of Implementing Penalties	228
6.7.1	Test Selection and Implementation	228
6.7.2	Performance Test Results	229
6.7.3	Aggregate Plots	233
6.8	Discussion of Penalty Findings	233
6.8.1	Impact of Penalties on Time Warping Methods	233
6.8.2	Normalisation Coefficients	236
6.8.3	Penalty Implementation	236
6.9	Conclusions	237
7	Impact of Movement and Motion Data Characteristics on the Performance of Time Warping Algorithms	240
7.1	Introduction	240
7.1.1	Identification of Most Impactful Characteristics	242
7.2	Under Utilised and Under Constrained Joints	242
7.3	Data Capture Errors	248
7.3.1	Noise	248
7.3.2	Inconsistent Expression of Joint Rotations	250
7.3.3	Flat Spots	251
7.3.4	Comparison of BCU and HDM05 Data-sets	253
7.4	Mistimed Starts	254
7.4.1	Description of Mistimed Starts	254
7.4.2	Identifying and Visualising Mistimed Starts	255
7.4.3	Evaluation of Alignments	259
7.4.4	Relationship Between Size of Mistiming and Window Size	262

7.4.5	Impact on Motion and Alignment Quality	264
7.4.6	Tolerance of On-line Warping Algorithms for Mistimed Starts	267
7.4.7	Impact of Constraints on Mistimed Starts	270
7.4.8	Impact of Penalties on Mistimed Starts	277
7.4.9	Conclusions on Mistimed Starts	283
7.5	Speed Differentials	285
7.5.1	Visualising the Impact of Speed Differentials on Alignment . .	285
7.5.2	Impact of Speed Differentials on Standard Time Warping Al- gorithms	286
7.5.3	Impact of Constraints and Penalties on Dealing with Speed Differentials	291
7.5.4	Conclusions on Speed Differentials	293
7.6	Summary	296
8	Conclusions and Recommendations	298
8.1	Recommendations	299
8.1.1	Measuring Similarity and Alignment	299
8.1.2	On-line Time Warping of Motions	300
8.2	Review of Aims and Objectives	302
8.3	Potential Applications of This Work	304
8.4	Implementation Challenges	305
8.5	Limitations and Future Work	306
8.5.1	Joint Weightings	306
8.5.2	Impact of Motion Characteristics	306
8.5.3	Simulating Time Warps	307
8.5.4	Perception Testing	307
8.5.5	Similarity Metrics	308
8.5.6	COVID-19	308
8.6	Impact	308
	Bibliography	310
	A Joint Range of Motions	332

List of Tables

2.1	An overview comparing the advantages and disadvantages of different types of motion capture solutions.	51
2.2	Overview of the characteristics of each method of parameterising rotations.	62
2.3	General approaches to detecting relational features in human motion, as defined by Müller and Röder (2008)	66
3.1	Optimal weighting of joints within a similarity metric, determined by a regression(Wang and Bodenheimer, 2003) and human perception(Harada et al., 2004).	77
4.1	To create a data-set of aligned motion pairs, time-warping was performed on the following permutations of input and target motions within each motion-set (a, b, c) , with the resulting aligned motions.	113
4.2	The composition of motions pairs within the Non-aligned data-set. For each motion set (a, b, c) , nine motion pairs are created from the combinations identified below. Note that aligned motions such as $a \rightarrow b$ are also incorporated into the data-set.	115
4.3	The number of motion pairs contained in each data-set used in this study.	115
4.4	Equations for calculating the size of the data-sets that would generated from j motion-sets each consisting of k captures. Note that ${}^n P_r$ and ${}^n C_r$ represent the number of possible permutations and combinations of r elements in a set of size n	116

4.5	Overview of the joint parameterisation methods used in this study.	117
4.6	The performance of each similarity metric as a measure of alignment. The ability of metrics to measure alignment was evaluated by comparing the similarity scores of motion pairs from the <i>Aligned</i> and <i>Non-Aligned</i> data sets.	127
4.7	The performance of each similarity metric as a measure of similarity. The ability of metrics to measure similarity was evaluated by comparing the similarity scores of motion pairs from the <i>Similar</i> and <i>Dissimilar</i> data-sets.	131
4.8	The mean similarity scores for each motion set. The ticks indicate the features contained within each motion. The mean similarity is based on the scores attained from motion pairs in the aligned data-set.	137
5.1	The composition of motions within the data-set used for this study.	156
5.2	A power analysis of the alignment test results in Figure 5.8a, showing the number of samples required to have confidence in a performance difference existing between any two time warping algorithms. The analysis was conducted with a statistical power of 80% and significance level of 5%. Each algorithm was tested with 3248 samples, giving confidence that differences do exist between the performance of almost all the time warping algorithms tested. A small number of outlying pairs of algorithms, require more than 3248 samples, as they have a similar distribution of results in the alignment test.	174
5.3	The means μ and standard deviations σ of the alignment test scores achieved by each time warping algorithm, for motions containing different types of movement as identified in Table 5.1	177
5.4	The average time each algorithm takes to process a single frame of motion data in microseconds.	179
6.1	Types of local constraints, expressed as sets of allowable paths $\{P_1 \dots P_t\}$	200

6.2	The q values for different types of constraint. The local constraints result in a search area resembling an Itakura Parallelogram, the smaller the q value the more constrained the alignment and the smaller the search area.	201
6.3	Constraint state table for the adapted Type V constraint.	207
6.4	The means and standard deviations of the alignment test scores achieved with the Type V constraint applied to each algorithm, for motions containing different types of movement. Green and red values indicate higher or lower values respectively, than those of the same algorithm and motion type without a constraint applied. The higher values in green indicate an improvement in performance.	211
6.5	The average time each algorithm takes to process a single frame of motion data in microseconds.	215
6.6	List of the penalty coefficients to be tested	228
6.7	The mean performance test scores attained by time warping algorithms with different penalty configurations applied. Values in green and red indicate algorithms with penalties applied performing better or worse, respectively, than the same algorithm without the penalty applied.	231
6.8	Overview of which on-line time warping algorithms met key performance thresholds, both with and without the constraint or penalties applied. U means the algorithm performed better than an offline UTW time warp. F means that the majority of alignments produced by the algorithm, were so close to the that produced by offline DTW, that differences may not be perceptible. Black cells represent test that were not performed.	238
7.1	Table of the mean similarity test scores attained by different on-line time warping algorithms, for each of the different movements used from HDM05 data-set.	246
7.2	The number of times flat spots occur and the length of the longest flat spot, in motions sourced from each data source.	252

7.3	A comparison of the mean similarity test scores attained by on-line time warping algorithms for comparable motions sourced from the BCU and HDM05 data sets.	253
7.4	The average deviation in frames between alignments produced by various online time warping algorithms and the offline DTW algorithm when applied to Walk Circle 6 Steps R Start and Turn Right motions, with mistimed leading inputs of a given number for frames, using a variety of online time warping algorithms. Green and red are used to indicate if the deviation is below the threshold at which the difference between the online time warp and DTW alignments would be visually perceptible (i.e. below 18 frames). The alignment paths plotted for these time warps can be seen in Figure 7.8.	260
7.5	The average deviation in frames between alignments produced by various online time warping algorithms and the offline DTW algorithm when applied to Walk Circle 6 Steps R Start and Turn Right motions, with mistimed trailing inputs of a given number of frames, using a variety of online time warping algorithms. Green and red are used to indicate if the deviation is below the threshold at which the difference between the online time warp and DTW alignments would be visually perceptible (i.e. below 18 frames). The alignment paths plotted for these time warps can be seen in Figure 7.9.	260
7.6	The performance of different time warping algorithms at aligning pairs of motions, starting with mistimed leading inputs of various sizes. Cells shown in green are where most of the alignment paths plotted, deviate less than the benchmark of 18 frames from alignment path plotted by the DTW algorithm.	284
7.7	The performance of different time warping algorithms at aligning pairs of motions, starting with mistimed trailing inputs of various sizes. Cells shown in green are where most of the alignment paths plotted, deviate less than the benchmark of 18 frames from alignment path plotted by the DTW algorithm.	284

7.8	The performance of different on-line time warping algorithms at aligning pairs of motions, in which the movements are performed faster in the input motion.	292
7.9	The performance of different on-line time warping algorithms at aligning pairs of motions, in which the movements are performed faster in the target motion.	292
7.10	Tables showing the performance of different online time warping algorithms, when aligning pairs of motions with either an input or target motion that has a shorter duration. The tables are split into algorithms with no penalty or constraint (top), with constraints (middle) and with penalties (bottom).	295
A.1	A specification for acceptable ROMs for key joints. The mean, μ , and standard deviation, σ , for each ROM are obtained from Boone and Azen (1979) and used to determine the maximum acceptable ROM for the respective joint axis using equation 5.8.	332

List of Figures

1.1	Two examples of interaction between live actors and virtual creatures from the film 'Harry Potter and the Order of the Phoenix' (Yates, 2011). On the left a character's hand needs to tightly follow the movement of an actor's neck close to camera. On the right an actor is being moved in a rig to match a reference movement in a monitor.	37
1.2	An overview of the thesis structure, showing the aim of each chapter and it's contribution to the thesis objectives stated in Section 1.5 . . .	45
2.1	Joint data represented as a transform matrix (left). Joint rotations are represented as a rotational matrix in the red part of the matrix. The length of the joint is represented as translation in the blue part of the matrix, in the form shown on the right, where joint length J_l is represented as a translation in the x axis.	53
2.2	Example of a simple kinematic chain	53
2.3	An example of a skeletal hierarchy with kinematic chains starting at the root and ending at end effectors: head end, thumb, finger and toe end.	54
2.4	Motion curves plotted for in the x, y, z axis for the right hip joint. . .	54
2.5	Examples of redundant axis, left unused by joints with limited degrees of freedom.	55

3.1	Comparison of UTW and DTW time warping, applied to a walking motion. The motion curves of the x axis of the left hip and poses show the accuracy of DTW and UTW alignment techniques at aligning the input motion to the target motion.	89
3.2	An example of a DTW alignment, overlaid over a normalised version of the accumulated cost matrix through which the alignment was plotted. The DTW was performed using rotational distance based on the hip, knee, shoulder and elbow joints, which were each equally weighed.	93
3.3	Two constrained search areas within a cost matrix. Sakoe-Chuba Band (left) and the Itakura Parallelogram (right).	94
3.4	An example of a sub optimal alignment path plotted by an on-line warping algorithm.	98
4.1	A comparison of the effects of different signal transforms on distance and correlation based measurements of similarity. The original signal of a right elbow in a walking motion, is shown in blue, while the transformed signal, with a similar form, is shown in red. The poses of the original motion can be seen at the top of the figure. The measurements under each graph show the inaccuracies that can occur when using distance as a measure of similarity.	105
4.2	Examples of the three different types of motion pairs used in this study.	111
4.3	The Vicon motion capture system, used in this study. Only five cameras of the eight cameras are visible in the picture, they have been highlighted in red. The tracking markers which can be seen from this perspective have been highlighted in green. An approximate start and end point for motions involving a straight movement across capture volume, such as walking and jumping, have been marked out on the floor.	113

4.4	A comparison of how different methods of joint parametrisation represent the motion of a joint with one degree of freedom. The motion curves show the rotation of the left knee joint changing over time during the same walking motion.	119
4.5	A comparison of how different methods of joint parametrisation represent the motion of a joint with two degrees of freedom. The motion curves show the rotation of the left elbow joint changing over time during the same walking motion.	119
4.6	Configuration of joints used in study.	122
4.7	Results of the normal distribution tests performed on the similarity scores yielded by the similarity metrics in this study for each data-set of motion pairs. Passed tests ($p \geq 0.05$) are green, failed test ($p < 0.05$) are red.	126
4.8	Histograms visualising the overlap between the similarity scores of motion pairs in the <i>Aligned</i> and <i>Non-Aligned</i> data-sets. The probability of an aligned motion pair scoring less than a non-aligned pair is shown by p . The lower the value of p for a given similarity metric the better it is a differentiating between aligned and non-aligned motion pairs.	128
4.9	Histograms visualising the overlap between the similarity scores of motion pairs in the <i>Similar</i> and <i>Dissimilar</i> data-sets. The probability of a dissimilar motion pair scoring less than a similar motion pair, is shown by p . The lower the value of p for a given similarity metric the better it is a differentiating between the different types of motion pairs.	130
4.10	The motion curves of right hand joints and character poses, of a pair of walking motions, where the red motion has been aligned to the blue motion. Within the aligned data-set, correlation based similarity metrics scored this pair of motions the highest (μ score of 0.87), with a low deviation between the different metric scores ($\sigma = 0.086$). . . .	135

4.11	The motion curves of right hand joints and character poses, of a pair of motions with identical movements performed while sitting down, where the red motion has been aligned to the blue motion. Within the aligned data-set, correlation based similarity metrics scored this pair of motions the lowest (μ score of 0.81). with a higher deviation between the different metric scores ($\sigma = 0.182$).	136
5.1	An example of the lack of continuity between the alignment paths plotted for sequential frames of a target motion, when the mapping of the last incoming target frame is unconstrained. The input frames aligned to frames 15 and 16 break the monotonic constraint resulting in backwards jumps in the playback of the input motion.	142
5.2	The skeletal joint rig used in this study, showing the 'T' pose the joints form when their orientations are set to zero.	151
5.3	Search window used in Frame Matching algorithm, to find the pre-recorded input frame that best matches the incoming target frame . .	157
5.4	The performance of different smoothing levels at predicting human motion. Shows the error resulting from a given number of frames used to predict a motion (smoothing level s) i frames in to the future.	163
5.5	Plotting of path aligning prerecorded frames $\{I_{M_n} \dots I_{M_n+(w-1)}\}$ against a mixture of existing frames $\{T_n, T_{n+1}\}$ and predicted target frames $\{P_0 \dots P_{w-2}\}$, contained in set F	164
5.6	How the alignment path is used to determine if frames in input motion I are inserted, matched or deleted to align it with a target motion T , using the frame in I that is aligned to T_{n+1}	166

5.7	Determining which frame in input motion I to map to target motion T based the lowest cell in frame T_{n+1} the path goes through. After the path plots through (T_{n+1}, I_{M_n+2}) , it moves to frame, T_n , in the left example and continuous downwards in the same frame T_{n+1} in the right example. As only one frame of I can be mapped to frame T_{n+1} , in the right example the mapping of frame T_{n+1} is replaced as the plot continuous downwards in that frame, eventually resulting in the lowest point the the path plots in frame T_{n+1} being mapped to that frame.	166
5.8	The results of the performance tests. Ma, Mb and Mc refer to methods A, B and C respectively, while w specifies the window size used. .	173
5.9	Absolute deviation from the standard DTW alignment in frames. The blue line represents the threshold at which timing errors in character interactions can be perceived (Hoyet, McDonnell and O’Sullivan, 2012).	178
5.10	Heat maps visualising the alignment paths plotted by each algorithm. Each heat-map shows an aggregate of the alignment paths plotted by all 3248 samples in the data-set for a given time warping algorithm. Every alignment path has been normalised to a size of 100 by 100, each cell in the heat map shows the number of alignment points plotted within that cell. An optimal alignment algorithm will cluster alignment paths around the diagonal as shown with the DTW algorithm. Hot areas under the diagonal indicate alignment paths incorrectly sticking on frames, while hot areas above the diagonal indicate alignment paths incorrectly skipping frames.	181
5.11	A detailed presentation of the alignments produced by online time warping algorithms, showing: the alignment paths plotted by each time warping algorithm, aligned motion curves in comparison to the target motion, and rendered character posed from different elevations.	185
5.12	Performance results for Method C when applied to motions sampled at 60Hz.	188

6.1	Equations for defining slopes to determine the boundaries of global path constraints. Sakoe-Chuba Band (left) and the Itakura Parallelogram (right). In the example Itakura Parallelogram on the right $q = 2$	199
6.2	Two continuity constraints. On the left is a Type V continuity constraint used when plotting an alignment backwards backwards. On the right is an adapted version of the Type V constraint for plotting an alignment path forwards.	205
6.3	The constraint steps used to implement the adapted Type V constraint. The coloured arrows show how each end position in the constraint can be reached.	205
6.4	The alignment and distortion performance tests results for different time warping algorithms, with and without the Type V constraint applied. In both tests the higher the result the better. The blue line represents the alignment and distortion achieved by the standard offline DTW algorithm, and the red line represents the alignment achieved using UTW with no alignment. Ma , Mb and Mc refer to methods A, B and C respectively, while w specifies the window size used.	210
6.5	The accuracy performance tests results for different time warping algorithms, with and without the Type V constraint applied. Ma , Mb and Mc refer to methods A, B and C respectively, while w specifies the window size used. Results closest to zero are best. The blue line in the bottom chart represents the threshold at which timing errors in character interactions can be perceived (Hoyet, McDonnell and O’Sullivan, 2012).	213
6.6	Heat maps visualising the alignment paths plotted for all samples in the data-set, by time warping algorithms with the Type V constraint applied.	217
6.7	A comparison between the aggregate plots of algorithms Ma , Mc $w40$, and Mc $w80$ with and without the Type V constraint applied.	218

6.8	A visualisation of how well constraints with a given q value, fit the time warping requirements of the data-set used in this study.	221
6.9	Determining the penalty factors to apply to input frames within methods A and C	224
6.10	Example penalty factor matrices generate by equation 6.10	226
6.11	The results for the alignment, distortion and accuracy performance tests for selected time warping algorithms with different penalties of varying strength and penalty coefficient types, applied. The performance of the time warping algorithm with no penalty applied can be seen in red.	230
6.12	Heat maps visualising the alignment paths plotted by algorithms <i>Ma</i> , <i>Mc w10</i> and <i>Mc w40</i> , with different configurations of penalties applied.	234
7.1	Examples of motions which are under constrained. Due to the variety of ways in which the movement in the motion can be achieved, there is more variation between the poses and motion curves of different takes (recordings) of the same motion.	244
7.2	Examples of motions which are constrained. Due to the limited ways in which the movement can be achieved, there is less variation between the poses and motion curves of different takes (recordings) of the same motion.	245
7.3	An example of noise in a motion signal. Noise artefacts in the motion signal have been highlighted in red.	249
7.4	The noise present in motions sourced from the BCU and HDM05 data-sets.	249
7.5	Examples of the manner in which a joints orientations are expressed using Euler rotations, changing mid motion. In the top example the x and z axis jump from 180 to -180 as arm rotates in a complete circle. In the bottom example, the way in which the rotations in each axis are used to orientate the joint, appears to change arbitrarily between frames 160 and 175.	251

7.6	Examples of flat spots in motion data. The top example is caused by constraints stopping a knee joint from over extending, the bottom example is a motion with starts with a series dropped frames, which are substituted with zero values.	251
7.7	An example of mistiming between two jump motions, in which the recordings starts at different points in the motion. The recording of <i>Jump 01</i> starts the end initiation phase of the first jump, while the recording of <i>Jump 02</i> starts at the beginning of the initiation phase	255
7.8	Alignment paths plotted by different time warping algorithms, to align a selection of Walk Circle 6 Steps R Start and Turn Right motions. Every plot has a leading input which starts ahead of the target motion by a given number of frames. This should result in an alignment starting with the first frame of the input motion being mapped to multiple frames of the target motion, as the target motion catches up to the input motion. The deviation between the alignment paths produced by each online time warping algorithm and offline DTW, can seen in Table 7.4.	257
7.9	Alignment paths plotted by different time warping algorithms, to align a selection of Walk Circle 6 Steps R Start and Turn Right motions. Every plot has a trailing input which starts behind the target motion by a given number of frames. This results in an alignment that should start with the first few frames of the input motion being skipped, and not being mapped to the target motion as they are not required. The deviation between the alignment paths produced by each online time warping algorithm and offline DTW, can seen in Table 7.5.	258
7.10	A detailed presentation of the alignments produced by online time warping algorithms, when used to align motions with a leading input of a given number of frames.	265

7.11	A detailed presentation of the alignments produced by online time warping algorithms, when used to align motions with a trailing input of a given number of frames.	266
7.12	Charts showing the capability of different on-line time warping algorithms at aligning pairs of motions starting with mistimed leading inputs and trailing inputs of various sizes. Each bar represents the median sample from each group, the Error is the average number of frames by which the sample alignment deviates from the offline DTW alignment. The dashed blue line represents the threshold at which timing errors in character interactions can be perceived (Hoyet, McDonnell and O’Sullivan, 2012). The numbers above each group of bars indicates the number of alignments the results are based on.	268
7.13	Charts showing the capability of different on-line time warping algorithms, with and without local constraints applied, at aligning pairs of motions starting with mistimed leading inputs and trailing inputs of various sizes. Each bar represents the median sample from each group, the Error is the average number of frames by which the sample alignment deviates from the offline DTW alignment. Bars with a hash pattern, represent time warping algorithms with a the Type V constraint applied. The dashed blue line represents the threshold at which timing errors in character interactions can be perceived. The numbers above each group of bars indicates the number of alignments the results are based on.	271
7.14	A detailed presentation of alignments produced by online time warping algorithms, with (right) and without (left) constraints implemented, when used to align motions with a leading input of a given number of frames.	273
7.15	A detailed presentation of the alignments produced by online time warping algorithms, with (right) and without (left) constraints implemented, when used to align motions with a trailing input of a given number of frames.	274

7.16	Charts showing the capability of the <i>Ma</i> and <i>Mb w10</i> time warping algorithms, with and without different penalties applied applied, at aligning pairs of motions starting with mistimed leading inputs and trailing inputs of various sizes, against the allowable Error or deviation from DTW, shown in as a dashed blue line.	279
7.17	Charts showing the capability of the <i>Mc w10</i> and <i>Mc w40</i> time warping algorithms, with and without different penalties applied applied, at aligning pairs of motions starting with mistimed leading inputs and trailing inputs of various sizes, against the allowable Error or deviation from DTW, shown in as a dashed blue line.	280
7.18	A detailed presentation of alignments produced by online time warping algorithms, with and without penalties implemented, when used to align pairs of motions starting with mistimed leading inputs (left) and trailing inputs (right) of a given number of frames.	281
7.19	The alignment paths and motion curves, resulting from time warping two motions in which the same movements have been performed at different speeds. The same two motions have been time warped in different directions in samples 214 and 215. The input motion is performed faster than the target motion in the left hand sample and visa versa on the right.	287
7.20	The alignment paths and motion curves, resulting from time warping two motions in which the same movements have been performed at different speeds. The same two motions have been time warped in different directions in samples 334 and 335. The input motion is performed faster than the target motion in the left hand sample and visa versa on the right.	288

7.21	The alignment paths and motion curves, resulting from time warping two motions in which the same movements have been performed at different speeds. The same two motions have been time warped in different directions in samples 842 and 843. The input motion is performed faster than the target motion in the left hand sample and visa versa on the right.	289
7.22	The alignment paths and motion curves, resulting from time warping two motions in which the same movements have been performed at different speeds. The same two motions have been time warped in different directions in samples 2642 and 2643. The input motion is performed faster than the target motion in the left hand sample and visa versa on the right.	290

List of Algorithms

1	getCostMatix() Function	90
2	getAccumulatedCostMatix() Function	91
3	The DTWPlot() Function	92
4	The Method A: Frame Matching time warping algorithm. The algorithm is applied to each captured frame of a target motion, mapping it to the best fitting frame within a specified window of the input motion.	158
5	The Method B time warping algorithm. Determines the best input frame to map to a target frame by forecasting w frames of the target motion, then plotting an alignment between predicted target frames and a subset of frames from the input motion. The algorithm is applied to each captured frame of a target motion.	160
6	The forecastJoints() Function. Given two sequential frames of a motion, this function uses dead reckoning to predict the poses of a set of joints J over the next f frames.	161
7	The plotPath() Function. Plots an alignment path through accumulated cost matrix D , using a slightly adapted version of the established approach used in DTW.	165
8	The Method C time warping algorithm. Determines the optimal input frame to map to a given target frame, by forecasting w frames of the target motion, then using an accumulated cost matrix between the predicted frames and a subset of frames from the input motion, to determine the optimal input frame to map. The algorithm is applied to each captured frame of a target motion.	168

9	The <code>getReverseTotalCostMatix()</code> Function. Accumulates the costs in cost matrix C starting at (m, n) and ending at $(0, 0)$. The opposite direction to that used within a standard DTW algorithm.	169
---	---	-----

Acronyms

AHES Adaptive Exponential Smoothing. 145

AMC Acclaim Motion Capture data. 149

BCU Birmingham City University. 149

CCA Conical Correlation Analysis. 95

CDTW Continuous Dynamic Time Warping. 94

CMU Carnegie Mellon University. 149

CNN Convolutional Neural Networks. 49

CTW Conical Time Warping. 95

CoTW Correlation-Optimized Time Warping. 95

DDTW Derivative Dynamic Time Warping. 94

FFT Fast Fourier Transform. 64

HES Holt's Exponential Smoothing. 145

HMM Hidden Markov Model. 86

IK Inverse Kinematics. 149, 151

IMU Inertial Measuring Unit. 50

KNN k-Nearest Neighbour. 85

LMA Laban Movement Analysis. 65

LR Logistic Regression. [86](#)

MD – DTW Multidimensional DTW. [96](#)

MSN – WDTW Multiple Segmentation Norm - Weighted DTW. [96](#)

ODTW On-line Dynamic Time Warping. [98](#)

PCA Principle Component Analysis. [62](#)

PCC Pearson Correlation Coefficient. [103](#)

ROM Range of Motion. [151](#)

SNR Signal to Noise Ratio. [171](#)

SVM Support Vector Machine. [70](#)

TAM Time Alignment Measurement. [146](#)

UTW Uniform Time Warping. [87](#), [88](#)

WDTW Weighted Dynamic Time Warping. [202](#)

WTW Windowed Time Warping. [99](#)

Chapter 1

Introduction

Editing and manipulation of motion capture data is a time consuming process requiring specialist tools and skills. The need to edit motion capture data can be motivated by: i) the desire to reuse existing motion capture data, to avoid the expense of capturing and cleaning up new motion performances ([Geng and Yu, 2003](#)), and ii) the need to adapt a motion to fit the highly specific and ever changing requirements of a typical film or game production. Solutions which automatically edit a motion sequence to fit an environment or meet a specified constraint are therefore particularly desirable.

Real-time solutions, such as [Ho, Komura and Tai \(2010\)](#); [Kim et al. \(2016\)](#), which automatically adapt motion sequences to spatial changes in an environment as they occur, allow productions that utilise motion capture, more freedom to make spatial alterations. The addition of real-time temporal adaptation, would allow motion sequences to more fully adapt to the movement of a user or the performance of a live actor as they are captured, for example to interact with a performer on stage or in camera during film production. However, real-time temporal alignment, or on-line time warping, of time series data to an incomplete motion, as it is being performed, is a challenging problem to solve. Existing approaches to on-line time warping start by aligning the last known frame of the incomplete motion, then plot an alignment backwards until reaching the first frame ([Tormene et al., 2009](#); [Hülsmann et al.,](#)

2017). This approach can cause continuity issues between frames, as each new frame of the incomplete motion requires a completely new alignment, which aligns the new frame independently of the previous frame. This leads to a stuttered or non-monotonic playback of a motion, which is a problem for applications that need to visualise the motion as it is being aligned. This thesis proposes on-line time warping techniques for aligning prerecorded motion sequences, to a live motion as it is being captured, maintaining continuity between the alignment of each captured frame for a smooth playback of the motion as it is being temporally aligned.

1.1 Motivation

1.1.1 On-line Time Warping of Opposing Motions

The temporal alignment of time series data, referred to as time warping, uses techniques such as Dynamic Time Warping (DTW) (Senin, 2008) to temporally align the features of a input time series with those of a similar target time series.

The application scenarios presented below, often require a prerecorded motion sequence to be aligned with a completely different live motion. For example a virtual character’s performance being aligned with that of a live actor performing distinctly different actions, for example when a virtual character is performing opposite a live actor. In these scenarios the performance of both the virtual character and the live actor are recorded during pre-production, capturing the required performance for the virtual character and a reference of the actor’s motions. This allows the recorded reference of the actor’s motions, to be aligned to a live actor’s motion during production, whilst the time warp is being applied to both prerecorded motions, to align the virtual characters performance.

There are a number of markerless motion capture solutions, which could potentially be used to discreetly capture the motion of an actor on set or performer on stage, and re-targeted to a kinematic joint chain that matches a prerecorded motion. These include solutions based on extracting and fitting silhouettes from multiple synchronised image cameras (Michoud et al., 2007), using a single depth camera (Grest,

Woetzel and Koch, 2005), or using discrete electromagnetic and inertial sensors which could be hidden in clothing (Hu, Jin and Ni, 2012).

1.1.2 Interaction Between Performers and Virtual Characters

Film visual effects are made from a combination of real elements; such as video footage and stills; and virtual elements; such as computer generated 3D graphics and animation. Producing convincing and intricate interaction between real and virtual elements is a particular challenge, especially between live actors and virtual creatures, however, it is vital to submerging the audience within a film and maintaining their belief.

Figure 1.1 demonstrates two particularly challenging shots which require tight interaction between a live actor and a virtual creature. Tight interaction is achieved through either careful and time consuming manipulation of digital elements during post-production, for example the character's hand in the left example, or through spatially and temporally coordinating an actors performance to fit an existing digital element, for example the actor being moved in a rig to a reference in the right example. In either case the digital element does not automatically adapt itself to an actor's performance.

Innovations in camera tracking technology and real-time rendering, have made it possible to render digital elements from the correct perspective in real-time and place them into a camera's view. This approach, referred to as onset-previsualization or in camera visual effects, allows the director, actors and crew to see how digital elements fit into a live action shot as it is being filmed. Both Avatar (Schatz and Derry, 2013) and Jack the Giant Slayer (Singer, 2013), recorded motion captured performances of creatures before production, then used onset-previsualization with a camera tracking technology called Simulcam, to integrate these elements into the cameras view during filming.

Current onset-previsualization or in camera visual effects solutions focus on tracking the camera to visualise digital elements, however, there is a desire for these



Figure 1.1: Two examples of interaction between live actors and virtual creatures from the film 'Harry Potter and the Order of the Phoenix' (Yates, 2011). On the left a character's hand needs to tightly follow the movement of an actor's neck close to camera. On the right an actor is being moved in a rig to match a reference movement in a monitor.

digital elements to interact more directly with the performer, collaborating with them and allowing them to more fully utilise their talent. This inspired a number of projects including: the Dreamspace project which developed a set of virtual production tools called VPET (Spielmann et al., 2016), which allowed real-time editing of virtual elements, and #SEVEN (Claude et al., 2014) which enable digital elements such as animation and dynamic effects to be triggered by events within and actor's performance.

However, neither of the projects above focused on continuous spatial and temporal alignment of a virtual character with an actor's performance. The ability to accurately and automatically adapt a virtual character's performance on-set, to collaborate with a live actor, would allow more creative freedom to explore and experiment with ideas during production. In addition, the capability of real-time production techniques for visual effects, such as hybrid virtual production (Kadner, 2019), would be increased if more accurate and flexible real-time interactions between live actors and virtual characters were possible.

1.1.3 Additional Application Areas

There are a number of potential application areas for real-time temporal alignment of human motion, outside of onset-previsualization and virtual production.

Virtual avatars are being increasingly used in theatrical and high profile live music productions, such as the ABBA Voyage tour (Plaete et al., 2022) and bringing artists such as Prince, Tupac Shakur and Frank Zappa back to life (Stiegler, 2021). Allowing these avatars to interact with live performers on stage would open up a range of creative opportunities to engage and surprise an audience.

Real-time alignment would facilitate real-time feedback in movement training scenarios, where specific timed movements need to be learnt or performed, such as: dance (Chan et al., 2010), cardiopulmonary resuscitation (CPR), surgical procedures (Cifuentes et al., 2017), sports (Osawa, Ishikawa and Watanabe, 2020; Chantaprasert, Chumchuen and Wangsiripitak, 2019) or steps in manufacturing processes (Menolotto et al., 2020). Temporally aligning a prerecorded correct motion to a trainees motion, as a set of actions are performed, would allow real-time feedback to be given to the trainee, as well as potentially distinguish between spatial and temporal errors.

The expansion of VR and the Metaverse requires moving beyond recognising human gestures in real-time, to parameterising and aligning responses to human gestures (Wilson and Bobick, 1998; Heloir et al., 2006).

Real-time alignment could allow collaborative robots (cobots) to work more interactively with humans, potentially allowing them to anticipate when and where an interaction with person might occur. This would facilitate moving cobot applications beyond simply operating as an uncaged robots performing low-level tasks (Michaelis et al., 2020; Ateş, Stølen and Kyrkjebø, 2022).

1.1.4 Additional Capabilities

On-line alignment of human motion, could facilitate new modalities for driving and manipulating performances of virtual characters, in real-time. A number of methods

for manipulating and combining motions, which require motions to be temporally aligned, would be able to be used in a real-time scenario. For example accurately blending between an existing animation or motion sequence with a live captured performance (Kovar and Gleicher, 2003), or transferring motion styles between a live actor and a virtual character (Xia et al., 2015).

Although the on-line alignment algorithms proposed within this thesis are evaluated using human motion data, they can be applied to any time series data, regardless of whether it is singular or multi-dimensional. This opens up a variety of potential financial and medical (Tormene et al., 2009) applications.

1.2 Requirements

1.2.1 Motion Adaptation

In order to be usable, any potential real-time solutions for adapting motion sequences must satisfy the following requirements:

- Run with little latency. A study of human perception of latency in virtual character interactions (Hoyet, McDonnell and O’Sullivan, 2012), suggests that on average people typically cannot perceive delays in character interaction of less than 150ms.
- Maintain realism, with motions resulting from any adaptation or adjustment remaining physically plausible. (Reitsma and Pollard, 2003) proposes a potential metric for evaluating this.
- Capable of accurately adapting a motion sequence to fit both spatially and temporally with a live actor’s performance. As spatial alignment can only be performed after temporal alignment, it is consequently dependent on it. There have been number of models proposed for modeling the spatial relationships between virtual characters (Hwang, Suh and Kwon, 2014; Al-Asqhar, Komura and Choi, 2013; Oh et al., 2016).
- Changes in the live action environment and the actor’s actions and poses, need

to be captured and extracted in real-time.

The studies presented in this thesis focus on approaches to real-time temporal alignment of human motion. Therefore, the requirements of particular relevance to this aspect of the solution, such as low latency, accuracy and physical plausibility, will be particularly relevant, when evaluating and comparing these approaches.

1.2.2 Artistic Control

An alternative approach to adapting a prerecorded motion, is to synthesis a new motion in real-time, which fits the constraints of the environment and actor's actions. However, this approach does not always provide the level of control needed for applications such as visual effects, in which director typically directs an actor to get the performance they want.

1.3 Challenges

1.3.1 On-Line Time Warping

When performing an on-line time warp, unlike a standard offline time warp, which performs an alignment on a complete recorded motion, either one or both of the time series being aligned will only be partially known. In the use cases described above a wholly known prerecorded motion sequence is being aligned to a live partially know motion sequence as it is being captured.

Time warping partially know time series presents two challenges. First, the boundaries of the warp (i.e. the last frame of the sequence) are unknown, meaning the goal or end point of the time warp is also unknown. Second, it is not possible to determine a temporal alignment that is optimal for the entire motion, only the segment that is known.

On-line time warping algorithms must also be computationally efficient, aligning each frame faster than the sample rate of the motion capture system, which is typically 120Hz. This is particularly challenging, considering that the computational

requirements of standard time warping algorithms such as DTW are a quadratic function of the length of the motion being aligned.

1.3.2 Measuring Alignment

To compare and evaluate the performance of various time warping methods, accurate and objective metrics are required, to assess how closely aligned the temporal features of two motions are to one another.

There is not an established approach to measuring alignment. There have been a number of studies proposing (Kovar, Gleicher and Pighin, 2002) and discussing different metrics (Yang and Guan, 2005) for measuring the similarity of two motions. In addition metrics specifically intended for measuring the alignment of two motions have also been proposed (Etemad and Arya, 2015). However, evaluations and comparisons of these metrics, have focused on their ability to discriminate between similar and dissimilar motions (Valcik, Sedmidubsky and Zezula, 2016; Chan et al., 2010), rather than to measure alignment. Consequently there remains a need to determine which metrics are most optimal for measuring alignment.

1.4 Research Aim

The overall aim of this thesis is to develop and evaluate several approaches to automatic on-line temporal alignment of human motion sequences.

1.5 Research Objectives

The following objectives are to be completed in support of the overall aim:

1. Review existing approaches measuring motion alignment and trends within on-line time warping research.
2. Design a robust approach to evaluating and comparing the performance of different similarity metrics for measuring motion alignment.
3. Explore and implement appropriate approaches to measuring the similarity

and alignment of two motions, then determine which approach is optimal for measuring alignment.

4. Develop and implement solutions for on-line time warping of human motions, evaluating the accuracy and quality of the alignments produced.
5. Assess the impact of additional optimisation techniques on performance of on-line time warping solutions.
6. Assess the impact of characteristics in human motion and motion data, on the performance of on-line time warping solutions, making recommendations on how these characteristics could be managed.

1.6 Methodology

To address the aim of this thesis the following approach will be adopted. First, an optimal approach to measuring alignment will be established for use in subsequent studies. A review will be undertaken of different approaches to measuring the similarity and alignment of motion sequences. These will then be evaluated and compared to determine an optimal approach. Second, a number of novel techniques for on-line time warping of human motion will be implemented and evaluated. Additional optimisation techniques such as constraints and penalties will also be explored, as well as the feasibility of using these techniques in real-world applications. Third, explore the potential impact of movement and motion data characteristics, on the performance of the proposed on-line time warping algorithms, with a view to controlling their impact and further optimising their performance.

1.7 Approach

Due to the time based multidimensional nature of motion capture data, specialist tools are required to interpret and analysis it. The studies in the thesis required a platform that allowed specialist low level analysis and manipulation of motion capture data, while supporting the batch processing of large data sets.

Existing products such as MotionBuilder ([Autodesk, 2021](#)) incorporate a powerful user interface with a variety of tools for manipulating and reusing motion capture, while supporting automation, data analysis and motion manipulation, through C++ and Python plugins. Code libraries specialising in working with motion capture data such as [Alemi \(2019\)](#), often have sporadic development and are focused on a small set of tasks rather than providing a more general library of features. The FBX Python SDK [Autodesk \(2020\)](#), provides an extensive library for working with 3D files in Autodesk’s FBX format. FBX has become a standard for moving 3D files between content creation applications and is suited to storing 3D skeletal joint data.

To support this thesis an extensive Python library has been created to perform a variety of tasks including: re-sampling motion data; plotting motion curves; measuring similarity using a variety of metrics; measuring the correlation of motions; extracting joint parameters; and time-warping using a number of different algorithms. Additionally a number of C++ plugins were also created for use with MotionBuilder.

The approaches implemented and used in this thesis focus on geometric techniques, as they tend to be computationally efficient and allow for more control of the resulting motions. Other approaches to manipulation motion capture data can be divided into two other categories, space time constraints and artificial intelligence. Space time constraints ([Witkin and Kass, 1988](#); [Gleicher, 1997](#)), use physical models to determine realistic and plausible motions that meet specified constraints. These approaches do not work well when a motion sequence is only partially known and often take a number of iterations to reach a solution. Artificial Intelligence uses either statistical techniques such as PCA or trained models to create a solution. They typically require large training sets of motions, sometimes require experts to label motions or set-up criteria and allow limited abstract control of the results produced. These issues are particularly undesirable within a film visual effects production in which: actions are highly controlled and bespoke; recording of large sets of training motions is impractical; the solution is to be operated by a naive production crew; and a director wants to be able to control and direct the actions being performed.

1.8 Thesis Structure

Chapter 2 presents a background on the motion capture, reviewing its applications as well as the storage, interpretation and manipulation of motion capture data.

Chapter 3 reviews of approaches to measuring the similarity of two motions and time warping of human motion sequences. This includes state of the art techniques and approaches, working with multi dimensional data and on-line time warping, which are particularly pertinent to the studies in this thesis.

Chapter 4 compares a number of approaches to measuring similarity of motion data and evaluates their ability measure the similarity and alignment of two motions. This chapter presents a robust methodology for evaluating similarity metrics and makes recommendations of best practice when measuring the similarity and alignment of motions.

Chapter 5 proposes a number of novel methods for on-line time warping of human motion data, suited to the application areas described in this chapter, that plot contiguous alignment paths in a forward direction. A detailed comparison and evaluation of the performance of each method is presented.

Chapter 6 investigates the impact of two optimisation techniques, constraints and penalties, on the performance of the on-line time warping methods established in the previous chapter. Recommendations for selecting and configuring an optimal on-line time warping solution are presented.

Chapter 7 Explores the impact of characteristics and features within the captured movements and motion data, on the performance of the on-line time warping algorithms proposed in this thesis. Consideration is given as to how to optimise these characteristic to potentially improve the performance these algorithms.

Chapter 8 summarises and discusses the findings from previous chapters, detailing limitations and directions for future work.

A review of the aims of each chapter and the contributions of each chapter can be seen in Figure 1.2.

Chapter 2	Chapter 3	Chapter 4	Chapter 5	Chapter 6	Chapter 7
Aim Review current practices for interpreting and manipulating motion capture data.	Aim Review state of the art methods for measuring similarity and time warping human motions.	Aim Explore correlation as a method of measuring alignment and determine the optimal similarity metric for measuring the alignment of human motions.	Aim Evaluate the performance of on-line time warping methods based forward plotting at aligning human motions.	Aim Assess the impact of optimisation techniques on the performance of on-line time warping methods based forward plotting.	Aim Explore the impact of movement and motion data characteristics, on the performance and behaviour of on-line time warping methods based forward plotting.
Objectives Reached ①	Objectives Reached ①	Objectives Reached ② ③	Objectives Reached ④	Objectives Reached ⑤	Objectives Reached ⑥

Figure 1.2: An overview of the thesis structure, showing the aim of each chapter and it’s contribution to the thesis objectives stated in Section 1.5

1.9 Contributions

The central contribution of this thesis is the proposal of an on-line approach to time warping human motion data, temporally aligning a prerecorded performance with a live performance as it is being captured. In the process of achieving this goal, a number of other contributions have also been made:

- An evaluation of the ability of different similarity metrics to discriminate between aligned and non-aligned motions (chapter 4).
- An assessment of the impact of different approaches to correlation and parameterising joint angles on the performance of correlation based similarity metrics (chapter 4).
- A comparison between the performance of distance based and correlation based similarity metrics, when measuring the similarity and alignment of two motions (chapter 4).
- A novel approach to on-line time warping, using forecasting windows and forward plotting of alignment paths, to align human motion. (chapter 5).
- Adaptation of established contiguous local constraints to forward plotting of alignment, and representing constraint logic using state tables (chapter 6).

- Identification and evaluation of how the characteristics of a movement or artefacts within the motion data, can impact the performance of a time warping algorithm (chapter 7).

1.10 Publications

Work contributing to or contained within this thesis has been published as follows:

- A Predictive Approach to On-line Time Warping of Motion ([Randall, Williams and Athwal, 2017](#)): presents a novel approach to using forecasting within a time-warping algorithm. Although the idea of applying a time warp to a motion every frame was not taken forward into this work, the idea of forecasting the next few frames of a partially know motion was.
- Correlation as a Measure for Alignment and Similarity of Human Motions ([Randall, Harvey and Williams, 2023a](#)): presents a study evaluating correlation as a method of measuring the similarity and alignment of human motions, based on the study presented in Chapter 4.
- Online alignment of Human Motion Using Forward Plotting-Dynamic Time Warping ([Randall, Harvey and Williams, 2023b](#)): proposes and tests novel approaches to on-line time warping using forecasting and forward plotting, based on the studies presented in Chapters 5 and 6.

Chapter 2

Interpretation and Parameterization of Motion Capture Data

2.1 Introduction

Motion capture is a powerful technique for recording motion, particularly human motion. It is proven technology, regularly used in a variety of application areas including: the production of animation, film visual effects and games; training and tracking within sports ([Van der Kruk and Reijne, 2018](#)); controlling and monitoring industrial processes ([Menolotto et al., 2020](#)); and movement training for physiotherapy ([Yurtman and Barshan, 2014](#); [Walugembe et al., 2020](#)) and dance ([Jang et al., 2017](#)). The research in this thesis is particularly motivated by real-time challenges within these application areas which are discussed in more detail in [Chapter 1](#). These different application areas use a variety of different motion capture solutions, ranging from specialist optical motion capture systems to marker-less systems based on accelerometers or mono-optical solutions.

Within games and film production, an actor's motion captured performance is rarely directly applied to a virtual character without any modification, as editing and modifying motion capture data is a routine part of working with motion data in this area, either to achieve the vision of the director or to fit a motion to a particular

character or environment.

This chapter first briefly explores a variety of different motion capture solutions and presents fundamental concepts related to working with and interpreting motion capture data. It then presents a number of different approaches to parameterising motion capture data, more specifically parameterising joint rotations and the core mathematical techniques applied to joint rotations within this study. The concepts and joint parameterization methods presented in this chapter are used within the studies presented in Chapters 4 and 5. Finally this chapter presents an overview of different approaches to manipulating and editing motion capture data, providing a wider more contextual understanding of manipulating human motion, before temporal manipulation and time warping of motions are discussed in more detail in Chapter 4.

2.2 Motion Capture Solutions

Motion capture systems can be divided into four types of solutions: Optical, Image Processing and Inertial (Van der Kruk and Reijne, 2018). These are described in more detail in the sections below.

2.2.1 Optical Systems

Optical motion capture systems such as Vicon and Optitrack systems use multiple static cameras to track reflective markers attached one or more subjects within a capture volume. They are considered to be the most accurate method of capturing motion and are often used as a standard for evaluating other motion capture systems against (Freire et al., 2020; Thomas et al., 2022; Tian et al., 2015). These systems can capture the position of each marker in 3D space at high sample rates. Typically markers are applied to subject in a set pattern in order to capture the position and orientation of their joints (Vicon, 2022). Additionally, multiple markers can be applied to a rigid object to track it's orientation.

The capability of an optical motion capture system is dependent, amongst other

things, on the number of cameras used and the specification of the cameras. More cameras are required to cover larger volumes and/or avoid markers from being occluded from the view of too many cameras, when there are many subjects in the volume. Therefore optical systems are restricted to capturing the motion of a specified number of subjects within a given volume of a restricted size.

These systems are very sensitive, requiring regular calibration and making them sensitive to any movement of the cameras. They require a lot of hardware, making them expensive and not particularly portable.

2.2.2 Image Processing Systems

Advances in image capture and processing allow human motion to be extracted from video signals without the use of markers. These solutions come in a variety of forms:

- **Multi camera solutions** that track image segments from multiple viewpoints ([Patlolla, Mahotra and Kehtarnavaz, 2012](#)).
- **Single camera solutions** that use trained Convolutional Neural Networks (CNN), to estimate the joint poses of multiple subjects within the camera's field of view ([Cao et al., 2017](#)).
- **Depth-sensing cameras** such as Kinect ([Emanuel and Widjaja, 2018](#)) or the TrueDepth camera on the iPhone, that measures the distance between the camera and an array of points in front it, to create a depth image which can be processed to estimate the joint ([Zelenskaya and Harvey, 2019](#)) or facial ([Strassberger and Sikkema, 2018](#)) pose of subject. Additionally, the Leap Motion controller ([Walugembe et al., 2020](#); [Shin, Hasan and Maniruzzaman, 2022](#)), is a depth sensor that has been developed to estimate hand poses to facilitate gestural interaction with computers.

Motion capture systems based on image processing are often considered more accessible than other motion capture systems, due to the low cost of the cameras, not requiring markers and greater flexibility in terms of the portability and the environments in which they can be used. However they are less accurate and tend to

sample motion at low frequencies.

These solutions tend to use algorithms which are specifically designed to detect and track a person or part of person (i.e. face or hand), therefore, they often cannot be easily repurposed to track a different types of subject or object.

2.2.3 Inertial Systems

Inertial Systems utilise one or more Inertial Measuring Units (IMUs), which combine accelerometers and gyroscopes to sense movement and orientation respectively. Single IMU can be used to track a part of a subject such as a hand (Srivastava and Sinha, 2015) or head (Hachaj and Ogiela, 2019) or multiple IMUs can be combined to create a motion capture solution that can track the overall pose of a subject (Yang et al., 2010a; Ciklacandir, Ozkan and Isler, 2022).

While IMUs can detect motion and orientation they cannot detect position. An estimated position can be inferred from the motion information, however, this approach often results in the position of a subject drifting over time. Motion capture suits based on IMUs, also often utilise electromagnetic tracking in which a transmitter creates magnetic fields and sensors are used to detect them.

IMUs are quite versatile and minimally invasive, making them suitable for specialist applications such as hand writing recognition in a pen (Hsu et al., 2014) or tracking jaw movements in an assistive input device (Sun et al., 2021).

Although not as accurate as optical motion capture systems, especially in relation to positional tracking, they are more flexible, as they do not rely on a motion capture volume and can capture motion over larger distances.

2.2.4 Motion Capture Solutions Summary

The studies presented in this thesis are all based on motion data captured using an optical motion capture solution, however, the techniques and algorithms developed within the studies in Chapters 4, 5 and 6, can be applied to motion data captured using any of the approaches presented in this section.

Solution Type	Advantages	Disadvantages
Optical	<ul style="list-style-type: none"> • Accuracy • High Sample Rate • Can track different types of subjects or objects 	<ul style="list-style-type: none"> • Expensive • Bulky Hardware • Bound to capture volume • Affected by occlusions • Sensitive to cameras being moved
Image Processing	<ul style="list-style-type: none"> • Markerless • Inexpensive 	<ul style="list-style-type: none"> • Affected by occlusions • Often optimised to track a single type of subject • Not as accurate as optical systems
Inertial	<ul style="list-style-type: none"> • Minimally Invasive • Capture of larger ranges • Not effected by occlusions • Can track different types of subjects or objects • Can be inexpensive 	<ul style="list-style-type: none"> • Not as accurate as optical systems • Position can drift

Table 2.1: An overview comparing the advantages and disadvantages of different types of motion capture solutions.

Table 2.1 summarised the information presented within this section, outlining the advantages and disadvantages of each type of motion capture solution. Approaches have also been proposed which combine these solutions, for example combining IMU sensors and Kinect depth sensor (Tian et al., 2015).

2.3 Working with Motion Capture Data

A motion capture system, captures and stores a set values for each frame of motion. Although modern motion capture system are capable of higher frames rates, publicly available motion data-sets (Müller et al., 2007) (CMU Graphics Lab, 2001) and studies in human motion (Longo et al., 2022), typically utilised a sample rate of 120Hz.

Optical marker base motion capture systems, record the data captured for each frame in two forms (Parent et al., 2009):

- **Translational data:** the position of each marker in single global three dimensional space. Translational data contains no information on the orientation of the markers, or the hierarchy of objects or joints that the markers might be attached to.
- **Rotational data:** stored as joint orientations, typically within a skeletal hierarchy. Information on the skeletal hierarchy such as: the connection between the joints expressed as a hierarchy; the length of the bones; and the range of the motion for each joint are stored once, sometimes in a separate file as is the case of the Acclaim File Format. The rotation of each joint is recorded for each frame, in the form of rotations in local space.

The motion data-sets and captured motions used within the studies in this thesis are stored as rotational data. Therefore the mathematical approaches presented in this section are designed to work with rotational data.

The local rotation of each joint for each frame is often stored as Euler angles. Despite some of the short comings of this approach to representing joint angles, which are discussed later in this chapter, they are a data efficient approach to storing rotational information and can be easily converted to other more useful representations such as quaternions or rotational matrices.

2.3.1 Forward Kinematics

Expressing the local joint rotations as transform matrices as shown on the left of Figure 2.1, allows a captured pose to be recreated. The global position of each joint can be determined by applying each transform in order, following the kinematic chains of the skeletal hierarchy shown in Figure 2.3, from the root joint to the end effectors, in a process known as forward kinematics (Parent, 2012). Within the transform the joint rotation is represented as a rotational matrix in the red area. The conversion of Euler to rotational matrix is discussed in more detail later in the chapter. The joint length is represented as a translation in the x axis, in the form shown on the right of Figure 2.1, and applied to the blue area of the transform matrix.

$$\begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} J_l \\ 0 \\ 0 \end{bmatrix}$$

Figure 2.1: Joint data represented as a transform matrix (left). Joint rotations are represented as a rotational matrix in the red part of the matrix. The length of the joint is represented as translation in the blue part of the matrix, in the form shown on the right, where joint length J_l is represented as a translation in the x axis.

If we consider the kinematic joint chain in Figure 2.2, the global position of the end effector at j_2 can be determined using equation 2.1. R and T represent the rotation and translation elements of a transform matrix. To determine the translation of j_2 in global space, a series of matrix transforms are created following the joint chain from j_0 to j_2 , and applied by multiplying each transform in turn. The first transform represents the position and orientation of joint j_0 in global space. The second transform represents joint j_1 being translated by bone length l_0 . The translation is done within the rotation and translation space of j_0 as specified in the previous transform, before applying its own rotation, $R(j_1)$. Finally the third transform represents the end effector j_2 being translated by bone length l_1 , within the local space specified by the accumulation of the first two transforms. As j_2 is an end effector it has no rotation value hence $R(0)$.

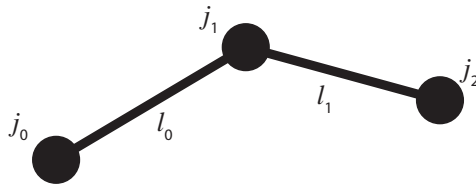


Figure 2.2: Example of a simple kinematic chain

$$T_g(j_2) = \begin{bmatrix} R(j_0) & T(j_0) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R(j_1) & T(l_0) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R(0) & T(l_1) \\ 0 & 1 \end{bmatrix} \quad (2.1)$$

In a more general form, if we consider the skeletal hierarchy in Figure 2.3, equation 2.2 can be used to determine the translation of a given joint J_k in global space, where

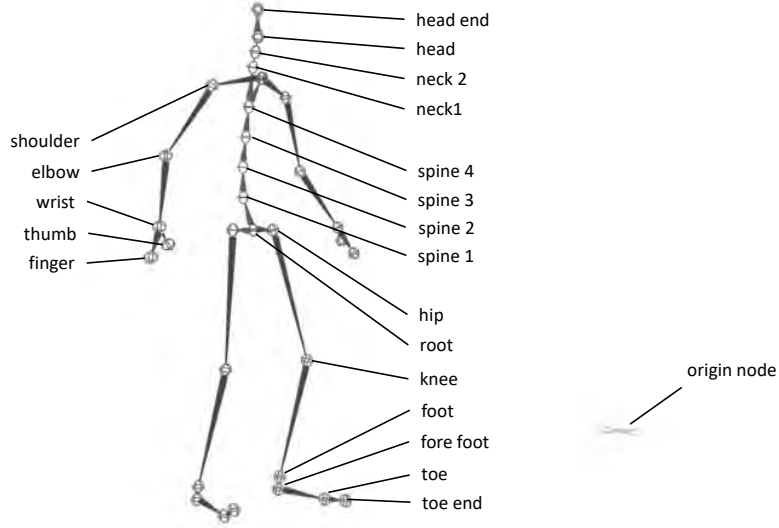


Figure 2.3: An example of a skeletal hierarchy with kinematic chains starting at the root and ending at end effectors: head end, thumb, finger and toe end.

n is the fixed origin node, r is the root joint, k is the number of joints J_k is away from the root and j_n^x is the length of the bone between joints j_n and j_{n-1} (Radke, 2013).

$$T_g(j_k) = \begin{bmatrix} R(n) & T(n) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R(Jr) & T(Jr) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R(j_1) & T(j_1^x) \\ 0 & 1 \end{bmatrix} \bullet \bullet \begin{bmatrix} R(j_{k-1}) & T(j_{k-1}^x) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R(0) & T(j_k^x) \\ 0 & 1 \end{bmatrix} \quad (2.2)$$

2.3.2 Joint Axis

When represented using Eulers, the rotation of each joint is expressed using three axis x, y, z . Changes in the orientation of a joint over time, can visualised as motion curves as shown in Figure 2.4. These are plots of the values of individual joint axis over time.

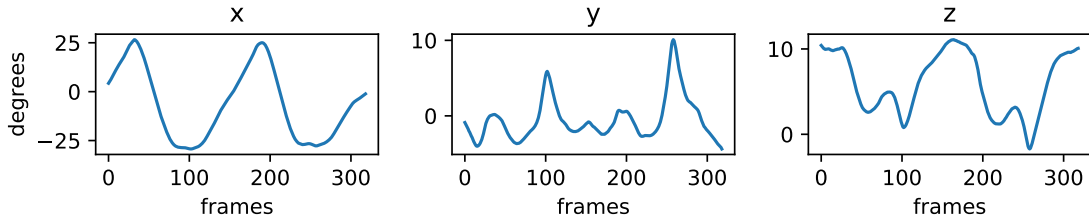


Figure 2.4: Motion curves plotted for in the x, y, z axis for the right hip joint.

Some joints have limited degrees of freedom and therefore only utilise one or two of the joint's x, y, z axis, rather than all three. Figure 2.5 shows examples of these with the elbow and knee joints which have only two and one degrees of freedom respectively. The unutilised axis is referred to as a redundant axis, and is populated with a horizontal line at 0° . Some motion capture systems, populate redundant axis with a horizontal line at 0° . Some motion capture systems, populate redundant axis by setting a single key at the start of the motion, while others will set a value of 0° for the redundant axis for every frame.

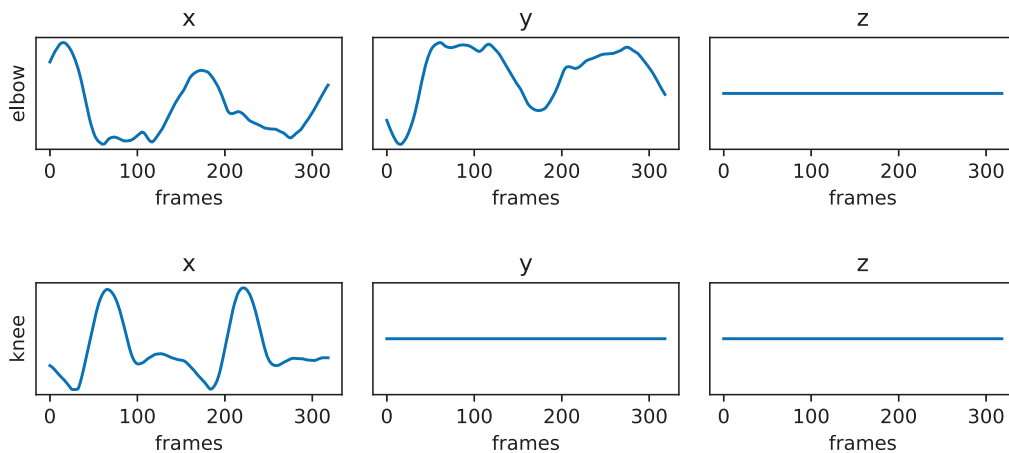


Figure 2.5: Examples of redundant axis, left unused by joints with limited degrees of freedom.

2.3.3 Re-sampling

Despite all the motion files recorded or sourced within this study being captured at 120Hz, some motion files can contain missing frames of data or redundant axis with only one key frame. To facilitate the efficient processing of motion files, there is a need to re-sample the motions and fill in any gaps in the data, ensuring that joint rotations are specified in all three axis of every joint for each frame.

Any gaps in the data can be filled in using interpolation. There are number of python libraries that can be used to interpolate values, the Autodesk Open FBX Python SDK (Autodesk, 2020) has an evaluate curve function that can be used for this purpose.

2.4 Parameterising Joint Angles

There are a variety of approaches to parametrising joint rotations in a three dimensional space. In this section five different methods of parameterising joint angles are presented and evaluated, they are either a widely used representation or have been used in previous research related to motion data.

2.4.1 Euler

Euler angles are the most widely understood representation of orientation, using three angles to specify rotations in three axis, which are perpendicular to each other. The Cartesian axis of x, y, z are commonly used, with rotations performed in a specified order.

While they are a compact, human readable representation, they have a number of weaknesses when used in computer graphics and computation. They often introduce ambiguities, as they are able to specify the same orientation in multiple ways, for example, Euler angles $(0, 0, -135)$ and $(180, 180, 45)$ both specify the same orientation. Additionally, depending on the order in which the rotations are performed in each axis, different orientations can be realised for the same set of Euler angles.

Euler angles are also notorious for gimbal lock in which two axes align in such a way, that two of the angles duplicate each others affect on orientation, resulting in the three angles only being able to affect a change in orientation in two dimensions (Parent, 2012). Angles represented in this manner are also cumbersome to work with for certain mathematical operations such as interpolation.

2.4.2 Quaternions

Alternatively a quaternion can be used to represent orientation within a three dimensional space. The motivation of using this approach is that it avoids gimbal lock and simplifies interpolation between angles. A quaternion is a hypercomplex number composed of real and imaginary numbers in the form $q = w + xi + yj + zk$ where w, x, y, z are real numbers and i, j, k are imaginary numbers. When used to

represent an orientation, i, j, k form imaginary basis vectors, allowing rotations to be expressed using four parameters $[w(x, y, z)]$, where w is the a scalar component and (x, y, z) is the vector component.

The quaternion for a given set of three dimensional x, y, z Euler angles, can be determined using Equation 2.3, where Euler rotations are applied in XYZ order.

$$Q = \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} c_x c_y c_z - s_x s_y s_z \\ c_x s_y s_z + s_x c_y c_z \\ c_x c_y s_z + s_x s_y c_z \\ c_x s_y c_z - s_x c_y s_z \end{bmatrix} \quad (2.3)$$

where $c_x = \cos(x/2)$, $c_y = \cos(y/2)$, $c_z = \cos(z/2)$

$s_x = \sin(x/2)$, $s_y = \sin(y/2)$, $s_z = \sin(z/2)$

There are also some drawbacks to quaternions, they still have potential for ambiguity, as negating every element of a quaternion will result in the same angle (i.e. $[w(x, y, z)] = [-w(-x, -y, -z)]$). Additionally, unlike Euler angles, quaternion rotations are not linear which can be problematic when determining angular differentials and means that they are not appropriate for some statistical techniques such as PCA analysis.

Quaternion mathematics make it easier to manipulate angles and perform operations such as measuring the distance between angles. The following subsections outline the more common mathematical operations performed on quaternions.

Inverting Quaternions

As well as rotation, quaternions can also specify a length or magnitude, however, the common practice when using quaternions to represent rotations, is to specify them with a unit length or magnitude. This is useful as the inverse of a unit magnitude quaternion is equal to its conjugate.

The conjugate of a quaternion, can be determined using Equation 2.4.

$$Q_* = [w(-x, -y, -z)] \quad (2.4)$$

Quaternion Dot Product

The dot product is an operation that is frequently used in other mathematical operations and can be determined using Equation 2.5.

$$Q_1 \cdot Q_2 = w_1w_2 + x_1x_2 + y_1y_2 + z_1z_2 \quad (2.5)$$

Geodesic Distance Between Quaternions

The distance between two angles can be interpreted in a number of ways. The geodesic distance is the distance between two angles if they were projected onto a sphere with a circumference of π . More specifically it is the shortest distance between the two points along the surface of the sphere.

The geodesic distance can be determined using Equation 2.6 (Krzyszowski et al., 2014). As the maximum possible distance is $\pi/2$, the distance is multiplied by $2/\pi$ to produce a normalised value where $0 \leq d(Q_1, Q_2) \leq 1$.

$$d(Q_1, Q_2) = \frac{2}{\pi} \arccos |Q_1 \cdot Q_2| \quad (2.6)$$

Quaternion Mean

The mean angle of a set of unit magnitude quaternions can be estimated as follows (Johnson, 2003; Markley et al., 2007):

1. Construct a 4 by n matrix, M , from the unit quaternions.
2. Let M_o be the outer dot product of matrix M , calculated using Equation 2.7.
3. Choose the eigenvector of M_o with the largest absolute eigenvalue, as the estimated mean quaternion, \hat{Q} .

$$M \otimes M = MM^T \quad (2.7)$$

Quaternion Hemispherization

As mentioned previously, the negative of a given quaternion represents the same angle. This double covering, or antipodal symmetry, can be visualised as two hemispheres, where all three dimensional angles can be specified within each hemisphere. Hemispherization, is a process of making sure that a set quaternions are all specified in the same local hemispherical space. Hemispherization is done using the following steps:

1. Find the mean, \hat{Q} , for the set of quaternions, S .
2. For each quaternion Q in S , if $\hat{Q} \cdot Q_i < 0$ then $Q_i \leftarrow -Q_i$.

Summary of Quaternions

Although not as human readable as Eulers, quaternions are less ambiguous and easier to perform certain calculations with, for example measuring the distance between two angles in three dimensional space is relatively straight forward using the dot product of the quaternions, which is particularly relevant to the studies within this thesis.

2.4.3 Rotation Matrices

Transform matrices are a useful and widely used method of representing orientation in computer graphics and can be used to specify not just rotations but other transformations, such as translation and scale. The 3 x 3 matrix, that specifies rotation part of the transform, as shown in Figure 2.1, can be derived from a quaternion using equation 2.8 (Dunn and Parberry, 2011).

$$R = \begin{bmatrix} 1 - 2Q_y^2 - 2Q_z^2 & 2Q_xQ_y - 2Q_zQ_w & 2Q_xQ_z - 2Q_yQ_w \\ 2Q_xQ_y + 2Q_zQ_w & 1 - 2Q_x^2 - 2Q_z^2 & 2Q_yQ_z + 2Q_xQ_w \\ 2Q_xQ_z + 2Q_yQ_w & 2Q_yQ_z - 2Q_xQ_w & 1 - 2Q_x^2 - 2Q_y^2 \end{bmatrix} \quad (2.8)$$

Matrices specify the relative transformation between two coordinate spaces, and allow different transformations such as scale and translate to be easily combined into a single matrix. Additionally the combined effect of multiple transforms can be determined by simply multiplying them together. Being able to combine transforms is useful when working with a kinematic chain of joints, for example when determining the global position of a joint from the local rotations and joint lengths of the joints in the kinematic chain.

2.4.4 Displacement Vector

A number of approaches to representing joint angles as an \mathbb{R}^3 vector in a form that avoids gimbal lock have been explored and utilised with respect to motion capture data.

Displacement vectors use a unit vector in the form of $V(x, y, z)$ to represent orientation. Given two points one unit apart at a given angle, x, y, z is the distance or displacement between the two points in their respective axis. The displacement vector, v' for a given joint, can be derived by multiplying the joint's rotation matrix R_j by unit vector $v = [0, 1, 0]$ in shown in equation 2.9.

While this representation loses the *roll* dimension of the angle, it has previously been used with PCC to measure the correlation between motions (Etemad and Arya, 2015).

$$v' = vR_j \tag{2.9}$$

where $v = [0, 1, 0]$

2.4.5 Logarithmic Map

A number of different approaches have been explored using logarithmic maps to convert quaternions to an \mathbb{R}^3 vector representation, that avoids gimbal lock and still specifies the axis and magnitude of the rotation (Johnson, 2003; Grassia, 1998). This angular representation can also be used to calculate geodesic distance between angles

(Bin, Weibin and Weiwei, 2020) and are considered particularly useful for machine learning or deep learning applications (Świtoński et al., 2010). Logarithmic maps exploit the fact that when a logarithm is applied to a quaternion, the real component w becomes zero, resulting in the definition $Q_{log} \equiv \begin{bmatrix} 0 & x & y & z \end{bmatrix}$. Equation 2.10 uses a natural logarithm as a logarithmic map to convert quaternion Q into a linear vector, Q_{ln} , where Q_{\Re} and Q_{\Im} are the respective real and imaginary vector components of Q (Johnson, 2003). Before applying this equation, quaternions do need to be on the same hemisphere.

$$Q_{ln} = \frac{1}{\text{sinc}(\arccos(Q_{\Re}))} Q_{\Im} \quad (2.10)$$

2.4.6 Rotation Vector

A rotation vector, is an \mathbb{R}^3 vector around which a rotation is performed. This representation specifies the axis of rotation but not the magnitude. This approach would not be effective for working with joint rotations as some joints only have one degree of freedom. For these joints the axis of rotation is invariant and the important magnitude variant is lost.

2.4.7 Overview of Parametrization Methods

Table 2.2 provides an overview of the characteristics and capabilities of different methods of parameterising rotations. Although rotation itself is a linear transform, within Table 2.2 *linear* refers whether the parameters are a linear function of rotation. *Unique specification* refers to whether a given angle can only be specified in one way using the parameterization method, while *Unique Interpretation* refers to whether a rotation specified using the parameterization method can only be interpreted to result in a single orientation.

Table 2.2 shows that no single method of parameterising rotations entirely full-fills all of features outlined in the table. This is fundamentally why a variety of different rotation parameterisation methods are used within the field of computer graphics.

Table 2.2: Overview of the characteristics of each method of parameterising rotations.

Parametrization Method	Linear	Prevents Gimbal Lock	Unique Specification	Unique Interpretation	No. Axis
Euler	yes	no	no	no	3
Quaternion	no	yes	no	yes	3
Rotational Matrix	no	yes	yes	yes	3
Displacement Vector	no	yes	yes	yes	2
Logarithmic Map	yes	yes	no	yes	3

2.5 Parameterising Motions

In addition to parameterising the translation and rotation of joints, there are a variety of other approaches to parameterising motion data. These parameters can represent abstract elements, such as components in a feature space (Krüger et al., 2010), or more tangible motion features such as speed and acceleration (Aouaidjia et al., 2019). As well as a constantly changing variable, parameters can also represent discrete features occurring within a motion (Müller and Röder, 2008). The approaches used to parameterise a motion can also be motivated by the sensors used to capture the motion, such as accelerometers (Hussain and Rashid, 2012) or microphones (Sun et al., 2021). It is often desirable to reduce the number of parameters used to represent a motion, either by removing less important parameters or using dimensional reduction techniques such as Principle Component Analysis (PCA) (Heloir et al., 2006). These approaches to parameterising motion are explored in more detail within this section.

2.5.1 Sensor Parameters

Rather than using a specialist motion capture solution, in some use cases it can be desirable to utilise cheaper more general purpose solutions or sensors, to capture human motion. This creates a need to translate a sensor’s data into joint position and translations or utilise other approaches to parameterising motion data.

While RAW sensor data is often not detailed or well formed enough, to directly support motion tracking for production applications, it can still often be used to recognise motions based on training sets which have been captured using the same

type or configuration of sensor.

Inertial Measuring Units

Raw sensor data from IMUs are often used to recognise motions in mobile phones either using purely accelerometer data on older phones (Kratz, Rohs and Essl, 2013), or combining accelerometer and gyroscopic data available in more modern hand held devices (Min, Choe and Cho, 2010). IMU sensor data has also been utilised to recognise handwriting movements in a pen (Hsu et al., 2014), head movements in a VR display (Hachaj and Ogiela, 2019) and hand movements in tennis (Srivastava and Sinha, 2015).

Sensor data can be treated with filters to reduce the impact of anomalies or noise in the sampled data, or normalise and segment it to standardise the data (Janaki, Babu and Sreekanth, 2013).

To cope with variances between how a user might hold a mobile device, fit an IMU sensor or change their overall posture when performing a movement, a solution called Protractor3D was proposed to compensate for this in accelerometer data. Protractor3D finds optimal registration points with which to fit two sets of 3D accelerometer data (Kratz and Rohs, 2011), a technique later applied to gyroscopic sensor data (Kratz, Rohs and Essl, 2013). When capturing the joint poses of a person using IMU sensors a similar problem occurs with the root joint, which is defined in global space. This can be solved by comparing the gyroscopic data for each motion to a vertical plumb line, for example the y axis, (Yang et al., 2010a). For other joints this is not an issue as they are typically defined in local space in relation to a parent joint.

Image Sensors

Image sensors, provide video data in a two dimensional plane. For motion capture applications there is typically a need to extract three dimensional information from this plane, either through multiple cameras, depth sensors or artificial intelligence techniques. However, some gesture recognition applications are designed too work

with two dimensional data (Christensen, Jebara and Pentland, 1999; Tan et al., 2010) or extracted features only expressed in two dimensional space (Osawa, Ishikawa and Watanabe, 2020), eliminating the need to determine or estimate the position of joints of features in the third dimension.

Microphones

Microphones are a cheap and readily available sensor on many devices. The amplitude of the sound at different frequencies, obtained using Fast Fourier Transform (FFT) can be treated as a set of features (Sun et al., 2021).

2.5.2 Feature Parameters

It is often desirable to extract parameters that represent specific features of a motion, either to improve the performance or accuracy of a solution using the motion data, or to add useful semantic information to a motion.

Velocity

One of the simplest features to extract from motion data is joint velocity, which is the differential of the joints position or rotation. For a given joint axis or feature, j , at frame, f , a differential can be obtain using $\Delta(j_t) = j_t - j_{t-1}$. Another approach is to determine the transform of a joint, as a transform matrix, using the Kabsch algorithm (Aouaidjia et al., 2019).

A common motivation for using joint velocity is that motion recognition solutions based on the magnitude of a velocity do not require motions to be spatially aligned (Aouaidjia et al., 2019), which is useful when working with three dimensional points defined in a single global space. Velocity is also a useful feature on which to segment a motion, as low velocity or changes in velocity often define the end of one movement or gesture and the beginning of another (Yabe and Tanaka, 1999).

Kovar, Gleicher and Pighin (2002) takes the positions of a set of joints in every frame over a window of time, to form a point cloud. Point clouds represent both the pose and movement of a motion within a single set of parameters.

Relational Features

Features associated with the relationship between the joints within a given motion can also be parameterised. It is important for features to be based on the relationships between joints within the skeletal hierarchy, so that they are independent of the global position and orientation of each motion.

Example relational features include the distance between joints which are not adjacent to each other in the kinematic chain (Aouaidjia et al., 2019) or parameterising the correlations between joints (Zhao et al., 2020; Yang et al., 2010b). Relational features encode how the movement of one joint relates to the movement of another, allowing those relationships to be maintained when editing a motion. Rather than joint data existing as a multivariate time series, relationships can be determined between these variables.

Müller and Röder (2008) proposed a set of general approaches for extracted relational Boolean features from a motion, which can be seen in Table 2.3. The features are based on movements crossing specified positional, angular and velocity thresholds, resulting in a set of Boolean values for each feature. A set of 39 features have been specified, using the approaches in Table 2.3, for encoding the relational features within a motion.

Labelled and Categorizing Features

Motions can be analysed in order to label or categorize different segments of motion. For example Laban Movement Analysis (LMA) is often used to attach movement descriptions to dance motions that describe aspects such as body, effort, shape and space (Jang et al., 2017; Bao and Yao, 2020). An approach has been developed by Aristidou et al. (2015), that utilises the Boolean features proposed by Müller and Röder (2008), and presented earlier in this section, to perform an LMA analysis.

Another approach to labeling motion segments is to use solutions such as speech to text (Saund et al., 2022) to capture additional data to augment the motion data.

Table 2.3: General approaches to detecting relational features in human motion, as defined by Müller and Röder (2008)

	<p>Crossing 3 point plane: $plane(j_1, j_2, j_3, j_4)$ Evaluates if joint j_4, shown in red, is in front of or behind a plane which intersects joints (j_1, j_2, j_3), shown in blue. For example a plane intersecting the root, right hip and right foot joints, could be used to evaluate if the left foot is front of the right foot in walk cycle.</p>
	<p>Crossing normal plane: $nplane(j_1, j_2, j_3, j_4)$ Evaluates if joint j_4, shown in red, is in front of or behind a plane which is perpendicular to the vectors between joints j_1 and j_2, shown in green and intersecting joint j_3. For example a plane perpendicular to the bone going from the chest to the neck and intersecting the neck, could be used to evaluate if the right hand is above the neck.</p>
	<p>Angular threshold: $angle(j_1, j_2, j_3, j_4, \theta)$ Evaluates if the angular distance between the vectors going through joints j_1, j_2 and j_3, j_4, shown in red, is below the threshold set by θ. For example, if the angular distance a, between the bones going from the right hip to the right knee and from the right knee to the right foot, crosses below a given threshold, this indicates that the right knee is bent.</p>
	<p>Move threshold: $move(j_1, j_2, j_3, \theta)$ Evaluates if the movement of joint j_3, shown in red, is moving in the direction of the vector between joints j_1, j_2, shown in green, above a threshold set by θ. For example, the vector between the chest and neck, could be used to evaluate if the left hand is moving upwards beyond a certain speed.</p>
	<p>Move threshold (using normal): $nmove(j_1, j_2, j_3, j_4, \theta)$ Evaluates if joint j_4, shown in red, is moving along the normal of the plane intersecting joints (j_1, j_2, j_3), shown in blue, above a threshold set by θ. For example the normal of plane going through the neck, right hip and left hip, could be used to evaluate if left hand is moving forwards.</p>
	<p>Speed threshold: $fast(j_1, \theta)$ Evaluates if joint j_1, shown in red, is moving faster than the threshold set by θ. For example, evaluating when the speed of a foot exceeds a threshold to detect phases in a kicking motion.</p>
	<p>Touch threshold: $touch(j_1, j_2, \theta)$ Evaluates if joints j_1 and j_2, shown in red, are within the distance of each other set by θ. For example, evaluating the distance between hands to detect phases in the a clapping motion.</p>

Motions are often labelled and segmented so that they can be re-sequenced to synthesise new motions. [Rose, Cohen and Bodenheimer \(1998\)](#) segments motions into *verbs* and *adverbs* allowing new motions to be synthesised in an intuitive manner.

Dimensional Reduction

The high dimensional nature of motion capture data and features extracted from motion capture data, often motivates the use of dimensional reduction techniques. Generalised dimensional reduction techniques such as: PCA (Principle Component Analysis) ([Heloir et al., 2006](#); [Johnson, 2003](#)), which identifies the principle eigenvectors within a features space; or Isomaps ([Seward and Bodenheimer, 2005](#)) which identifies transforms between features, can be applied to motion data.

[Heloir et al. \(2006\)](#) split a motion into hands, upper body and lower body zones and utilised PCA to determine the eigenvector with the largest eigenvalue in each zone. This approach reduced the number of dimensions, but also allowed each zone to be weighted independently. Deep learning techniques have also been used for dimensional reduction, with Neural Networks being used to encode multidimensional data ([Xiao and Chu, 2017](#)).

2.6 Motion Editing

A variety of techniques can be used to manipulate, blend and adapt human motion data. This section explores some of the motivations for editing motion data before reviewing the main approaches to editing motion data, starting with more fundamental approaches such as warping motions and applying constraints, before presenting more complex techniques such as blending and synthesising motions.

[Xiao, Zhang and Bell \(2004\)](#) identifies the following motivators for editing or controlling motion data:

- Attaining an accurate spatial and temporal alignment of the motion to a real or computer generated environment or apposing character's actions.
- Overcoming the spatial restrictions of a motion capture volume.

- Allowing motion data to be reused on different character's or in different environments.
- Seamlessly blending and combining motion data to synthesis new motions.

2.6.1 Editing Motion Signals

The movement of each joint axis is represented in the form time series data, referred to as a motion curve and shown in Figure 2.4. Editing a motion curve can be considered a signal manipulation problem, allowing a variety signal manipulation and filtering techniques to be applied to motion editing (Bruderlin and Williams, 1995).

Motion curves can be displaced to edit a motion, as offsetting the values of a motion curve will result in a change in a joints orientation. While it is rarely desirable to offset the values of an entire motion curve by a single fixed value, offsetting the joint by the same amount for the entire duration of the motion, it useful to be able to offset a motion curve by varying amounts, as required, at different points in the motion curve. Witkin and Popovic (1995) proposed specifying displacements or offsets using a sparse set of key-frames, allowing the motion curves to be displaced as required.

Time warping can be used to temporally warp a signal, to temporally displace the data points in a motion curve. Uniform Time Warping (UTW) (Fu et al., 2008) can be used uniformly scale the duration of a motion to make it faster or slower. Monotonic temporal warps can also be specified using key frames to create a time warp curve (Witkin and Popovic, 1995; Kovar and Gleicher, 2003). Dynamic Time Warping (DTW), warps each frame of a motion to optimally align the features of one motion with another (Bruderlin and Williams, 1995; Sakoe and Chiba, 1978). Time warping is a key step in blending motions and transferring styles and features between motions. Time warping algorithms are presented and discussed in more detail in Section 3.3. They also a key topic within the studies presented in Chapters 4, 5 and 6.

Frequency filters can be applied to motion curves to separate low frequency gross motion patterns from small high frequency motion elements. This allows the overall movement of a motion to be edited without effecting high frequency details (Lee and Shin, 1999), or for high frequency stylistic characteristics of a motion to be extracted (Unuma, Anjyo and Takeuchi, 1995).

Filters can also be applied to a motion curve, such as low pass filters to smooth a motion (Yu et al., 1999; Fazlali et al., 2020) or shape functions can be applied to clamp or accentuate a motion's characteristics (Bruderlin and Williams, 1995).

Neff and Fiume (2003) utilises signal processing techniques to edit more semantically meaningful motion attributes such as the extension (distance of joints from hips) and amplitude (range of movement) of a motion.

2.6.2 Applying Constraints

Constraints are a widely used approach in computer animation. Parent/Child constraints can be used to attach one element to another, for example attaching an axe to a character's hand. Aim constraints can be used to rotate an object to face another object for example aiming a camera at a character. Constraints are supported by most animation tools, using the ideas set out by Cohen (1992), for example allowing the weighting of a constraint to be adjusted at different points in the animation (Choi, Park and Ko (1999)).

Constraints are often used when re-targeting motion data (Choi and Ko, 1999), a process in which the motion data is re-targeted from one set of kinematic joints to another, a process that is usually required when transferring motion data from one character to another. A challenge within re-targeting is utilising the destination joints as efficiently as possible when reproducing the motion, to minimise the amount by which joints are manipulated.

Re-targeting a motion from one set of joints to another, invariably means the motion is being reproduced in a set of joints of different sizes to the original motion. This means a decision has to be made as to whether the objective is to reproduce the pose of

the motion, or the position of end effectors, in order preserve any interaction with the environment. [Shin et al. \(2001\)](#) proposes an approach to determining when it is important to preserve the position of end effectors or the original pose of the motion.

Physical constraints, referred to as spacetime constraints ([Witkin and Kass, 1988](#)), can be imposed to ensure that a motion remains physically plausible. These could be in the form of the range of motion of each joint or the maximum speed at which a joint can move or accelerate ([Gleicher, 1997](#)).

While individual spatial constraints are relatively straightforward to specify, a generalised approach to detecting interactions and maintaining spatial constraints is a more complex problem. Spatial constraints can be identified and specified base local minima distances between character joints and objects being interacted with or using zero crossings in velocity to detect changes in movement direction ([Bindiganavale and Badler, 1998](#)). Interactions can also identified using trained data to recognise motions classified using a Support Vector Machine (SVM) ([Oh et al., 2016](#)). Volumetric meshes have been used to detect spatial interactions with more complex environments or objects, which may involve multiple points of contact, such as a character sitting in a chair, riding a bike or getting into a car ([Ho, Komura and Tai, 2010](#); [Kim et al., 2016](#)).

In some scenarios a motion is required to satisfy multiple constraints, for example when simultaneously re-targeting a motion, maintaining an eye-line, adjusting a wall interaction to a spatial change in the environment and making sure joints do not move to fast. This has motivated solutions for determining the correct or most effective way of prioritising these constraints ([Zhu, Wang and Xia, 2006](#); [Boulic et al., 2003](#)), including an approach that also incorporates simulation of dynamics on a character using the Open Dynamics Engine ([Li et al., 2009](#)).

2.6.3 Editing via Abstract Models

Kinematic joint hierarchies are not the most intuitive or straightforward model to work with, this has motivated the development of simplified models for representing motion, which are potentially more natural to work with. Physical models of

motions have been abstracted from motions to support more physically plausible manipulation of motions, including an inverted pendulum model representing the balance and center of mass of a motion along with joint torques (Tsai et al., 2009), and modelling of contact forces when walking (Ye and Liu, 2010). There has also been some interest in editing motions in low dimensional space, motivated by the fact that this would leave connections between the movement of independent joints intact (Carvalho, 2009).

Models have also been created to model the physics of soft tissue and skin to deform the mesh of a character (Loper et al., 2015).

2.6.4 Blending and Synthesising Motions

There is often a desire to blend together a number of short motions to create a single longer motion, either to get around the limitations of the small capture volume or to synthesis new motions. In order to blend motions they need to be accurately aligned, both spatially and temporally (Kovar and Gleicher, 2003; Kovar, Gleicher and Pighin, 2002). To support this there is a need to extract key frames and segment motions to create libraries of motions that can be blended together. Segmentation can be based on identifying key features (Lim and Thalmann, 2001; Müller, Röder and Clausen, 2005; Janaki, Babu and Sreekanth, 2013) or recognising sub-sequences within a larger motion or stream of motion data (Li and Prabhakaran, 2005; Xiao and Liu, 2015).

As well as concatenating motions, movement styles can also be transferred between motions, taking the motion style of one motion and applying it to the gross movements of another. Motion style transfer solutions have been developed based on the following techniques: identifying and transferring the difference between motions (Hsu, Pulli and Popović, 2005); using time warping and linear transforms to best fit one motion to match another Xia et al. (2015); or using trained neural networks (Tao et al., 2022; Smith et al., 2019). A key step of style transfer is accurately time warping motions to align their temporal features, allowing style attributes to be transferred between the two motions.

2.6.5 Impact on Perceived Realism

It is important to understand the impact of editing and manipulation of motion data on the perceived realism of the motion. [Pražák, McDonnell and O’Sullivan \(2010\)](#) evaluated the impact of time warping on the perceived realism of a motion and found that slowing down a motion was more acceptable than speeding it up. Changes to ballistic movements such as jumping are particularly perceptible, motivating a study into the impact of editing of a throwing animation on perceived realism ([Vicovaro et al., 2014](#)) and the development of metric for estimating the perceived realness of ballistic motions ([Reitsma and Pollard, 2003](#)).

2.7 Summary

Human motions can be encoded in a wide variety of formats and parameters to produce motion data, which is complex to work with, being both spatial and temporal in nature and typically multidimensional. It is desirable and often necessary to be able to manipulate, edit and adapt motions, either for stylistic purposes, or to spatially or temporally fit the motion to a meet new constraints, either to align contact points or simulate interactions.

This chapter has reviewed a variety of approaches to parameterising joint data as well other motion features. In addition several methods for editing motion have been explored. Temporal alignment of interactions and motion editing methods, such as motion blending and style transfer, are dependent on accurate time warping methods to aligning the temporal features of motions. The following chapter reviews a number of approaches to time warping human motions, the majority of which are based on measuring the similarity of motions, which are also explored at the start of the same chapter.

Chapter 3

Measuring Similarity and Time

Warping of Human Motions

3.1 Measuring Similarity

3.1.1 Introduction

The ability to measure similarity of motions is a key tool in motion retrieval and motion editing. Many techniques and approaches to re-using and combining motion captured data are dependent on reliable methods of identifying motions within a motion database, which are similar to a query motion. This has led to development and use of similarity metrics, equations and algorithms which measure and score the similarity of two motions for a variety applications.

A similarity metric based on the positional and rotational differences between the respective joints of two motions, has been used to retrieve motions from a database to control characters in a virtual environment ([Lee et al., 2002](#)). A metric based on the point clouds of joints over a window of time, has been used to retrieve motions to be blend together in a motion graph ([Kovar, Gleicher and Pighin, 2002](#)). [Müller and Röder \(2008\)](#) extracted Boolean features from a motion to allow motions to be retrieved from a database more efficiently. [Guerra-Filho and Bhatia \(2011\)](#) compared the performance of similarity metrics based on PCA, rotation distance,

points clouds and Boolean features, at correctly classifying motions in a database, and found that point clouds performed best. A proposed or selected metric is often optimised for a particular application.

3.1.2 Types of Similarity Metric

There are a number of different metrics that can be used to measure the similarity of two motions, which can be separated into the following categories:

- **Distance Based Metrics** are the most established approach to measuring similarity. They are based on measuring the differences between particular parameters of corresponding joints, such as: position; rotation; velocity; and acceleration. Using normalisation, multiple parameters can be combined within the same metric.
- **Correlation Based Metrics** assess similarity using the correlation of corresponding parameters rather than the distance between them. They evaluate the variation between the overall forms of the motion curves associate with corresponding joint parameters, rather than the differences between the values that make up the motion curve.
- **Feature Based Metrics** assess similarity based on the occurrence of discrete features within the motion such as: feet crossing; hand above head; and foot or hand in front or behind the hips.

Regardless of the method used to parameterise the motions, each parameter can be treated as a time series of n data points, where both motions consists of n frames, defined by $t = 1, \dots, n \in \mathbb{N}$. The similarity of the two motion sequences, A and B , is measured by comparing the parameters of respective frames in each motion, (A_1, A_2, \dots, A_n) , and (B_1, B_2, \dots, B_n) .

3.1.3 Preparing Data

The steps required to prepare for motion data before applying a similarity metric, will vary depending on the use case but can be broadly outlined as follows ([Janaki](#),

Babu and Sreekanth, 2013):

1. **Sampling:** The capture of the motion. It is important to consider the sample rate the device is capable of and in some cases the sensor delay or latency.
2. **Filtering:** A smoothing filter can be applied to motions to smooth out potential errors or noise in the sample data, for example applying a Butterworth filter (Yu et al., 1999).
3. **Segmentation:** Segmenting the motion based on features within the motion such as: pauses between actions identified by low joint velocities (Yabe and Tanaka, 1999), or changes in Boolean features defined using thresholds Müller and Röder (2008) as described in Section 2.5.2.
4. **Conforming Data:** Making sure that the motions being compared have same number of samples or frames. Making sure that the joint data is represented consistently in both motions (i.e. using the same joint system). Conforming motion data is outlined in more detail in Section 3.1.4.
5. **Normalization:** The parameters for each joint or feature sometimes need normalising to insure the weighting of each joint is independent of the magnitude or range motion occurring in the joint.

3.1.4 Conforming Motions

To facilitate these similarity measurements, both motions must have the same number of frames (i.e. the same duration and sample frequency) and use the same joint system. This typically requires one of the motions being compared to be adjusted to conform to the length and joint system of the other.

The length of a motion can be changed using a time warping algorithm such as: Uniform Time Warping (UTW) or Dynamic Time Warping (DTW). UTW uniformly stretches or squashes a motion, then re-samples the frames at the required frequency. DTW monotonically aligns the frames of an input motion to best match the frames of a target motion, deleting and duplicating frames of the input motion as required. Which method is used will depend on the use case scenario. Both of these time

warping approaches are described in more detail in the Section 3.3.

As mentioned previously, similarity metrics can only reliably compare motions applied to the same joint system or skeletal hierarchy. Differences in the joint systems will affect the joint angles used to encode character poses, resulting in an identical pose on two different joint systems being defined using different joint angles. A re-targeting algorithm, such as the one built into Autodesk MotionBuilder (Autodesk, 2021) can be used to plot motions onto different joint systems.

3.1.5 Joint Weightings

Motion data is multidimensional in nature, for example a simple joint rig of 18 joints, each encoded with 3 parameters, would use 54 dimensions to define a character's pose in a given frame. Any measurement of similarity between motions is therefore based on the aggregate of multiple joints.

Similarity metrics typically allow joints to be weighted, so that some joints can be equally weighted or given a higher or lower importance when measuring similarity. A reasonable assumption would be to give joints which have a higher impact on a pose or are particularly salient to general human motion such as: shoulders; elbows; hips; and knees, more importance than other joints (Lee et al., 2002), and ignore less significant joints such hands and feet. These less significant joints tend to be peripheral and more invariant to the motion being performed, while having a limited range of movement, this has motivated giving joints that move the most more weighting (Lee and Lee, 2016; Patrona et al., 2018).

Studies have been performed to determine the optimal set of joint weightings to use with a similarity metric. Wang and Bodenheimer (2003) used a constrained regression technique to determine a set of optimised joint weightings to minimise the cost of transitioning from one motion to another, while Harada et al. (2004) determined a set of optimal joint weightings based on viewer perception tests. Both sets of optimal joint weightings can be seen in Table 3.1. The weightings determined using human perception (Harada et al., 2004) ranged between 0 and 50, and were based on the perception of changes in joint position. To aid comparison, these

weightings were scaled to a range between 0 and 1 and the weights for a given joint swapped for weighting of it’s parent joint, as the Wang and Bodenheimer (2003) study was based joint rotations and a joint’s position is largely a derivative of its parent’s rotation.

Table 3.1: Optimal weighting of joints within a similarity metric, determined by a regression(Wang and Bodenheimer, 2003) and human perception(Harada et al., 2004).

Joint	Determined by Regression	Determined by Human Perception
Hips	1.000	0.667
Knees	0.090	0.333
Ankles	-	0.067
Shoulders	0.788	1.000
Elbow	0.025	0.000
Neck	-	0.333
Head	-	0.667

The weighting assigned to each joint, w_j , with a set of m joints, is typically in the range of $0 \leq w_j \leq 1$, where $\sum_{j=1}^m w_j = 1$.

3.1.6 Distance Base Metrics

Distance based metrics evaluate the similarity of two motions by extracting an identical set of parameters from the joints of both motions, then measuring the amount of difference between the respective parameters as a cost function.

Distance based metrics typically score a given joint parameter, based on the average difference between the corresponding values of the same parameter in each motion, across all frames. This means that joints that contain a larger variation between the two motions, will have more impact or weight on the overall similarity score than joints with less variation between the two motions. Therefore, in order to equally weight joints or control the weighting of each joint, the differences between each set of joint parameters need to be normalised across the entire set of motions being compared.

In this section distance based metrics based on the following joint features: rotation; position; and velocity will be presented. These metrics are based on measuring

the difference between features which are represented using multiple parameters. Although not explored in this section, there are also a variety of techniques for evaluating the difference between individual sets of parameters (Akila and Chandra, 2013).

Rotational Distance

A common approach to determining the distance between two angular rotations (Krzeszowski et al., 2014), is to use quaternions and take the absolute value of their inner product $|q_a \cdot q_b|$, giving the geodesic distance between them. The angular distance between motion sequences A and B , of length n frames, with m joints, with w weighting, can be determined using equation 3.1. The dot product is multiplied by $\frac{2}{\pi}$ to normalise the rotational distance to between 0, matching, to 1, opposite.

$$c_\theta = \frac{\sum_{f=1}^n \sum_{j=1}^m w_j \frac{2}{\pi} \arccos |Q_j^{A_f} \cdot Q_j^{B_f}|}{n} \quad (3.1)$$

Positional Distance

When measuring the similarity of two motions using position it is often desirable to parameterise positions in local space, relative to the hips, rather than in global space. This eliminates the need to align joint rigs before measuring their similarity. Moreover, most applications that require aligned or similar motions, such as: motion blending; motion recognition and translating motion styles, do not have an explicit need for the hip joint to be considered. To determine the position of a joint with respect to the hip, the transform of the hip joint needs to be removed by reversing it. This can be done by multiplying the inverted transform matrix of the hip joint, H^{-1} , by the global position of a given joint $T_g(j)$, giving the local translation, $T_h(j)$, between the hip and joint j . This allows the similarity between the joint positions of motions A and B , to be assessed by measuring the difference between the hip to joint translation, $T_h(j)$, of corresponding joints in equation 3.2.

$$c_p = \frac{\sum_{f=1}^n \sum_{j=1}^m w_j |p_j^{A_f} - p_j^{B_f}|}{n} \quad (3.2)$$

$$\text{where } p_j = T_h(j) = T_g(j)H^{-1}$$

Difference in Velocity

A velocity distance metric is based on comparing the change in a given parameter or vector over time. Measuring the change of a single parameter only provides information the magnitude of the change (i.e. speed of movement), it is more desirable to measure the change in a vector as this will give a velocity, which has information on both the speed and direction of a change.

Equation 3.3 demonstrates how the rotational distance metric in equation 3.1, can be adapted to create a metric based on the difference in rotational velocity, where $\Delta R_j^{\Upsilon_f}$ is the rotational velocity of a joint j within motion Υ , represented as a transform matrix. The rotational velocity is determined by multiplying a joint's rotation in a given frame, by the inverse of its rotation in the previous frame. The rotation in the matrix transform $\Delta R_j^{\Upsilon_f}$ is converted to quaternion $\Delta Q_j^{\Upsilon_f}$.

$$c_{\Delta\theta} = \frac{\sum_{f=1}^n \sum_{j=1}^m w_j \frac{2}{\pi} \arccos |\Delta Q_j^{A_f} \cdot \Delta Q_j^{B_f}|}{n} \quad (3.3)$$

$$\text{where } \Delta R_j^{\Upsilon_f} = R_j^{\Upsilon_f} (R_j^{\Upsilon_{f-1}})^{-1}$$

Equation 3.4 is an adaptation of the positional difference similarity metric in Equation 3.2, creating a metric based on the difference in the positional velocity of each joint. The positional velocity is determined by magnitude of the translation resulting from subtracting the translation between the hip and joint j , $T_h(j)$, in the previous frame from that of the current frame.

$$c_{\Delta p} = \frac{\sum_{f=1}^n \sum_{j=1}^m w_j \|\Delta p_j^{A_f} - \Delta p_j^{B_f}\|}{n} \quad (3.4)$$

where $\Delta p_j = T_h(j)^f - T_h(j)^{f-1}$

[Aouaidjia et al. \(2019\)](#) proposed combining global and local joint velocities in a single metric. The global joint positions were determined using Equation 3.2, which subtracts the global hip joint transformation from a given global joint position to compensate for any difference in the overall orientation of the motion. The local joint positions were determined relative to the parent joint, which is analogous to a Displacement Vector as discussed in Section 2.4.4. Rather than determine the velocity by subtracting the parameters of adjacent frames a Kabsch algorithm was used to determine the transform of each joint between frames.

Cosine Similarity

A more generic approach to measuring the similarity of vectors is to use cosine similarity, these can be applied to any feature vector, such as a Cartesian coordinate representing a position in three dimensional space. Cosine similarity projects one unit length vector onto another, giving a measure of the difference between the orientation of the two vectors, where -1 is the opposite orientation, 0 is perpendicular and 1 is the same. Both [Hegarini et al. \(2016\)](#) and [Chantaprasert, Chumchuen and Wangsiripitak \(2019\)](#) used this approach to measure cosine similarity of joint rotations parameterised as displacement vectors, described in Section 2.4.4.

The cosine similarity of two vectors, A and B , can be calculated using Equation 3.5. If the vectors are both unit length then the Equation 3.6 can be used. Equation 3.5 can be adapted into similarity metric shown in Equation 3.7, where where $V_j^{\Upsilon f}$ is the displacement vector of joint j at frame f of motion $\vec{\Upsilon}$.

$$\cos\theta = \frac{\vec{A} \cdot \vec{B}}{|\vec{A}| \cdot |\vec{B}|} \quad (3.5)$$

$$\cos\theta = \vec{A} \cdot \vec{B} \quad (3.6)$$

$$c_{cos} = \sum_{f=1}^n \sum_{j=1}^m w_j \frac{\vec{V}_j^{Af} \cdot \vec{V}_j^{Bf}}{|\vec{V}_j^{Af}| \cdot |\vec{V}_j^{Bf}|} \quad (3.7)$$

Combining Parameters

Both [Lee et al. \(2002\)](#) and [Arikan and Forsyth \(2002\)](#) propose similarity metrics based on multiple parameters, respectively combining joint rotation with joint velocity and joint position with joint velocity. This can be achieved by combining the equations already presented in this section as demonstrated in the Equation 3.8. Both rotational and positional distance metrics evaluate the difference in the overall pose of the joints, therefore combining both in the same metric would potentially place too much weight on the pose of the joints over the movement of the joints. However, as shown Equation 3.8, the parameters can be weighted to compensate for this or prioritise parameters which may be more pertinent to a given scenario.

$$c = w_{\theta}c_{\theta} + w_{\Delta p}c_{\Delta p} \quad (3.8)$$

When comparing the similarity of hand features extracted from a video [Tan et al. \(2010\)](#) combined joint shape and velocity features using key frames. Shape features were only compared on a sparse set of key frames, while the velocity of the hand was compared on every frame.

[Janaki, Babu and Sreekanth \(2013\)](#) utilised a similarity metric which combined DTW distance and Mahalanobis distance ([Moonen and De Moor, 1995](#)), to evaluate the similarity of signals from accelerometer and gyroscope sensors in mobile phones. The DTW distance, is a measure of how much a motion needs to be time warped

to align it with another motion. It is the distance of the alignment path plotted using Dynamic Time Warping, which is discussed in Section 3.3.3, from a linear alignment. Mahalanobis distance is based on the mean and covariance of a given set of joint parameters and provides a distance measure that is independent of the range motion of a particular joint, eliminating the need for normalisation.

Point Clouds

Kovar, Gleicher and Pighin (2002) create point clouds from the joint positions of a motion at a given frame and its neighbouring frames. The positions of the points are all defined in the same co-ordinate space and represent both the pose (position) and movement (velocity and acceleration) of a motion over a small window of time. To eliminate the effect of any differences between the position and orientation of the two motion's root joints, a transformation (T) is applied to the point cloud of one motion, optimising the fit (sum of the squared distances) between the two sets of point clouds. The transform is constrained, only allowing the point cloud to be rotated around the y (vertical) axis and translated horizontally in the x and z axis. Once the optimization transform is applied, the distance between the respective points in each motion can be summed using equation 3.9, giving a measure of similarity, where ω is the number of neighbouring frames to be included each side of the current frame, w_p is the weighting for the point p in the point cloud and A_p and B_p are points in the point cloud created from motions A and B respectively.

$$c_{pc} = \sum_{f=1}^n \sum_{p=1}^{m(2\omega+1)} w_p |A_p - T_{\theta,x,z} B_p| \quad (3.9)$$

The optimal transform T has the following closed form solution:

$$T_{\theta} = \arctan \frac{\sum_{p=1} w_p (x_{A_p} z_{B_p} - x_{A_p} z_{B_p}) - \frac{1}{\sum_{p=1} w_p} (\bar{x}_{A_p} \bar{z}_{B_p} - \bar{x}_{B_p} \bar{z}_{A_p})}{\sum_{p=1} w_p (x_{A_p} x_{B_p} + z_{A_p} z_{B_p}) - \frac{1}{\sum_{p=1} w_p} (\bar{x}_{A_p} \bar{x}_{B_p} + \bar{z}_{A_p} \bar{z}_{B_p})} \quad (3.10)$$

$$T_x = \frac{1}{\sum_{p=1} w_p} (\bar{x}_{A_p} - \bar{x}_{B_p} \cos(T_\theta) - \bar{z}_{B_p} \sin(T_\theta)) \quad (3.11)$$

$$T_y = \frac{1}{\sum_{p=1} w_p} (\bar{z}_{A_p} + \bar{x}_{B_p} \sin(T_\theta) - \bar{z}_{B_p} \cos(T_\theta)) \quad (3.12)$$

A potential weakness of this approach is the 2D optimization transform $T_{\theta,x,z}$. It assumes that both motions are captured on a level floor, any variance in the floor level (i.e. a slope) would therefore adversely effect the similarity results produced using this approach.

3.1.7 Correlation Base Metrics

Although not as established as distance based metrics, similarity metrics have been proposed based on the correlation between corresponding joint parameters, rather than the difference in their values. PCC (Pearson’s Correlation Coefficient) has been proposed as a similarity metric to measure the alignment of two motions ([Etemad and Arya, 2015](#)).

Unlike distance based metrics that evaluate the difference between features defined by sets of parameters, such as position and rotation, correlation based metrics evaluate the correlation between the individual parameters themselves. This needs to be considered when choosing parameters to evaluate. For example the ambiguous way in which Euler angles can be used to specify a rotation, coupled with the potential for Euler values to be rolled over at ± 180 , means that they are not suitable for use with a correlation based metric unless they are pre-treated to correct these issues beforehand. Another example is quaternions, which would need to be hemispherised to ensure that the quaternions for a given joint, across both motions, are encoded in the same hemisphere.

[Etemad and Arya \(2015\)](#) proposed parameterising joint rotations as displacement vectors for use with PCC correlation. While displacement vectors do get around the issues associated with Euler angles, they are not a linear function of the rotation

and therefore may not be the optimal choice, or could be better evaluated using a rank based correlation method such as Spearman's (Zar, 2014).

An example of correlation based metric using PCC can be seen in Equation 3.13, where each joint is represented using a set of k parameters and c_ρ is the PCC correlation between two series of values.

$$c_\rho = \frac{\sum_{j=1}^m \sum_{a=1}^k w_j \rho(A_j^a, B_j^a)}{k} \quad (3.13)$$

The motivation for using correlation based similarity metrics is that they compare the overall form of the motion curves defined by corresponding parameters, rather than the values of the parameters themselves. This means that a correlation based metric, would not evaluate two motion curves that are the same shape but with a constant offset as different, whereas a distance based metric would. This is explored in more detail in Section 4.2.2.

Correlation algorithms such as PCC score the correlation of each parameter within the range $-1 \leq \rho \leq 1$, removing the need to normalise the values for each joint as is required for distance based metrics. While this is an advantage it also means that every parameter is equally weighted, regardless of how much variation there is in the parameter values (i.e. movement in joint). If the joints are equally weighted, this results in joints with little movement having the same impact on the similarity metric as joints with lots of movement. It also means that parameters associated with redundant axes are also considered equally important.

Correlation can also be used to evaluate the relationship between different data types. For example it could be used to evaluate the correlation between joint parameters and a sound source for analysing dance movements.

3.1.8 Feature Based Metrics

Feature based metrics measure similarity based on the correspondence of features within the motions, such as the Boolean features in Section 2.5.2. Equation 3.14 can

be used to measuring the similarity of the Boolean features of two motions, where F^Υ is an f by s matrix of f frames and s Boolean features within motion Υ (Röder, 2006).

$$c_{rel} = \frac{\sum_{f=1}^n \sum_{r=1}^s |F_{(f,r)}^A - F_{(f,r)}^B|}{ns} \quad (3.14)$$

3.2 Motion Recognition

3.2.1 Motion Recognition Using a Similarity Metric

A common application of similarity metrics is motion recognition, typically using a time warping algorithm to match the duration of two motions and temporally align their features before applying the metric Osawa, Ishikawa and Watanabe (2020); Chantaprasert, Chumchuen and Wangsiripitak (2019); Wahyuni et al. (2021).

When classifying or reorganising a motion it can be useful determine an average motion to act as a reference for a given movement. Petitjean, Ketterlin and Gançarski (2011) uses Barycenter averaging to determine an average motion.

The followings sections explore other methods of recognising motions, or finding motions similar to a query motion, that do not utilise a similarity metric to score the overall similarity of a motion.

3.2.2 Motion Recognition Using Clustering Algorithms

Each parameter of a motion can treated as a feature within a feature space of size k . For example 20 joints each represented in three dimensions would give 60 parameters or a feature space of $k = 60$. Each frame of motion can be represented as a single unit length k dimensional vector (Taranta II et al., 2017) allowing approaches such as: Support Vector Machines (SVM) (Vantigodi and Babu, 2013; Janaki, Babu and Sreekanth, 2013) to be used to cluster or categorise motions or poses unsupervised; or k-Nearest Neighbour (KNN) (Shin, Hasan and Maniruzzaman, 2022; Zhao et al., 2020) to identify motions and poses which are most similar. As with similarity

metrics, a time warping algorithm is often used to temporally align the features of a motion before SVM or KNN is used. [Taranta II et al. \(2017\)](#) uses the inner product of the feature vectors that make up a motion to find the most similar motion using KNN, which is referred to as first nearest neighbour (1NN).

3.2.3 Motion Recognition Using Statistical Probability

Hidden Markov Models can also be used for motion recognition, to determine the probability of a sequence of motion poses belonging to given motion ([Oz and Leu, 2011](#)). HMMs can be used to recognise a motion from a partial query motion, making them potentially useful for real-time tasks.

Logistic Regression is an adaptation of linear regression, in which probability of given value or motion type occurring is estimated, for a given parameter or set of parameters (i.e joint orientations or velocities). [Kratz, Rohs and Essl \(2013\)](#) compared the performance DTW, LR and Protractor 3D (discussed in Section 2.5.1, when recognising a predefined set of movement gestures captured using accelerometers, and found that that both DTW and Protractor 3D performed better than LR.

Like LR, Naive Bayes can also be used to determine the probability of a given set of joint parameters or features occurring in a particular movement. As Naive Bayes considers every feature to be independent, they are less accurate when used with features that contain repeating values or have some correlation between them, a characteristic common in motion data. Despite this, Naive Bayes has been found to be useful for broad classification problems, when applied with CNN, such as the detection of abnormal motions occurring in a home ([Ali et al., 2022](#)).

Statistical probabilistic approaches such as HMM, LR and Naive Bayes all require training data, making them only suitable for reorganising a defined set of movements, for which a number of examples have already been captured, such as sign language or defined gestures.

3.2.4 Motion Recognition and Time Warping

All of the motion recognition approaches in this section are can implemented alongside time warping. Before applying a motion recognition algorithm, it is good practice to time warp motions, to match their length and align their temporal features, to optimise and improve the consistency of its performance.

3.3 Time Warping

3.3.1 Introduction to Time Warping

Time warping refers to the manipulation of the timing of a motion sequence. This can be as simple as uniformly scaling a motion to change its duration, to carefully changing the timing of frames within the motion to temporally align its features with another motion.

Time warping is an established and widely used procedure, with many techniques and approaches to re-using motion captured data dependent on reliable and accurate methods of temporally aligning an input motion to the features of a target motion. Accurate motion alignment allows sophisticated motion manipulation techniques to be used, such as extracting (Etemad and Arya, 2014a), blending (Rose, Cohen and Bodenheimer, 1998; Bruderlin and Williams, 1995) and translating (Hsu, Pulli and Popović, 2005; Xia et al., 2015) stylistic features between motions. Interactive applications can also utilise motion alignment, examples include: providing real-time feedback in movement training (Chan et al., 2010); recognising interactions (Yun et al., 2012); or allowing virtual characters to interact with a person in real-time (Randall, Williams and Athwal, 2017).

3.3.2 Uniform Time Warping (UTW)

Uniform Time Warping (UTW) (Fu et al., 2008) is the most straight forward and naive method of time warping, uniformly squashing or stretching an input motion to match either a specified duration or the duration of a target motion. Motion data can be uniformly warped to match a target duration using the following two steps:

1. Determine the total number of frames required to encode a motion of the target duration, n_t , using $n_t = t_t s$, where t_t is the target duration in seconds and s is the sample rate in Hz.
2. Re-sample the input motion using n_t samples, uniformly spread apart between the start and end of the input motion, using Equation 3.15, where t_i is the temporal position in seconds of sample i , s_i is the sample number (indexed from 0) and i_t is the duration of the input motion in seconds.

$$t_i = s_i \frac{i_t}{n_t - 1} \quad (3.15)$$

UTW is useful for efficiently length matching motions, however it is not able to align the temporal features of a motion.

3.3.3 Dynamic Time Warping (DTW)

Dynamic time warping (DTW) (Sakoe and Chiba, 1978; Bruderlin and Williams, 1995) is a widely accepted approach for temporally aligning motion data. Not only does DTW change the length of the input motion being warped, to match that of the target motion, each frame is individually warped as required, to temporally align the features of the input motion to those of the target motion.

DTW monotonically matches the frames of an input motion to a target motion, with the objective of minimising the differences between the two motions. The difference or cost of the time warp is determined using cost function based on the distance based similarity metrics presented in Section 3.1.6, but without summing the costs of every frame in the motion.

Figure 3.1 shows an example of input motion aligned to a target motion, using both UTW and DTW. While UTW does scale the length of the input motion to match the target motion, DTW aligns the features within walking motion resulting in a more accurate alignment, in which all the peaks and troughs of the motion curve align. While UTW successfully aligns frames at the beginning and end of the motion, the

poses between frames 200 and 400 show DTW producing a better alignment than UTW.

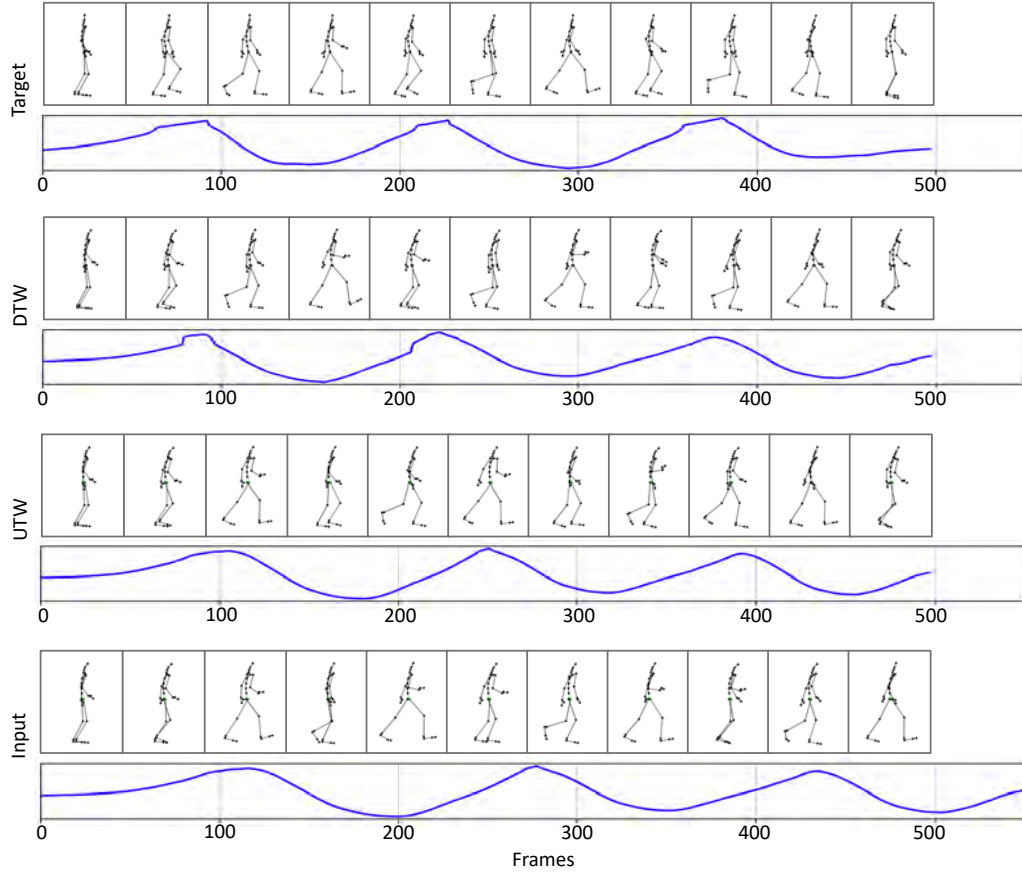


Figure 3.1: Comparison of UTW and DTW time warping, applied to a walking motion. The motion curves of the x axis of the left hip and poses show the accuracy of DTW and UTW alignment techniques at aligning the input motion to the target motion.

The DTW Algorithm

This section describes the DTW alignment process for aligning an input motion, I , of length m frames to a target motion, T of length n frames. A DTW alignment is performed using the following three steps:

1. Cost Matrix

Create a cost matrix, C , of the difference between each frame of the input motion, I , and the target motion, T . This is done using the Algorithm 1. The $dist(I_i, T_j)$ function can be based on any of the distance functions presented in Section 3.1. A commonly used cost function is the geodesic rotational

distance between the corresponding joints. The rotational distance similarity metric in Equation 3.1 can be adapted for use in a distance function as shown in Equation 3.16.

$$d_{\theta} = \sum_{j=1}^m w_j \frac{2}{\pi} \arccos |Q_j^{I_i} \cdot Q_j^{T_k}| \quad (3.16)$$

2. Accumulated Cost Matrix

Calculate an accumulated cost matrix, T , by accumulating the cost within matrix C , from $C_{0,0}$ to $C_{m,n}$, as shown in Algorithm 2.

3. Plot Alignment Path

Plot an alignment path through the accumulated cost matrix, T , backwards from $C_{m,n}$ to $C_{0,0}$. This is done using a dynamic programming algorithm that steps through the accumulated cost matrix using Algorithm 3, creating an alignment map P .

Algorithm 1: getCostMatix() Function

input : Sets I and T of size m and n respectively

output: Matrix C of size $m \times n$

```

1 for  $i \leftarrow 0$  to  $m$  do
2   for  $k \leftarrow 0$  to  $n$  do
3      $C_{i,k} = dist(I_i, T_k);$ 

```

Figure 3.2, shows an example of alignment path plotted by DTW, giving the optimal alignment between the features of an input and target walking motion. The accumulated cost matrix, through which the alignment was plotted, is visualised as a heat map. The DTW algorithm plots a path through the least cost areas if the accumulated cost matrix, which are darker than the high cost areas.

The alignment plotted by the DTW algorithm can be used to align the motions in either direction (i.e. align the target motion to the input motion, or visa versa). Additionally using DTW to align the target motion to input motion, will result in the same mapping between input and target frames, as using it to align the input

Algorithm 2: getAccumulatedCostMatix() Function

input : Matrix C of size $m \times n$

output: Matrix T of size $m \times n$

```
1 for  $i \leftarrow 0$  to  $m$  do
2   for  $k \leftarrow 0$  to  $n$  do
3      $T_{i,k} = infinity$ ;
4 for  $i \leftarrow 0$  to  $m$  do
5   for  $k \leftarrow 0$  to  $n$  do
6     if  $i = 0 \wedge k = 0$  then
7        $T_{0,0} = C_{0,0}$ ;
8     else
9        $T_{i,k} = C_{i,k} + \min\{T_{i-1,k}, T_{i,k-1}, T_{i-1,k-1}\}$ ;
```

to the target motion.

Constraints of the DTW Algorithm

Given a DTW alignment path, P , of length, k , mapping each input frame to a target frame, the alignment produced by the DTW algorithm meets the following conditions:

- **Monotonicity condition:** $P_1 \leq P_2 \leq \dots \leq P_k$. The alignment path cannot go backwards in time.
- **Boundary condition:** $P_1 = (0, 0)$ and $P_k = (n, m)$. The alignment path must start and end at the first and last frames of the aligned motions respectively.
- **Step size condition:** $P_i - 1 - P_i \in (1, 1), (1, 0), (0, 1)$. Limits the number of frames that can be jumped between points in the alignment path.

The DTW algorithm can be modified to adjust the boundary and step size conditions to either globally or locally constrain the alignment path respectively.

Algorithm 3: The DTWPlot() Function

input : Matrix T of size $m \times n$ **output:** Set P alignment map

```
1  $p \leftarrow$  next step as string;
2  $R = \{m, n\}$ ;
3  $P = R_0$  ;
4 while  $R_0 > 0 \vee R_1 > 0$  do
5    $p =$  "match";
6   if  $R_1 = 0$  then
7      $p =$  "delete";
8   else if  $R_0 = 0$  then
9      $p =$  "insert";
10  else
11    if  $\arg \min\{D_{R_0-1, R_1-1}, D_{R_0-1, R_1}, D_{R_0, R_1-1}\} = 1$  then  $p =$  "delete";
12    if  $\arg \min\{D_{R_0-1, R_1-1}, D_{R_0-1, R_1}, D_{R_0, R_1-1}\} = 2$  then  $p =$  "insert";
13    if  $p =$  "match" then
14       $R = \{R_0 - 1, R_1 - 1\}$ ;
15       $P = \{R_0\} \cup P$ ;
16    if  $p =$  "insert" then
17       $R_1 = R_1 - 1$ ;
18       $P = \{R_0\} \cup P$ ;
19    if  $p =$  "delete" then
20       $R_0 = R_0 - 1$ ;
```

Computational Requirements of DTW

The creation of the cost matrix within DTW, which requires the difference between every combination of frames in the input motion, I , and target motion, T , to be evaluated to determine their difference, means that the computational requirements of DTW are quadratic in relation to the lengths of the motions being aligned.

DTW Optimisation

A variety of approaches have been proposed to optimise the DTW algorithm, the main motivations for this is to either make it computationally more efficient or to ensure that the resulting aligned motions are physically plausible.

A common method of reducing the computation required to perform time warping is to constrain the search area within the cost matrix, reducing the number of points within the matrix that need to be calculated. The Sakoe-Chuba Band ([Sakoe and](#)

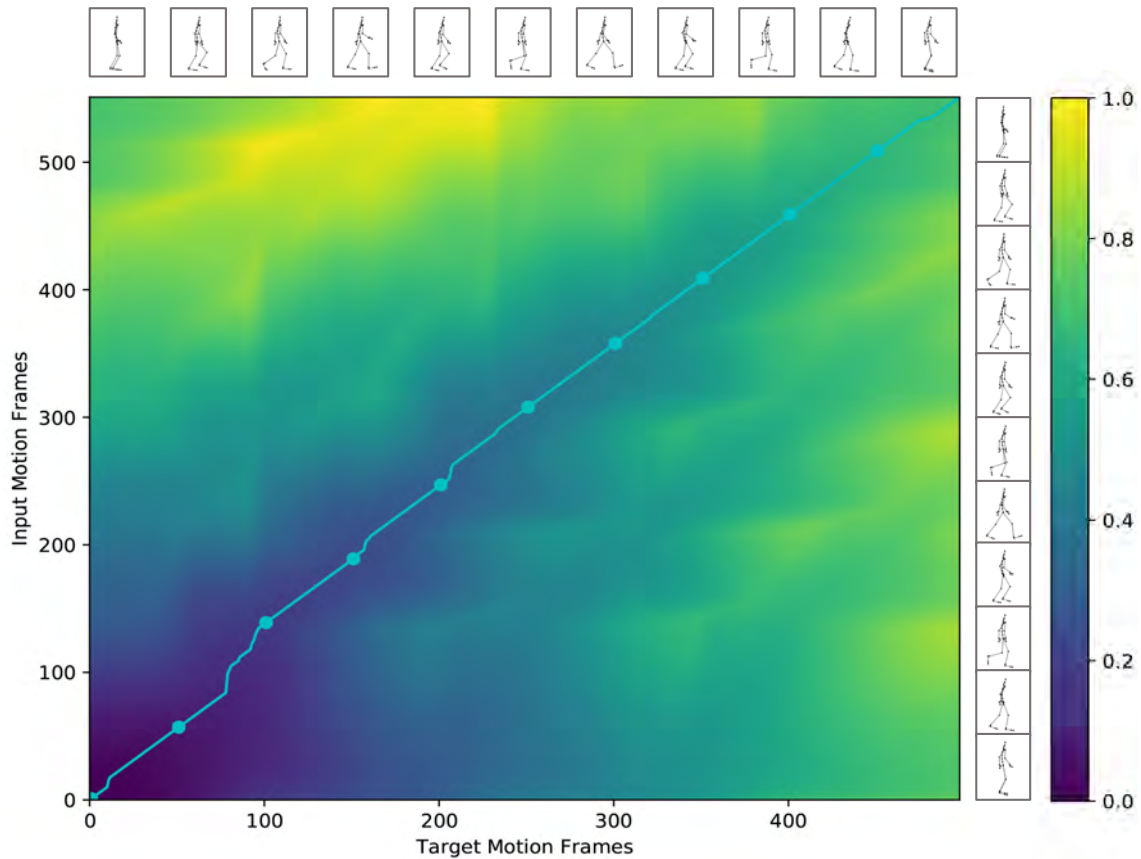


Figure 3.2: An example of a DTW alignment, overlaid over a normalised version of the accumulated cost matrix through which the alignment was plotted. The DTW was performed using rotational distance based on the hip, knee, shoulder and elbow joints, which were each equally weighed.

Chiba, 1978) and the Itakura Parallelogram (Itakura, 1975), shown in Figure 3.3, are commonly used to do this.

FastDTW (Salvador and Chan, 2004) addresses the efficiency problems of DTW by taking a multi-resolution approach, solving the alignment in multiple passes with increasing levels of granularity, using the previous pass to constrain the search area of the next pass. SparseDTW (Al-Naymat, Chawla and Taheri, 2012) uses a more directed approach to reducing the density of the cost matrix, determining the cost between frames of the two motions more sparsely where there is more similarity. Heloir et al. (2006) proposed using FastDTW to plot an initial alignment path, then constraining the search for a more accurate alignment, to an the area of the cost matrix around the initial alignment path.

As well reducing the number of cells in the cost matrix that need to be calculated,

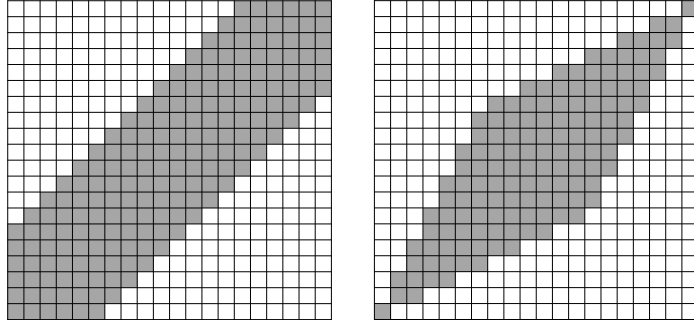


Figure 3.3: Two constrained search areas within a cost matrix. Sakoe-Chuba Band (left) and the Itakura Parallelogram (right).

by either constraining the search area or reducing its resolution, another approach to improving the efficiency of the DTW algorithm is improving the efficiency of plotting the alignment path. The approach shown in Algorithm 3 only searches adjacent cells for the next step in the alignment, this local search area for next step in the alignment path can be expanded allowing a path to skip frames and be plotted in fewer steps (Li et al., 2019).

In contrast to FastDTW and SparseDTW, Brankovic et al. (2020) proposed Continuous Dynamic Time Warping (CDTW), which increases the density of the cost matrix using integrals. This technique was proposed to cope with more sparsely sampled data, however, low sample rates are typically not an issue when working with motion capture data.

Improving the Accuracy of DTW

A number of approaches have been proposed to improve the accuracy of alignments resulting from DTW. A weakness of DTW is its dependence on a distance function to measure the difference between frames in the input and target motions. As discussed in Section 4.2.2, distance based functions do not cope well with motion curves that are offset or have peaks and troughs of different amplitudes, both variations that can naturally occur between two motions. This issue has also motivated the development of time warping algorithms based on correlation outlined in Section 3.3.4.

A number of adaptations to the DTW algorithm have been proposed to deal with this issue. For example Derivative Dynamic Time Warping (DDTW) (Keogh and

Pazzani, 2001) bases the cost function on the first order derivative, focusing the algorithm on measuring the difference in the overall shape or form of the motion curves. Vantigodi and Babu (2013) proposed synthesising a linear motion curve that started at a value of 1 on the first frame of the motion and ended at with a value of 2 at the end of the motion. For every joint axis evaluated within a similarity metric a copy of the synthesised motion curve was also added.

3.3.4 Correlation Time Warping

Conical Time Warping (CTW) (Zhou and Torre, 2009), combines Conical Correlation Analysis (CCA) and DTW to align multi-dimensional time series. The three rotational axis of each of the 20 joints that make up a skeletal hierarchy are used to represent each motion as a 60 dimensional feature vector. CCA is used to identify and select the features and frames within each motion that best correlate with one another, as well as translate them to optimise their fit. DTW is then used to temporally align the motions based on the features and frames selected using CCA.

Correlation-Optimized Time Warping (CoTW) (Etemad and Arya, 2015), uses correlation analysis to determine the optimal combination of temporal warps to apply to each segment of an input motion to align it to a target motion. An input motion is uniformly warped using UTW, then divided up into a given number of segments. Each of the frames at which one segment ends and another starts, set $\varepsilon \in I$, are allowed to be warped or moved within a given range, φ , called slack. The objective of CoTW is to find the combination of uniform warps, within the slack $\pm\varphi$, to apply to each segment that achieves the best alignment. Each possible combination of segment warps is evaluated using a correlation based similarity metric as described in Section 3.1.7.

3.3.5 Multidimensional Time Warping

Due to the multidimensional nature of motion data, the cost functions outlined in Section 3.1.6, must be either weighted for each joint or just averaged across a chosen subset of joints, depending on which joints are most pertinent to a given motion.

This section considers other approaches to dealing with the multi dimensional nature of motion data, within a time warping algorithm, in addition to the more general approaches presented in Section 2.5.2.

Multidimensional DTW (MD-DTW)(Walugembe et al., 2020) expresses the joints parameters or features of each frame in a motion as a feature matrix of n frames and s features. The distances between each frame within two motions, expressed as feature matrices A and B can be determined using Equation 3.17 where f and g are frames within motions A and B respectively. Weighting could also be incorporated as a $1 \times s$ matrix containing a weighting for each feature.

$$c(f, g) = \sqrt{(A_f - B_g)^T(A_f - B_g)} \quad (3.17)$$

Multiple Segmentation Norm - Weighted DTW (MSN-WDTW) (Zhao et al., 2020) segments the motions, then finds the internal correlations between joints which are specific to each segment. The motivation for this approach is to turn a collection of unrelated multivariate data, into univariate data based on joint relationships specific to each motion segment. When MSN-WDTW was compared to DTW, using datasets of accelerometer data, it was found that both approaches performed similarly with different approaches performing best on different data-sets.

3.3.6 Guided Time Warping

A time warp can be expressed as a sparse set of key frames, whether they are manually defined or extracted from another time warping algorithm. The Guided Time Warping algorithm (Hsu, da Silva and Popovic, 2007) implements warps using a sparse a set of purposefully positioned key frames, to minimise any distortion of velocities or acceleration in the motion, with the aim of maintaining the physical plausibility of the motion.

3.3.7 On-line Time Warping

Time warping algorithms have been adapted for real-time applications such as classification and recognition of human motions. Real-time time warping refers to accurately time warping a motion fast enough for real-time applications such as recognising a movement gesture. On-line time warping more specifically refers to the real-time alignment of a partially known sequence to complete reference sequence, which is required when time warping with streamed data sources such as data being streamed from a motion capture system.

The Challenges of On-line Time Warping

A challenge for real-time time warping is computation requirements of the DTW, which need to be constrained so that they do not quadratically increase in relation to the length of the motions being aligned.

[Dixon \(2005\)](#) also highlights the lack of explicit boundary conditions when warping on-line as a key challenge. As the length of one or more of the sequences is unknown, boundary conditions (i.e. the end frame of the input and target motion) and consequently the diagonal of the cost matrix are also unknown. This means that constraints and functions cannot be applied to the entire path. The consequences of this can be seen in figure 3.4, in which, unlike DTW, the on-line warp has not attempted to align the last frame of the input motion with the last frame of the target motion. This is because the algorithm does not have the information to do this.

Cost islands are particularly problematic with on-line time warping. Cost islands are areas of high cost cells within a cost matrix that are surrounded by low cost cells. While offline DTW's knowledge of the entire sequence allows it consider the entire cost island, in order to choose the optimal side on which to plot a path around the island. The dependence of on-line time warping on values within a local area of the cost matrix, means it can only see part of the island as it is plotting a path around it, potentially causing it to plot around the wrong side of the island.

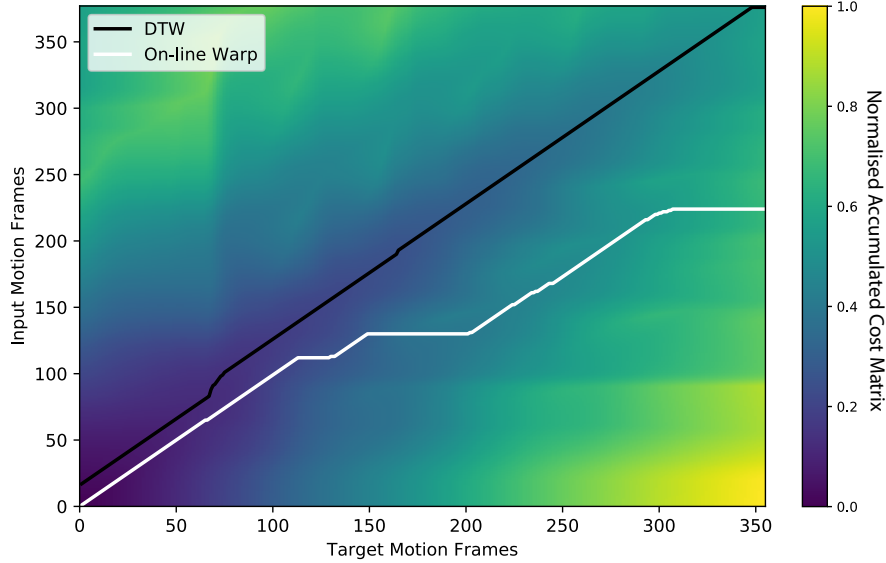


Figure 3.4: An example of a sub optimal alignment path plotted by an on-line warping algorithm.

Approaches to On-line Time Warping

An approach proposed by [Yabe and Tanaka \(1999\)](#) and Open Ended DTW or OE-DTW [Tormene et al. \(2009\)](#), both approach aligning a partially known sequence by selecting an optimal frame from the complete motion to align with the last frame of the partial motion, then plotting an alignment backwards using a standard or optimised version of DTW. This approach has been used to provide real-time feedback to patients undergoing post stroke motor rehabilitation [Tormene et al. \(2009\)](#), and recognising dance movements ([Kim et al., 2015](#)).

[Hülsmann et al. \(2017\)](#) proposed improvements to the accuracy of OE-DTW using regression techniques on training data-sets to optimise joint weightings, and implementing a path length weighting algorithm to prevent bias towards solutions which use shorter paths. The weighting algorithm reduced the tendency of motion recognition system based on time warping, to bias towards matching a query motion with the shorter motions in a motion library, which have shorter and therefore lower cost alignment paths.

An alternative approach to reducing computational complexity is to break the time series down into discrete chunks or windows. ODTW (On-line DTW) ([Oregi et al.,](#)

2017) splits the alignment into chunks, only preserving specific areas or windows of the cost matrix. It also introduces the concept of aligning a streamed time series, by weighting the distance or costs between more recent data points with more importance than older ones.

WTW (Windowed Time Warping) (Macrae and Dixon, 2010) breaks the time series down into discrete windows. Alignments are plotted within each window using DTW, which are referred to as sub-alignments. Each subsequent window starts somewhere on the alignment path of the previous sub-alignment and ends at a point the path is predicted to go through.

The Match music alignment tool (Dixon and Widmer, 2005), minimises the search area using a dynamic constraint, gradually adding more points to the search area within the cost matrix at each step of the alignment.

Rather than searching for an optimal point to start a time warp, more arbitrary approaches have also been taken. Examples include: aligning only last n frames of the on-line motion stream to prerecorded motion segments to identify the motion being performed (Emanuel and Widjaja, 2018; Patlolla, Mahotra and Kehtarnavaz, 2012); choosing a logical range of frames within a reference motion to align a partial motion against based on the duration of the partial motion (Jang et al., 2017); or segmenting the on-line motion stream into fixed lengths (Janaki, Babu and Sreekanth, 2013).

3.4 Summary

This chapter reviewed a number of approaches to measuring the similarity of two motions. It found that the majority of similarity metrics are based on the distance, or difference, between corresponding joint parameters, and are typically motivated by a need to measure the similarity of two motions or poses within frames. Metrics for measuring the temporal alignment of two motions have not been explored to the same extent, leaving a potential research gap. This gap motivated the study into correlation as a method for measuring the alignment of human motion, presented in Chapter 4.

The chapter also reviewed time warping algorithms, with a section on on-line algorithms that allows time warping of partially captured motions. On-line time warping methods typically plot alignment paths backwards from a selected frame of the complete motion and are design for use in motion recognition applications. Chapter 5 outlines the need for a forward plotting approach to support real-time monotonic alignment of human motions, for visualisation and feedback applications, before presenting and evaluating some novel on-line time warping approaches developed to satisfy this requirement.

Chapter 4

Measuring Alignment and Similarity of Human Motion

This work was published in the proceedings of the 2023 CASA conference on Computer Animation and Social Agents, then subsequently in the journal of Computer Animation and Virtual Worlds as: "Correlation as a measure for alignment and similarity of human motions" (Randall, Harvey and Williams, 2023a).

4.1 Introduction

To evaluate the performance of time warping algorithms, there is a need for a robust approach to measuring how well the temporal features of two motions have been aligned. This requirement forms the primary motivation for this study, however, as discussed in previous chapters, many techniques and approaches to effective re-use of motion captured data, are dependent on reliable and accurate methods measuring alignment. Examples include, aligning the temporal features of two motions to allow stylistic variations to be transferred between them (Xia et al., 2015) or to allow two motions to be blended together (Rose, Cohen and Bodenheimer, 1998; Bruderlin and Williams, 1995).

An established approach to measuring alignment two motions, is to measure their similarity using a similarity metric. As discussed in previous chapters, there are

a number of established techniques and approaches to measuring similarity. These techniques are primarily based on measuring the difference or distance between the joint parameters of two motions. Parameters such as position, orientation, velocity and acceleration are used and combined in different ways to form similarity metrics based on distance.

Although similarity metrics can be effective at measuring alignment, the similarity and alignment of two motions should not be considered the same. While similarity is concerned with whether the two motions contain the same movement, alignment is concerned with the temporal alignment of features within the motions. This study will show the importance of this subtle distinction when choosing an appropriate similarity metric for a given task.

[Yang and Guan \(2005\)](#); [Chan et al. \(2010\)](#) both review and compare a number of different similarity metrics based on angular and Euclidean distance. However, [Etemad and Arya \(2015\)](#) identified a number of shortcomings with using a distance based metric to measure alignment. They point to the fact that two motion curves with the same form, but offset, are incorrectly considered to be quite different by a distance based metric. They propose using a correlation based metric using Pearson's Correlation Coefficient (PCC).

PCC correlation has been utilised in a variety of ways to analyse ([Yang et al., 2010b](#)), time warp ([Zhou and Torre, 2009](#)) and manipulate motions ([Neff and Kim, 2009](#)). However, there has been very little study of rank based correlation methods such as Spearman's or Kendall Tau, which are potentially more optimal choices when working with non-parametric data such as motion curves.

Building on the knowledge of previous chapters, this study proposes a number of different correlation based similarity metrics, then evaluates their performance, comparing them with more established distance based similarity metrics. As well as comparing the effectiveness of PCC, Spearman's and Kendall Tau correlation coefficients, this study also explores the impact of different methods of parameterising joint angles on accuracy of these metrics.

A specially recorded data set of motions and statistical analysis approaches similar that used by [Chan et al. \(2010\)](#) and [Valcik, Sedmidubsky and Zezula \(2016\)](#) are used to evaluate and compare similarity metrics to determine which are optimal for measuring: (i) Alignment: how accurately the temporal features of a pair a motions are aligned, and (ii) Similarity: how similar a pair of motions are to each other.

4.2 Background

4.2.1 Comparing Time Warping Algorithms

There have been a number of comparisons of time warping techniques based on their ability to correctly identify similar motions ([Guerra-Filho and Bhatia, 2011](#); [Bankó and Abonyi, 2012](#)) rather than their ability to temporally aligning two motions.

The accuracy of temporal alignments can be visually evaluated, [Zhou and Torre \(2009\)](#) compared different methods of temporal alignment by projecting a series of rendered frames from the target motion and resulting aligned motions from the same angle for direct visual comparison. Although not applied to this study, this approach could be used with multiple participants scoring or comparing the resulting alignments. Although the inter-rater reliability of evaluating the temporal alignment of human motions has not been directly studied, medical and sports studies suggest that human poses ([Whatman, Hume and Hing, 2013](#)) and motions ([Herrington, Myer and Munro, 2013](#); [Ageberg et al., 2010](#)) can be reliably assessed in this way. Although this approach is useful for small studies, as the number of motion pairs grows quadratically in relation to the size of the data-set, mathematical approaches need to be used when working with larger motion databases.

Rather than use a distance based metric [Etemad and Arya \(2015\)](#) used Pearson Correlation Coefficient (PCC) to measure how well two motions align, using this method to compare a number of different approaches to motion alignment.

4.2.2 Correlation Vs Distance For Measuring Motion Similarity

A weakness of the distance based approach is the way in which it can be overly affected by any difference in the offset or amplitude of two signals, in comparison to correlation based approaches. Figure 4.1 shows a normalised signal of a joint parameter with an offset, multiplication, gamma and noise applied. Each of these transforms represent ways in which a repeated action or motion can vary between performance captures. A performer spatially missing a marked position would result in an offset, while a more exaggerated performance would result in some joint rotations being extended or amplified. Gamma represents the way in which a movement starts and stops, akin to the ease-in and ease-out of key-frame animation, this could be considered a pseudo representation of differences in a movements acceleration, which is also affected by the exaggeration of a performance. Just like any digital recording process, a motion captured performance will also contain a certain level of noise, this noise will vary between each recording.

Despite the original signal (blue) and transformed signal (red) being similar or identical in form, the distance based metric considers them to be dissimilar, where as the Pearson, Spearman and Kendall Tau correlation based approaches consider the signals to be very similar. While the distance based approach measures the similarity of each data point, correlation approaches consider the similarity between the overall forms of the two signals.

In Figure 4.1, the distance based metric considers the straight linear regression line to be more similar to the original signal than the offset or amplified versions of the signal, despite the line being a completely different shape. This is in contrast to the correlation methods which all gave the offset and multiplied versions of the signal a score of one, which is the same score two identical signals would achieve.

As well as PCC, which measures the linear relationship between two sets of variables, there are the Spearman's and Kendall Tau correlation coefficients, which are based on rank. These rank correlation coefficients are presented in more detail below, but have not been explored or utilised to same extent as PCC in relation to working

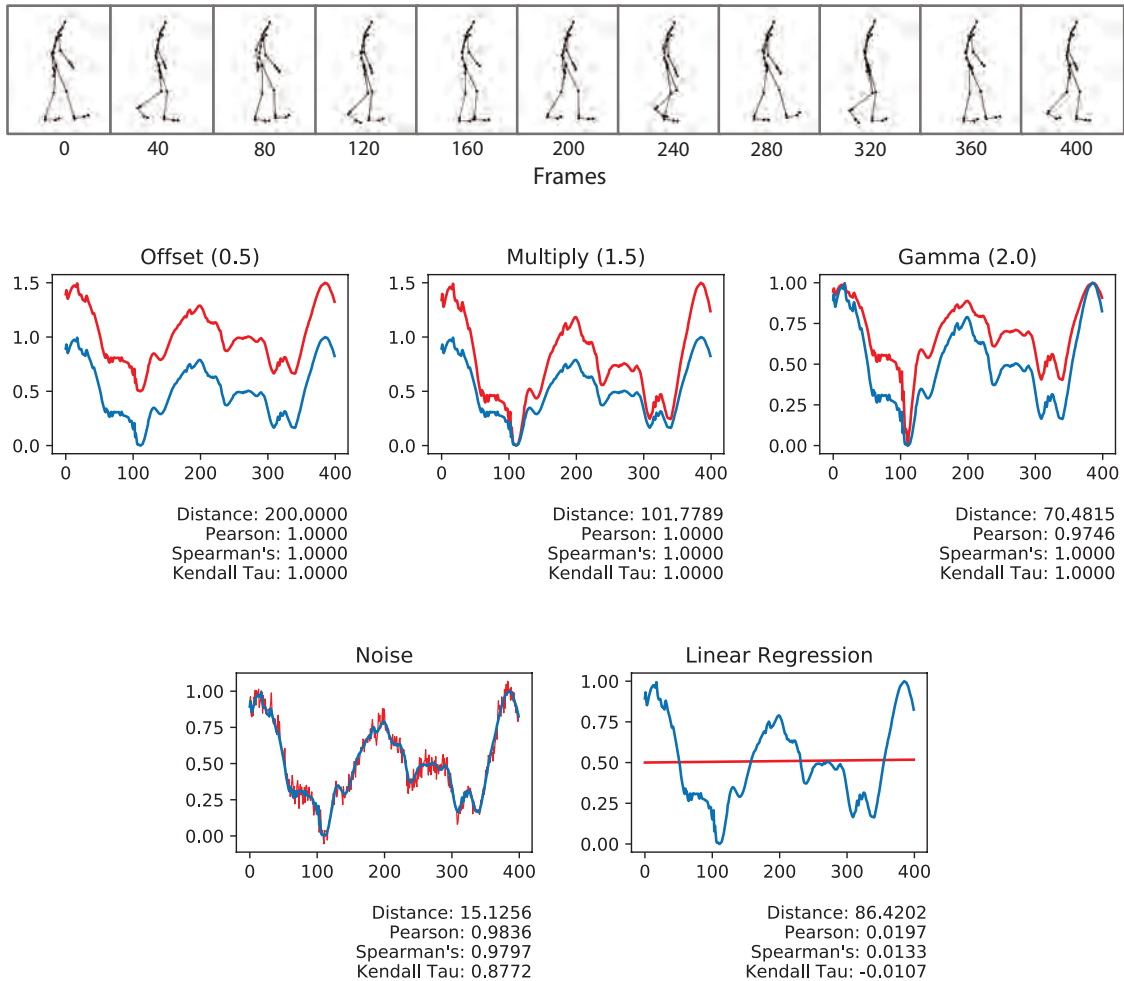


Figure 4.1: A comparison of the effects of different signal transforms on distance and correlation based measurements of similarity. The original signal of a right elbow in a walking motion, is shown in blue, while the transformed signal, with a similar form, is shown in red. The poses of the original motion can be seen at the top of the figure. The measurements under each graph show the inaccuracies that can occur when using distance as a measure of similarity.

with human motion data.

Unlike PCC, Spearman's and Kendall Tau rank correlation coefficients, are not impacted by non-linear transforms such as gamma. The impact of noise is also compared, with the Kendall Tau rank correlation coefficient being particularly sensitive to noise in comparison to other correlation approaches.

Unlike distance based metrics, correlation coefficients provide normalised results, where $r \in \mathbb{R} : -1.0 \leq r \leq 1.0$, regardless of how large the values within the sets are. This is useful when aggregating the results of multiple correlation coefficients.

For example when comparing two groups of multiple data series, as is often the case when working with motion data. This avoids the need to normalise data across an entire data-set, a data processing step sometimes required when working with distance based metrics. When applied to motion data, unless weightings are applied, the normalised results of correlation gives each joint an equal weighting, regardless of the range of motion occurring in each joint. Therefore there is potential for results to be overly influenced by joints with limited movement.

This comparison demonstrates the potential of correlation as a method for measuring similarity and alignment. It also shows the value of exploring rank correlation coefficients as an alternative to the more established PCC approach.

An addition to PCC this study will also consider the use of the rank correlation methods Spearman’s and Kendall Tau, which are potentially more suited to working with non-parametric data. Although rank based correlation methods have not previously been used to measure the similarity or alignment of two motions, they have proven effective in other related areas such as recognising human motions from movement sensors (Li et al., 2015) and measuring the alignment of protein sequences (Paluszewski and Karplus, 2009).

4.2.3 Linear and Rank Correlation Methods

PCC measures the linear relationship between two sets of variables, by dividing the co-variance of the two signals by the their variance. For two sets of values S_1 and S_2 of identical size m , the PCC (r_p) can be determined using Equation 4.1.

$$r_p(S_1, S_2) = \frac{\sum_{t=1}^m (S_1^{(t)} - \bar{S}_1)(S_2^{(t)} - \bar{S}_2)}{\sqrt{\sum_{t=1}^m (S_1^{(t)} - \bar{S}_1)^2 \sum_{t=1}^m (S_2^{(t)} - \bar{S}_2)^2}} \quad (4.1)$$

The nature of the offset and multiplication transforms maintains the linear relationship between the original and transformed signals, so does not impact PCC correlation. As gamma is a nonlinear transform, the linear relationship between the original and transformed signal is affected when gamma is applied, reducing the

correlation calculated using PCC, as shown in Figure 4.1,.

While PCC uses a data point's value, Spearman's and Kendall Tau rank correlation coefficients use a data point's rank, measuring the statistical dependence between the rankings of corresponding data points in two signals. Spearman's correlation coefficient performs a PCC on rank values using Equation 4.2 where $S_1^{(t)}$ and $S_2^{(t)}$ are rank values of data point t in sets S_1 and S_2 .

$$r_s(S_1, S_2) = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \quad (4.2)$$

where $d = r(S_1^{(t)}) - r(S_2^{(t)})$

The Kendall Tau rank correlation coefficient considers concordant and discordant pairs of ranked values. Where the sort order of the ranked values in S_1 and S_2 agree, they are considered concordant, otherwise they are considered discordant. The number of concordant pairs p_c and discordant pairs p_d are used with a binomial coefficient to determine correlation using Equation 4.3.

$$r_{k\tau}(S_1, S_2) = \frac{p_c - p_d}{\binom{n}{2}} \quad (4.3)$$

where $\binom{n}{2} = \frac{n(n-1)}{2}$

Unlike PCC, neither of the rank based correlation coefficients are effected by a gamma transform, as shown in Figure 4.1. Although a gamma transform results in a non-linear relationship between the original and transformed signal values, the rankings of these values are unaffected by the transform and are therefore identical for both signals. These identical rankings will have a perfect linear relationship, hence $r_s = 1.0$, and consist entirely of concordant pairs, hence $r_{k\tau} = 1.0$.

The Kendall Tau correlation coefficient is particularly sensitive to changes in direction (i.e the derivative of a signal changing from positive to negative or visa versa), occurring in one signal and not the other, as this produces discordant pairs in the ranked values. This characteristic is useful for measuring the alignment of two signals, as any misalignment between the signals will result in discordant pairs occurring around the peaks and troughs of the signals. It also makes this method particularly sensitive, arguably oversensitive, to the micro peaks and troughs that occur in noisy or jittery signals, as shown in Figure 4.1.

4.2.4 Correlation as a Motion Analysis and Manipulation Tool

The weaknesses of using distance as a measure of similarity, has motivated researchers to propose alignment techniques based on correlation, as alternatives to established distance based time warping methods such as DTW (Zhou and Torre, 2009; Etemad and Arya, 2015; Zhang et al., 2009).

Correlation and in particular PCC is used in a variety of applications related to motion capture and working with motion capture data. PCC has been used to evaluate the accuracy of motion capture systems by comparing their measurements to a known ground truth (Vander Linden, Carlson and Hubbard, 1992) or to the measurements of a more capable motion capture system (Pfister et al., 2014). It has also been used within a similarity metric to retrieve a best fit motion from a database that matches a given input motion (Etemad and Arya, 2014b).

Multivariate correlation is often used to analyse correlations of joints within a motion. For example correlations between a motion’s joints and parametric spaces (Neff and Kim, 2009) or latent subspaces (Zhu, Wang and Xia, 2006) have been utilised as well as correlations between the joints themselves (Yang et al., 2010b).

Within motion editing, correlation is being used to create tools that allow poses to be edited in a more intuitive parametric space (Neff and Kim, 2009), and analyse the relationships between joints and a dynamic model of the motion to create more natural interpolations between poses (Zhu, Wang and Xia, 2006). However, correlating

joint poses during editing does make it harder to accurately meet specific motion constraints, for example making sure a character reaches a particular contact point.

Correlations between joints have been used to evaluate the complexity of human motions using PCC (Yang et al., 2010b), they also underpin the use of analysis techniques such as Conical Correlation Analysis (CCA) to temporally align motions (Zhou and Torre, 2009) and PCA for dimensional reduction (Johnson, 2003) and compression of motion capture data (Arikan, 2006). Correlations between temporal as well as spatial aspects of a motion can be exploited, for example in the recognition of human motions (Shimada and Uehara, 2000).

4.2.5 Evaluating Similarity Metrics

The performance of a given similarity metric can be evaluated by measuring its ability to distinguish between aligned and non-aligned motions or between similar and dissimilar motions.

Chan et al. Chan et al. (2010) used a statistical approach to determine which of the three distance measures: joint position, velocity and angle are best for measuring the similarity of dance motions, and found that joint position performed best. Two groups of motion pairs were created, one consisting of similar motions and one consisting of dissimilar motions. Distance based metrics based on all three distance measures were applied to both groups of motion pairs and the resulting similarity scores compared. The distance measure which resulted in the least overlap between the scores of the two motion groups, was considered to discriminate the best and to be the optimal approach.

Valik et al. Valcik, Sedmidubsky and Zezula (2016) present a variety of measures for evaluating the impact of different feature based representations of human motion, on the retrieval of similar motions from a motion database. This included information retrieval measures, which measured the ability of different motion representations to correctly retrieve similar motions from a data-set.

4.3 Methodology

4.3.1 Overview

The aim of this study is to compare and evaluate the performance of a variety of distance and correlation based similarity metrics. A robust and objective approach is required to assess the capability of each metric to measure: (i) the overall similarity of two motion sequences, and (ii) the alignment of temporal features within two motion sequences.

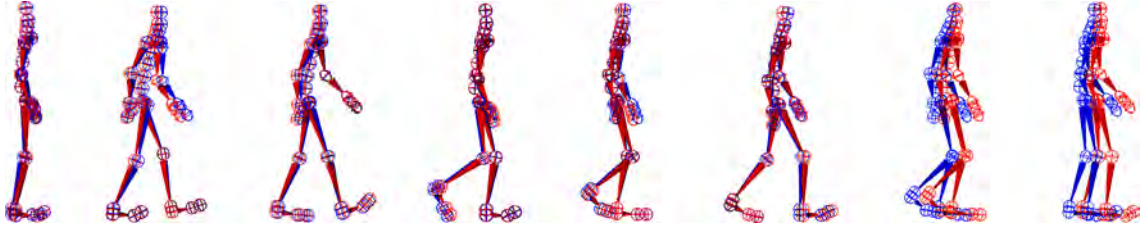
There are three key elements to the study’s design. First, a number of data-sets consisting of pairs of motion sequences were assembled. Four data-sets were created, comprising of aligned, non-aligned, similar and dissimilar pairs of motions. Second, a number similarity metrics were implemented, using consistent approaches to process motion data and deal with aspects such as: redundant axes; joint normalisation; and joints weightings. Third, the performance of each similarity metric was measured using the four data sets as follows: (i) the capability of a metric to measure alignment was assessed based on it’s ability to distinguish between pairs motions in the aligned and non-aligned data sets; and (ii) the capability of a metric to measure similarity was assessed based on it’s ability to distinguish between pairs motions in the similar and dissimilar data sets.

All the algorithms and tests used in this study were implemented using Python and the Python FBX SDK ([Autodesk, 2020](#)).

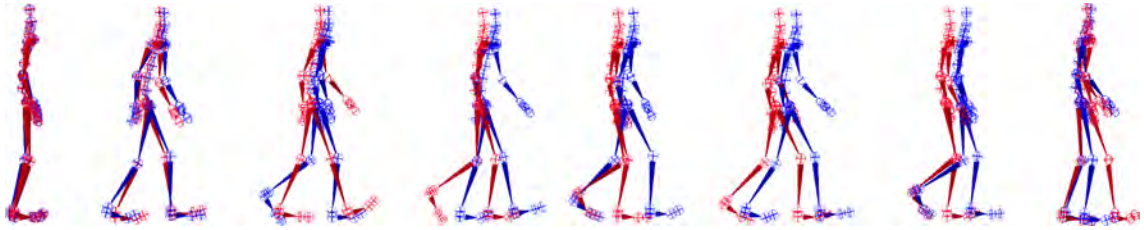
4.3.2 Sourcing the Data-Sets

To facilitate this study a number of data-sets, consisting of captured human motions, were required. Each data-set consisted of pairs of motions, referred to as motion pairs, each containing two motions that were either aligned, similar or dissimilar as described in Figure 4.2. In total four data-sets of motion pairs were required as follows:

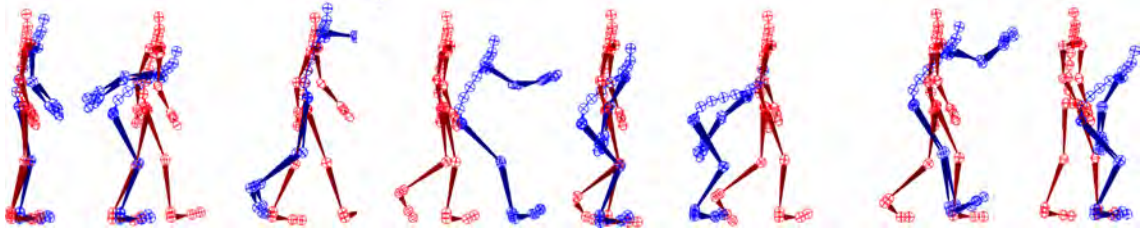
- To evaluate the performance of similarity metrics at measuring the alignment of two motions, two data-sets, one consisting of aligned motion pairs and the



(a) Aligned motion pair: two recordings of the same motion with one motion aligned to the other. This motion pair consists of blue walking motion that has been aligned to a red walking motion.



(b) Similar motion pair: two recordings of the same motion that have not been aligned. This motion pair consists of two walking motions.



(c) Dissimilar motion pair: two recordings of different motions. This motion pair consists of jumping motion (blue) and walking motion (red).

Figure 4.2: Examples of the three different types of motion pairs used in this study.

other similar motion pairs, was required. These data-sets are referred to as *Aligned* and *Non-Aligned* respectively.

- To evaluate the performance of similarity metrics at measuring the similarity of two motions, two data-sets, one consisting of both aligned and similar motion pairs and the other dissimilar motion pairs, was required. These data-sets are referred to as *Similar* and *Dissimilar* respectively.

Human Motion Capture

To create the data-sets a set of 63 motions were captured, consisting of 21 different movements each captured three times. The movements consist of common actions such walking, jumping, sitting and picking up objects, similar to Müller et al. (2007).

Each movement was carefully choreographed to achieve a high level of similarity between each of the three recordings.

Care was taken to match the number of steps, making sure the motions started and ended on the same foot, and markers were set out on the floor for the actor to hit with each step. If the actor failed to match any of these elements the capture was discarded and another capture was taken. The motions were all captured during a single capture session using the same actor. To ensure the results of this investigation were more widely applicable, a naive untrained actor was used.

The motions were captured using a Vicon motion capture system consisting of eight 2.2 megapixel Vero 2.2 cameras recording at 120 frames per second. The system was configured in 7x7 metre volume with a level floor and the markers applied to the actor in Vicon’s standard ‘FrontBackWaist’ 53 marker configuration, as shown in Figure 4.3. The Shogun Live software (Vicon, 2017) was used to capture the markers and fit its standard skeleton to the marker positions in real-time. The capture was performed on a high specification workstation with a Xeon processor and motions were checked to make sure that no frames were dropped during capture.

Shogun Post was used to convert the motion sequences to the .fbx format, with no manipulation or cleanup being performed.

4.3.3 Preparing and Assembling Data-sets

Before presenting how the data-sets were assembled, some terms and concepts need to be defined. Each set of three recordings of the same movement is referred to as a *motion-set* consisting of motions $\{a, b, c\}$. Each motion is considered to be similar to other motions within the same motion set and dissimilar to motions within any other motion set.

Aligned data-set

The motion pairs within the *Aligned* data-set consist of two similar motions from the same motion-set, with one motion time warped so that its length and temporal features match that of the other motion. For every permutation of two motions

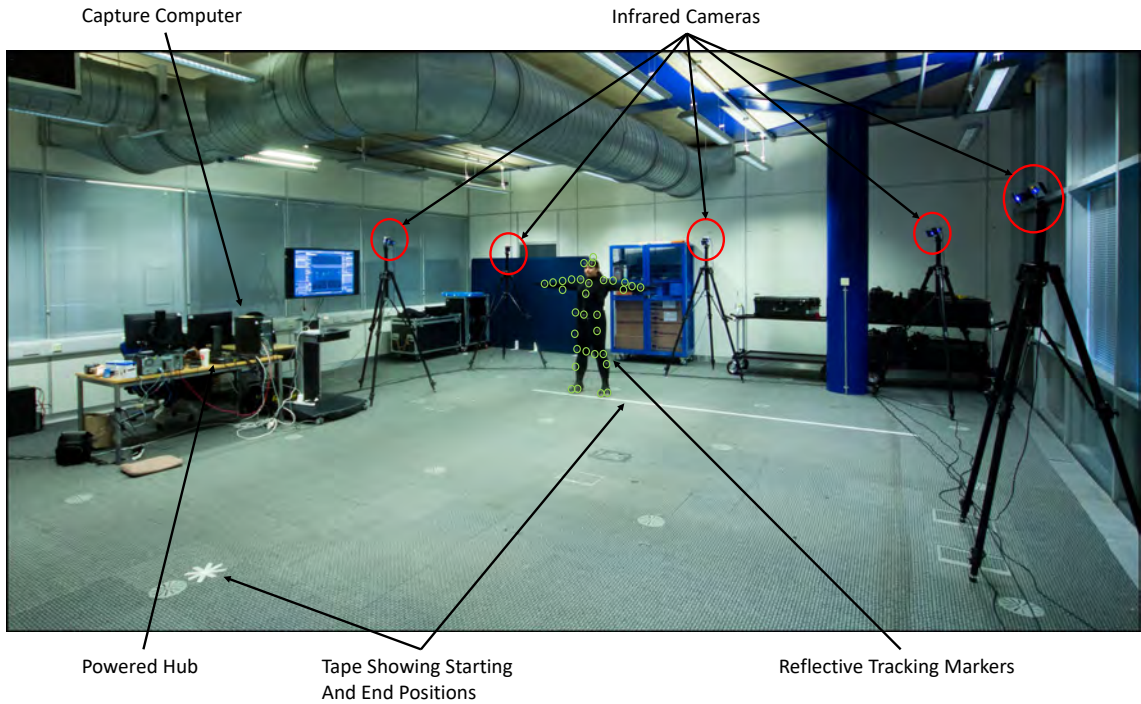


Figure 4.3: The Vicon motion capture system, used in this study. Only five cameras of the eight cameras are visible in the picture, they have been highlighted in red. The tracking markers which can be seen from this perspective have been highlighted in green. An approximate start and end point for motions involving a straight movement across capture volume, such as walking and jumping, have been marked out on the floor.

within a motion-set, a time warp was performed to align an input motion to a target motion, as shown in Table 4.1, with the target motion and the resulting aligned motion being combined to create a motion pair. For example input motion a and target motion b , would result in an aligned motion pair consisting of b and $a \rightarrow b$.

Table 4.1: To create a data-set of aligned motion pairs, time-warping was performed on the following permutations of input and target motions within each motion-set (a, b, c) , with the resulting aligned motions.

		Target Motion		
		a	b	c
Input Motion	a		$a \rightarrow b$	$a \rightarrow c$
	b	$b \rightarrow a$		$b \rightarrow c$
	c	$c \rightarrow a$	$c \rightarrow b$	

Time warping was performed using a standard DTW approach to warp an input motion, to optimise the correspondence of it's temporal features to that of a target

motion. The DTW time warping method is discussed in some detail within previous chapters and described by [Bruderlin and Williams \(1995\)](#). In this investigation an angular distance function based on quaternions and geodesic rotational distance was used ([Guerra-Filho and Bhatia, 2011](#)), which is also described in more detail in a previous capture.

The accuracy of the time warping functions used in this study were tested by warping an input motion a to a target motion b , then warping the resulting motion $a \rightarrow b$ back, aligning it back to the original input motion a . The resulting warped motion $a \rightarrow b \rightarrow a$ was then compared against the original motion a using a distance based similarity metric using joint rotations, to check for any deviation. In our tests the only deviations which occurred were extremely small and caused by frames being deleted during the warping process, as this information is not recovered when a motion is warped back.

Non-Aligned data-set

The *Non-aligned* data-set consists of similar motion pairs, made up of two recordings of the same motion which have not been aligned to each other. To produce a more comprehensive data-set, it is comprised of both unaltered captured motions and aligned motions produced for the *Aligned* data-set. The composition of these pairs can be seen in [Table 4.2](#), note that an aligned motion is never paired with the input or target motion use to create it.

Similar data-set

The motion pairs inside the *Similar* data-set consist of a mixture aligned and similar motion pairs. A motion pair was created from every combination of unaltered or aligned motion within a motion-set $\{a, b, c, a \rightarrow b, a \rightarrow c, b \rightarrow a, b \rightarrow c, c \rightarrow a, c \rightarrow b\}$. This approach generates a larger data-set, creating 36 similar motion pairs for each initial motion-set of 3 captured motions.

Table 4.2: The composition of motions pairs within the Non-aligned data-set. For each motion set (a, b, c) , nine motion pairs are created from the combinations identified below. Note that aligned motions such as $a \rightarrow b$ are also incorporated into the data-set.

		Motion A		
		a	b	c
Motion B	a		✓	
	b			✓
	c	✓		
	$a \rightarrow b$			✓
	$a \rightarrow c$		✓	
	$b \rightarrow a$			✓
	$b \rightarrow c$	✓		
	$c \rightarrow a$		✓	
	$c \rightarrow b$	✓		

Table 4.3: The number of motion pairs contained in each data-set used in this study.

Data-set	No. Motion Pairs
Aligned	126
Non-aligned	189
Similar	756
Dissimilar	840

Dissimilar data-set

Within the *Dissimilar* data-set, the motion pairs consist of two different motions from different motions-sets. To avoid creating an overly large data-set, which might bias results, not every possible combination different motions was used. Instead only motions a and b from each motion set were combined with their respective motions (a and b) in every other motion set.

Data-set sizes

Table 4.3 shows the size of each data-set used in this study.

A key motivation of using the approaches specified above, to assemble the data-sets, was to generate the most comprehensive data-set possible from the motions

Table 4.4: Equations for calculating the size of the data-sets that would generated from j motion-sets each consisting of k captures. Note that nP_r and nC_r represent the number of possible permutations and combinations of r elements in a set of size n .

Data-set	Equation For Determining Data-Set Size
Aligned	$j({}^kP_2)$
Non-aligned	$j({}^kC_2 + ({}^kP_2(k - 2)))$
Similar	$j({}^{k+{}^kP_2}C_2)$
Dissimilar	$j(j - 1)k^2$

captured. Table 4.4 provides equations for determining the size of data-sets that can be generated from using this approach on a set of motion captures consisting of j different movements, each recorded k times.

4.3.4 Measuring Motion Similarity

Metrics Included in Study

There is a wide range of different similarity metrics, many of which have been discussed in some detail in a Section 2.3.2. The choice of similarity metrics implemented in this study was motivated by two objectives: (i) compare the capability of distance and correlation based metrics to measure both similarity and alignment, and (ii) find an optimal approach to using correlation in a similarity metric.

Four distance based similarity metrics were chosen, consisting of established approaches to measuring similarity. Each metric is based around a different motion feature, as follows: joint orientation, joint position, joint velocity and point clouds. The point clouds were generated using a window size of seven, encompassing three frames immediately before and after the frame being sampled.

Overall 15 correlation based similarity metrics were implemented, one for each combination of three correlation and five angular parametrisation methods, being explored in this study. The three correlation methods implemented consisted of PCC; Spearman's; and Kendall Tau, and are presented in more detail in section 4.2.3. The

five joint angle parametrisation methods implemented consist of both established approaches such as Euler; quaternion; and rotational matrix, as well as couple of more novel approaches: displacement vector and logarithmic map, which have potential to work well in a correlation based metric. These parameterisation methods are presented in more detail in Chapter 2 and summarised in table 4.5.

In total 19 similarity metrics were tested in this study, 15 correlation and 4 distance base metrics.

Table 4.5: Overview of the joint parameterisation methods used in this study.

Method	Form	Description
Euler	x, y, z	Rotations in the x, y and z axes
Quaternion	w, x, y, z	Consists of a scalar w , and 3 complex numbers
Rotation Matrix	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	A 3 x 3 matrix specifying the orientation of each axis of a local space
Displacement Vector	x, y, z	Normalised vector in direction of angle
Logarithmic Map	x, y, z	Logarithm of a quaternion

Consistent Parametrisation of Joint Rotations

Unlike distance based similarity metrics, correlations based metrics are sensitive to any inconsistencies in how joints are parameterised. For example a common approach when encoding Euler angles is to roll a rotation value θ around when $-180 > \theta > 180$, such that a θ of 270° becomes -90° . Therefore Euler angles need to be unrolled to support the use of correlation based metrics. Similarly any three dimensional rotation can be expressed in two ways using a quaternion as $\pm Q$. To make sure that all quaternions are expressed consistently in the same hemisphere all joint rotations are hemispherised, using the hemispherisation process that was presented in Section 2.4.2.

Conforming Motion Lengths

Both distance and correlation based similarity metrics, require the motions being compared to have the same number of frames, however, only the aligned motion pairs will contain the same numbers of frames, as a results of the time-warping used to align them. Every other type of motion pair needs one of the motions to be uniformly squashed or stretched to match the number of frames in the other motion using UTW, before they can be used with a similarity metric.

Dealing With Redundant Axes

The joint parametrisation methods implemented in this paper are intended to express three degrees of movement, however, as discussed previously, not all three degrees of movement are required of every joint in order to encode human motion. The Vicon system used in this study encodes knees with one degree of freedom and effectively only uses two degrees of freedom to encode elbow joints, with very limited use of the z axis.

When joint angles are translated from their original Euler representation, in which the data was captured, to other joint representations. The unused axis can propagate low level noise into some of the joint parameters as shown in Figures 4.4 and 4.5.

Figure 4.4 shows the motion curves for the left elbow joint which has one degree freedom. The unused joints appear as flat lines, but still have a data point for each frame of captured motion. These flat lines will not impact the performance of a similarity metric, as long as the motions being compared have joints with matching degrees of freedom, as they will consistently score 1 with correlation based metrics and 0 with distance based metrics.

Instead of a flat line, the x axis of the displacement vector is comprised of very low amplitude noise, in this case 1×10^{-16} . This noise is caused by rounding errors when translating joint orientations between different parametric representations. As correlation based metrics normalise the motion curves, this low level noise is amplified to have the same weight or importance as any other within the similarity metric. This negatively impacts the accuracy of any similarity metric using this

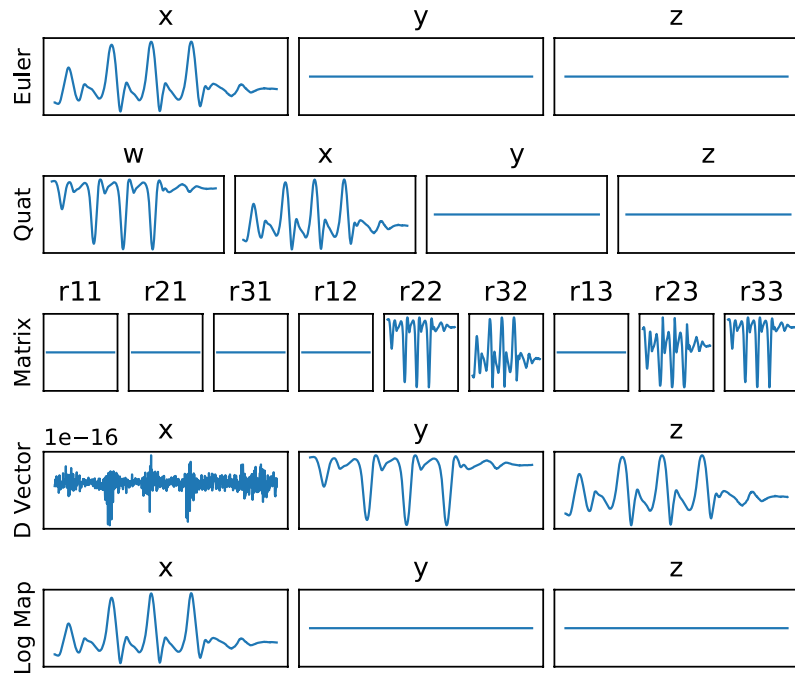


Figure 4.4: A comparison of how different methods of joint parametrization represent the motion of a joint with one degree of freedom. The motion curves show the rotation of the left knee joint changing over time during the same walking motion.

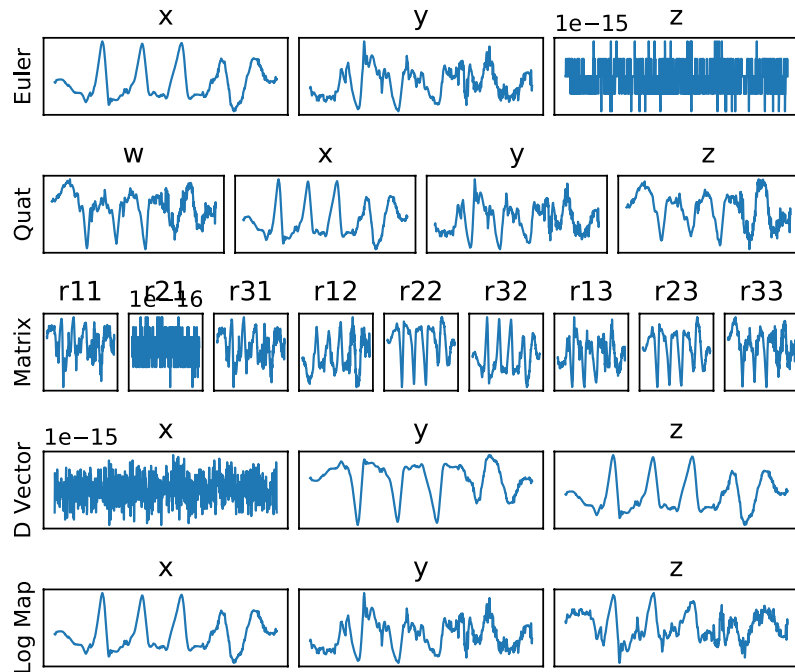


Figure 4.5: A comparison of how different methods of joint parametrization represent the motion of a joint with two degrees of freedom. The motion curves show the rotation of the left elbow joint changing over time during the same walking motion.

method.

While the motion curves of the left elbow joint shown in Figure 4.5, appear to show all 3 DOFs being used, the z axis of the Euler motion curves is comprised of low level noise, which will negatively affect the performance of a similarity metric as discussed above. The Vicon system used to capture the motion data sets, solves and records joint positions as Euler angles, therefore, the Euler motion curves represent the raw data recorded by the Vicon system. The noise in z axis is likely to come from a rounding error, occurring either when solving the pose of the joints to match position of the tracking markers or when translating rotations from some other form. This noise propagates into the matrix and displacement vector parameters, but not into the quaternion or log map parameters.

To resolve this issue a threshold was incorporated into each similarity metric, in which any joint comprised entirely of low level noise between $\pm x$, is treated in the same way as a joint with a constant value. In the case of this study $x = 1 \times 10^{-5}$.

Joints such as knees and elbows do not have to be constrained to using one or two DOF, this is a particular approach implemented by the Vicon motion capture system. Motion data from other data-sets such as the HDM05 (Müller et al., 2007) and CMU (CMU Graphics Lab, 2001), use all three DOF even when parametrising joints with a limited range of movement, thus avoiding the problems described here.

Normalisation of Joints for Distance Metrics

Before the similarity scores of the joints can be summed, to give a total score, they need to be normalised for each joint across the whole data-set, to ensure that each joint is equally weighted. This is not a concern for correlation based metrics (as joints scores are already normalised) or for point cloud distance metrics (which calculate a single translation for the entire motion), however, this step does need to be performed for the distance based metrics where each joint is scored independently (i.e. those based on angles, position or velocity of each joint).

Checking of Similarity Metrics

To check the similarity metrics were functioning correctly, they were tested by using them to compare every motion in the data-set to itself. As expected the correlation based metrics consistently provided a result of 1 and distance based metrics provided results of 0, indicating perfect matches.

4.3.5 Joint Weightings

When time-warping motions or measuring their similarity, rather than using all of a motion's joints, it is common to focus on joints which are particularly salient to general human motion, such as shoulders and hips, ignoring less significant joints such as hands and feet (Lee et al., 2002). What is considered a significant or less significant joint is somewhat affected by the metric being used. For example the joints which are significant to a metric based on joint rotations in local space will be different to those of one based on joint translations in a global space. These less significant joints tend to be peripheral and more invariant to the motion being performed, while having a limited range of movement. In the case of joint rotations defined in local space, less significant joints occur towards the end of the kinematic chain.

With the parametrisation of joint rotations in local space forming one of the focal points of this investigation, the time-warping and similarity measurements within study were performed using a subset of equally weighted joint as shown in Figure 4.6. The subset consisted of both the left and right: shoulders, elbows, hips and knees.

4.3.6 Evaluating Similarity Metrics

The performance of each similarity metric was evaluated using three different performance tests. The overlap and MAP tests measured the ability of each similarity metric to distinguish between aligned and non-aligned motions or between similar and dissimilar motions, while the correlation test measured how consistent the scores of a metric are with those of other metrics.

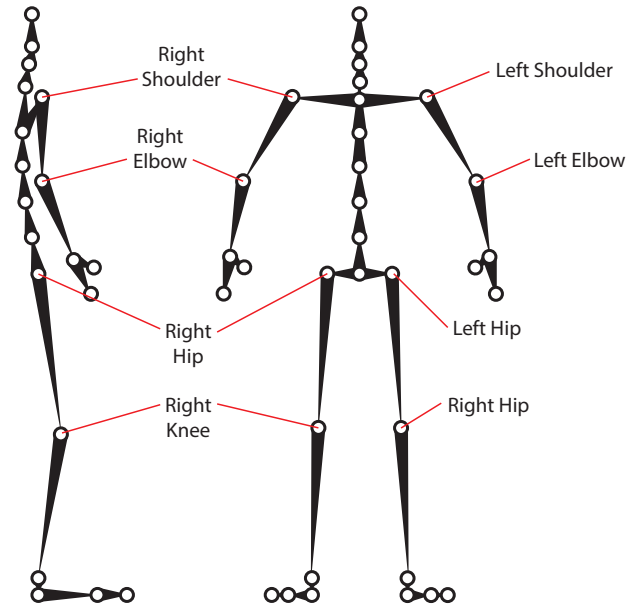


Figure 4.6: Configuration of joints used in study.

To evaluate a metric’s ability to distinguish between aligned and non-aligned motions, overlap and MAP test were used to compare the similarity scores of the aligned and non-aligned motion pair data-sets. In the same manner, these tests were also used to compare the scores of of the similar and dissimilar data-sets to evaluate how well a metric can distinguish between similar and dissimilar motions. A metric that is better at distinguishing between align and non-aligned motions is considered better at measuring alignment, while one that is better as distinguishing between similar and dissimilar motions is considered better as measuring similarity.

Overlap Test

A Mann–Whitney U test ([Haslwanter, 2016](#)) was used to measure how much the similarity scores of two data-sets overlapped each other, and is based on approach used by [Chan et al. \(2010\)](#). Each similarity metric was applied to both data-sets and the resulting scores compared using the test. The Mann–Whitney U was used as the similarity scores within the sample groups failed the Shapiro-Wilk, D’Agostino Skew and D’Agostino Kurtosis tests for normal distribution, therefore a non-parametric test was required.

The test measures the probability of a randomly selected score from the sample

group with the highest mean being less than a randomly selected score in the other sample group, using Equation 4.4. p is the probability of a random pair of motions from one data-set a with n_a motion pairs, scoring a lower similarity score than a random pair of motions from data-set b with n_b motion pairs, where U is the test statistic produced by the Mann-Whitney U test. If the distribution of two sample groups perfectly match, the probability of a score from one sample group being higher than the score of another sample group is 50%, hence p has a maximum possible value of 0.5, therefore $p \in \mathbb{R} : 0 \leq p \leq 0.5$.

$$p = \frac{U}{n_a n_b} \quad (4.4)$$

The similarity metrics whose scores have the lowest probability p of overlapping, differentiate better between the two data-sets and should be considered more optimal.

Mean Average Precision (MAP) Test

The Mean Average Precision (MAP) is commonly used to evaluate search algorithms, but has been used by [Valcik, Sedmidubsky and Zezula \(2016\)](#) to evaluate similarity models for human motion. Just like the overlap test, the MAP test measures how well a metric can distinguish between the motion pairs from two different data-sets, but using a contrasting approach. Rather than measuring overlap between two sets of scores, the MAP test combines the two sets of scores into single sorted list, then considers how many of the scores with the closest k values to each score are from the same data-set.

Given a query score q , precision P_k is the fraction of scores within the k nearest scores from the same data-set as q . This is determined using Equation 4.5, where $m \cup k$ is the number scores from the same data-set as q within the nearest k neighbouring scores.

$$P_k = \frac{m \cup k}{k} \quad (4.5)$$

To determine the average precision AP for a given query score q , the P_k for every value of k is considered in the range $\{1\dots n\}$ using Equation 4.6, where n is the total number of scores in the data-set that q belongs to, and rel_k is an indicator function which is equal to 1 if the k th nearest neighbouring score is from the same data-set as q , or 0 otherwise.

$$AP = \frac{\sum_{k=1}^n P_k rel_k}{n} \quad (4.6)$$

The mean average precision (MAP) is the average AP for every score within the combined list of scores. This is determined using Equation 4.7, where Q is the total number of scores in the combined lists.

$$MAP = \frac{\sum_{q=1}^Q AP_q}{Q} \quad (4.7)$$

The similarity metrics whose scores achieve the highest results in the MAP tests, show that they are able to differentiate better between the two data-sets and should be considered more optimal.

Correlation Test

The correlation test measures how well the scores of each metric correlate with average metric scores, indicating how consistent the scores for a given metric are with those of other metrics. To reduce the impact of any differences between the distributions of scores from different metrics, the scores from each metric are ranked, with averaging and correlation operations performed on the ranks values.

Before being ranked the scores from the distance based metrics were inverted by subtracting each score from 1, to bring them in line with the scores of correlation based metrics where 1 is a perfect match and 0 means totally different.

Once ranked, the metric scores attained by each motion pair are averaged, producing a set of average scores against which the correlation of each set of metric scores is measured.

4.4 Results

4.4.1 Interpreting Results

The results of the overlap, MAP and correlation performance tests for each similarity metric can be seen in Tables 4.6 and 4.7. Three different measurements (overlap, MAP and correlation), were used to measure the performance of 19 different similarity metrics when executing two different tasks, measuring the alignment and similarity of two motions. The results of each performance test are colour coded with a continuous linear colour grade from green (best) to red (worst). For the overlap tests the lower the result the smaller overlap indicating a better performance, for the MAP and correlation tests higher results indicate better performance.

The correlation method used in each correlation based metric is identified using the following symbols: ρ_p Pearson, ρ_s Spearman's, ρ_{kt} Kendall Tau. The results for performance tests on the correlation based metrics were averaged across each of the five joint parameterisation techniques and three correlation methods used. This allowed the performance impact of the different approaches to joint parameterisation and correlation to be evaluated independently of one another. Rows showing the mean scores are identified by μ .

The distributions of the similarity scores were tested for normal distributions using the Shapiro Wilk, D'Agostino Skew and D'Agostino Kurtosis tests, shown in Figure 4.7. Most of the results failed these distribution tests, supporting the use of the Mann-Witney U non-parametric test method for measuring overlap between results. For a set of results to be considered normally distributed they would have to pass all three normal distribution tests.

Figures 4.8 and 4.9 visually show the distribution of the scores obtained from the different similarity metrics for each type of motion pair and the overlap between aligned and non-aligned motion pairs (4.8a and 4.8b) and similar and dissimilar motion pairs (4.9a and 4.9b).

	Aligned			Non-aligned			Similar			Dissimilar		
Euler P_p	0.00	0.09	0.00	0.00	0.00	0.25	0.00	0.00	0.02	0.00	0.00	0.00
Euler P_s	0.00	0.09	0.02	0.00	0.00	0.48	0.00	0.00	0.02	0.00	0.00	0.00
Euler P_{kt}	0.00	0.14	0.01	0.00	0.00	0.08	0.00	0.00	0.56	0.00	0.00	0.00
Quaternion P_p	0.00	0.01	0.74	0.00	0.00	0.17	0.00	0.00	0.03	0.00	0.00	0.00
Quaternion P_s	0.01	0.20	0.06	0.00	0.00	0.42	0.00	0.00	0.02	0.00	0.00	0.00
Quaternion P_{kt}	0.01	0.44	0.02	0.00	0.00	0.05	0.00	0.00	0.61	0.00	0.00	0.00
Matrix P_p	0.00	0.02	0.28	0.00	0.00	0.21	0.00	0.00	0.03	0.00	0.00	0.00
Matrix P_s	0.00	0.22	0.01	0.00	0.00	0.42	0.00	0.00	0.03	0.00	0.00	0.00
Matrix P_{kt}	0.01	0.33	0.00	0.00	0.00	0.07	0.00	0.00	0.52	0.00	0.00	0.00
Displacement Vector P_p	0.00	0.00	0.30	0.00	0.00	0.51	0.00	0.00	0.00	0.00	0.33	0.00
Displacement Vector P_s	0.00	0.01	0.45	0.00	0.00	0.77	0.00	0.00	0.00	0.00	0.27	0.00
Displacement Vector P_{kt}	0.00	0.06	0.06	0.00	0.00	0.14	0.00	0.00	0.81	0.00	0.31	0.00
Logarithmic Map P_p	0.00	0.05	0.00	0.00	0.00	0.09	0.00	0.00	0.05	0.00	0.00	0.00
Logarithmic Map P_s	0.00	0.24	0.00	0.00	0.00	0.61	0.00	0.00	0.01	0.00	0.00	0.00
Logarithmic Map P_{kt}	0.00	0.35	0.00	0.00	0.00	0.12	0.00	0.00	0.98	0.00	0.00	0.00
Angular Distance	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.01	0.00
Positional Distance	0.14	0.66	0.58	0.00	0.00	0.54	0.00	0.00	0.00	0.00	0.00	0.00
Positional Velocity	0.00	0.27	0.25	0.00	0.48	0.77	0.00	0.44	0.70	0.00	0.00	0.00
Point Cloud	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02

Figure 4.7: Results of the normal distribution tests performed on the similarity scores yielded by the similarity metrics in this study for each data-set of motion pairs. Passed tests ($p \geq 0.05$) are green, failed test ($p < 0.05$) are red.

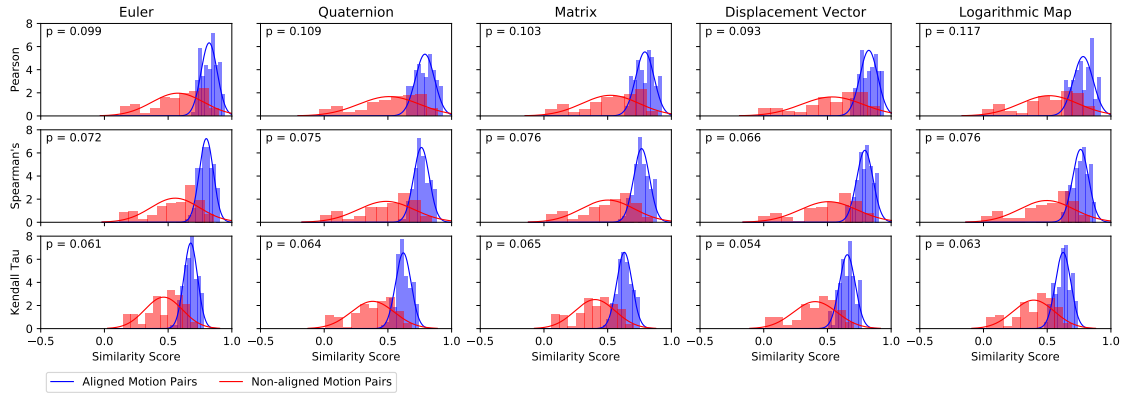
4.4.2 Measuring Motion Alignment

The ability of each similarity metric to accurately measure the temporal alignment of two motions was determined by applying them to two data-sets of paired motions, aligned and non-aligned. The resulting similarity scores were then compared using the three performance tests, to determine how well each metric distinguishes between the two data-sets. The metrics which performed best on these data-sets should be considered more optimal for applications which require motions to be identified with closely aligning features, such as finding motions to blend with existing motion, and for evaluating the performance of time warping algorithms.

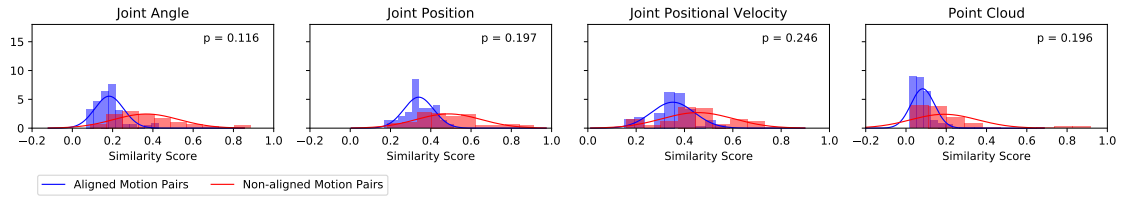
The results of the tests can be seen in Table 4.6 and Figure 4.8. Overall the correlation based metrics performed significantly better than distance based metrics at distinguishing between aligned and non-aligned pairs of motions. This suggests that correlation based metrics are a better choice when measuring alignment.

Table 4.6: The performance of each similarity metric as a measure of alignment. The ability of metrics to measure alignment was evaluated by comparing the similarity scores of motion pairs from the *Aligned* and *Non-Aligned* data sets.

	Overlap	MAP	Correlation Aligned	Correlation Non-Aligned
Euler ρ_p	0.111	0.481	0.920	0.976
Euler ρ_s	0.073	0.557	0.956	0.989
Euler ρ_{kt}	0.062	0.586	0.957	0.987
Euler μ	0.082	0.541	0.944	0.984
Quaternion ρ_p	0.115	0.478	0.896	0.974
Quaternion ρ_s	0.067	0.566	0.960	0.987
Quaternion ρ_{kt}	0.057	0.599	0.966	0.985
Quaternion μ	0.080	0.548	0.941	0.982
Matrix ρ_p	0.108	0.487	0.819	0.953
Matrix ρ_s	0.070	0.561	0.831	0.960
Matrix ρ_{kt}	0.059	0.592	0.865	0.968
Matrix μ	0.079	0.547	0.838	0.960
Displacement Vector ρ_p	0.090	0.511	0.839	0.945
Displacement Vector ρ_s	0.065	0.565	0.947	0.988
Displacement Vector ρ_{kt}	0.052	0.603	0.960	0.988
Displacement Vector μ	0.069	0.560	0.915	0.974
Logarithmic Map ρ_p	0.137	0.447	0.926	0.973
Logarithmic Map ρ_s	0.078	0.540	0.965	0.989
Logarithmic Map ρ_{kt}	0.063	0.581	0.972	0.988
Logarithmic Map μ	0.093	0.523	0.954	0.983
Peason (ρ_p) μ	0.112	0.481	0.880	0.964
Spearman's (ρ_s) μ	0.071	0.558	0.932	0.983
Kendall Tau (ρ_{kt}) μ	0.059	0.592	0.944	0.983
Angular Distance	0.116	0.514	0.656	0.862
Positional Distance	0.197	0.412	0.762	0.920
Positional Velocity	0.246	0.395	0.572	0.797
Point Cloud	0.196	0.399	0.566	0.655
σ (without outliers)	0.026	0.067	0.086	0.019



(a) The results of correlation based similarity metrics applied to Aligned and Non-aligned motion pairs. The greater the score the greater the similarity.



(b) The results of distance based similarity metrics applied to Aligned and Non-aligned motion pairs. The smaller the score the greater the similarity.

Figure 4.8: Histograms visualising the overlap between the similarity scores of motion pairs in the *Aligned* and *Non-Aligned* data-sets. The probability of an aligned motion pair scoring less than a non-aligned pair is shown by p . The lower the value of p for a given similarity metric the better it is a differentiating between aligned and non-aligned motion pairs.

The grouping and averaging of correlation based metrics, based on the correlation and joint parameterisation method used, identified the Kendall Tau method as the best performing correlation method in all three performance tests. Significantly both rank correlation methods consistently perform better than Pearson’s linear correlation in all tests. The overlap and MAP tests showed displacement vectors to be the best method of parameterising joints, with the approach also performing very well in the correlation tests. The optimal metric for measuring alignment is a correlation based metric in which joints are parameterised using displacement vectors and the correlation is measured using Kendall Tau.

The averages of correlation based metrics in the overlap and MAP tests show a smaller deviation between metrics that used different approaches to parameterising joint angles, than between those that used different methods of correlation. This suggests that the choice of correlation method is a more important factor to consider

than the choice of joint angle parameterisation.

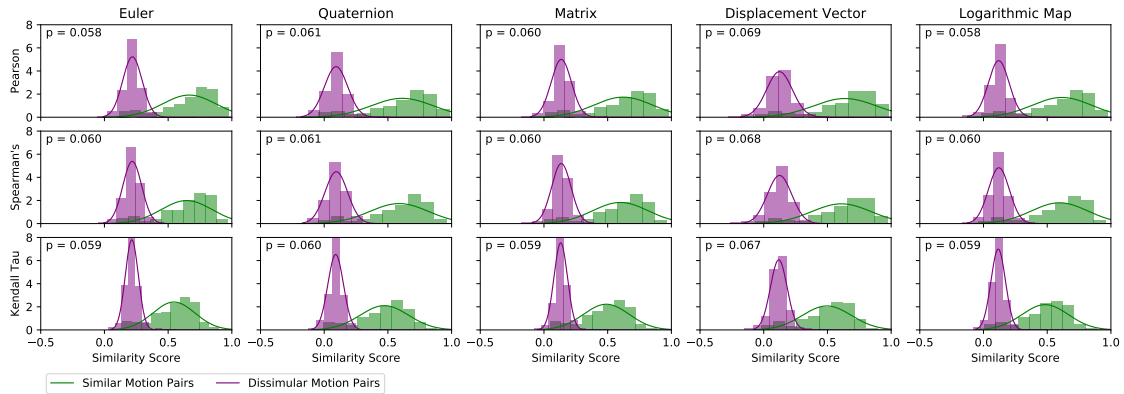
The angular distance metric also performed well in both the overlap and MAP tests at distinguishing between aligned and non-aligned motion pairs. This method would be a good choice where correlation based metrics are hard to implement, such as on-line applications in which there is incomplete knowledge of one or of the motions.

4.4.3 Measuring Motion Similarity

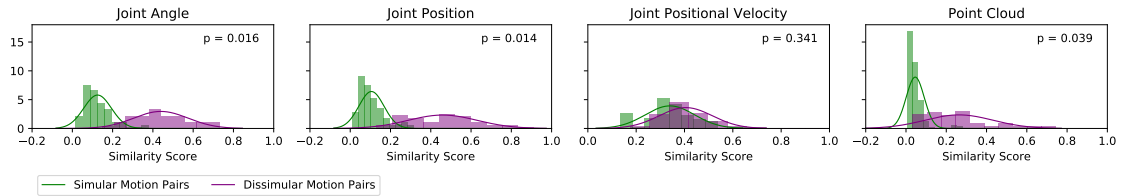
To determine the optimal metric for measuring the similarity of two motions, the metrics were applied to two data-sets of paired motions, one containing pairs of non-aligned motions with the same movement and the other containing pairs of motions with different movements. The resulting sets of similarity scores were then compared using the three performance tests, to determine how well each metric distinguishes between similar and dissimilar pairs of motions. This approach tests how well a metric can distinguish between different movements, identifying optimal metrics to use for tasks such as identifying or classifying motions.

The results of the tests can be seen in Table 4.7 and Figure 4.9a. In general the distance based metrics performed better than the correlation based metrics when measuring motion similarity. In particular, metrics based on angular and positional distance performed the best in both the overlap and MAP tests, with a clear performance gap between these and other metrics on the overlap test. The high performance of angular and positional distance based metrics in the MAP and overlap tests, shows that these metrics are ideal for performing discrete or Boolean decisions such as identifying a motion as the same or not the same. However, the lower and less consistent performance of distance based metrics in the correlation tests, suggests that they might be less suited to applications where the similarity of motions are being graded on a continuous scale. This implies that correlation based metrics would be a better choice for this type of application, however, these results are likely to be affected by potential bias in the correlation test method as discussed in section 4.4.4. Additionally, the limited differentiation between the performance of the correlation based metrics in this task, means that an optimal correlation based

metric cannot be clearly identified.



(a) The results of correlation based similarity metrics applied to Similar and Dissimilar motion pairs. The greater the score the greater the similarity.



(b) The results of distance based similarity metrics applied to Similar and Dissimilar motion pairs. The smaller the score the greater the similarity.

Figure 4.9: Histograms visualising the overlap between the similarity scores of motion pairs in the *Similar* and *Dissimilar* data-sets. The probability of a dissimilar motion pair scoring less than a similar motion pair, is shown by p . The lower the value of p for a given similarity metric the better it is a differentiating between the different types of motion pairs.

Given the marginal difference in performance between angular and positional based distance metrics, the optimal choice between these two metrics will be dependent on the use case. The angular distance metrics is a better choice for comparing the overall joint poses within the motions, as it is less affected by any potential differences in joint lengths between the two motions. However, if the priority is to measure similarities in the positions of end effectors, such as hands and feet, then a metric based on joint position should still be used.

The results of the performance tests from distance based metrics corroborate the findings of [Chan et al. \(2010\)](#) which also found that distance metrics based on joint angle and position performed best at discriminating between similar and dissimilar motions and that a distance metric based on joint velocity performed particularly poorly. For applications where joint velocity is important, such as working with

Table 4.7: The performance of each similarity metric as a measure of similarity. The ability of metrics to measure similarity was evaluated by comparing the similarity scores of motion pairs from the *Similar* and *Dissimilar* data-sets.

	Overlap	MAP	Correlation Similar	Correlation Dissimilar
Euler ρ_p	0.073	0.747	0.979	0.943
Euler ρ_s	0.073	0.750	0.990	0.949
Euler ρ_{kt}	0.072	0.743	0.989	0.949
Euler μ	0.073	0.747	0.986	0.947
Quaternion ρ_p	0.074	0.745	0.973	0.945
Quaternion ρ_s	0.072	0.748	0.989	0.960
Quaternion ρ_{kt}	0.071	0.740	0.988	0.959
Quaternion μ	0.072	0.744	0.983	0.955
Matrix ρ_p	0.074	0.755	0.954	0.870
Matrix ρ_s	0.072	0.759	0.964	0.904
Matrix ρ_{kt}	0.071	0.750	0.972	0.909
Matrix μ	0.072	0.755	0.963	0.894
Displacement Vector ρ_p	0.083	0.737	0.951	0.905
Displacement Vector ρ_s	0.081	0.741	0.988	0.931
Displacement Vector ρ_{kt}	0.080	0.732	0.988	0.928
Displacement Vector μ	0.081	0.737	0.976	0.921
Logarithmic Map ρ_p	0.073	0.748	0.977	0.942
Logarithmic Map ρ_s	0.072	0.754	0.991	0.966
Logarithmic Map ρ_{kt}	0.071	0.746	0.991	0.965
Logarithmic Map μ	0.072	0.749	0.986	0.958
Peason (ρ_p) μ	0.075	0.746	0.967	0.921
Spearman's (ρ_s) μ	0.074	0.750	0.984	0.942
Kendall Tau (ρ_{kt}) μ	0.073	0.742	0.986	0.942
Angular Distance	0.016	0.775	0.906	0.388
Positional Distance	0.014	0.769	0.916	0.224
Positional Velocity	0.341	0.288	0.790	-0.064
Point Cloud	0.039	0.670	0.757	0.285
σ (without outliers)	0.001	0.009	0.025	0.026

dynamic or ballistic motions, a point cloud metric should be considered, rather than a joint velocity metric. Point clouds metrics still account for velocity over a very small time window, while performing better than correlation based metrics.

The similarity scores attained by applying distance based metrics to the dissimilar data-set, were significantly more widely distributed, with a mean standard deviation of 0.146, than those attained by correlation based metrics, with a mean standard deviation of 0.0757, when applied to the same data-set. This potentially explains the better performance of distance based metrics when measuring similarity, although these metrics did achieve significantly lower scores in the correlation performance tests when compared against the correlation based metrics.

4.4.4 Review of Performance Tests

The results of the three performance tests frequently corroborated each other, particularly the overlap and MAP test which were similar in nature. The results of correlation tests appeared to be less consistent. To better understand which tests are more suited to measuring the performance of similarity metrics for which task (i.e. measuring alignment or similarity), the standard deviation σ of the results for each test was calculated, after outlier results outside of the interquartile range were removed, these can be seen at the bottom Tables 4.6 and 4.7.

Both the overlap and MAP tests were better able to differentiate between the performance of different correlation based metrics at measuring motion alignment than similarity. As the overlap and MAP test both clearly identified the same distance based metrics as being optimal for measuring motion similarity, this is probably more indicative of the suitability of the correlation based metrics for measuring similarity, than it is of the suitability for the overlap and MAP test for evaluating the performance of metrics for measuring motion similarity.

The results of correlation performance tests consistently showed a high or very high correlation between individual correlation based metric scores and the mean score. The standard deviation of the test results showed the correlation test to be better suited to measuring the performance of metrics at measuring alignment than

similarity.

The correlation test is particularly susceptible to potential bias in the results. As the averages against which the metrics scores are being correlated against in this study are based on 15 correlation based metrics and 4 distance based metrics, they will correlate better with the correlation based metrics than the distance based metrics. The effects of this bias is visible in the test results with a significant gap between the results of the correlation based metrics and the distance based metrics, particularly when evaluating a metrics' ability to measure similarity.

To some extent the results of correlation test are function of the distribution of the scores of each type of motion pair which can be seen in Figures 4.8 and 4.9. Where a metric's scores are more widely distributed the metric performs better in the correlation test than when they are more narrowly distributed.

Across all of the performance tests there is significantly more variation between the results of the distance based metrics than the correlation based metrics. This is to be expected as each distance based metric is based on a wider variety of motion features such as velocity, rotation or position, where as all the correlation based metrics are all based on joint rotation.

4.5 Discussion

4.5.1 Combining Similarity Metrics

Although the optimal metrics for measuring the similarity and alignment of two motions have been shown to be different, they can be used together. A robust approach to identifying a candidate motion to blend with another motion, could be to first use a distance metric based on joint position, to identify a short list of candidate motions. Then use a correlation based metric to identify which of the short listed motions has the best alignment. This approach of using a distance based metric to short list candidate motions, then a correlation based metric to identify the best candidate has been implemented in different forms by other researchers (Yang and Guan, 2005; Kim et al., 2009). This two stage approach could potentially allow

less accurate lightweight approaches to be used to short list similar motions when working with large data-sets. A number of lightweight similarity metrics based on comparing Boolean features have already been proposed for working specifically with motion data (Müller, Röder and Clausen, 2005) and more generally with time series data (Zhang et al., 2009).

4.5.2 Impact of Motion Characteristics

To explore the potential impact of a motion’s characteristics upon the accuracy of similarity metrics, deviations between scores obtained across the different correlation metrics for the motion pairs within the same motion set were analysed, using the aligned data-set. A motion set that obtains similarity scores with a wider deviation, could contain a movement with characteristics that may adversely effect the accuracy of a similarity metric. By focusing on the aligned data-set, factors such as the motions being dissimilar or out of alignment will have been largely removed, allowing other potential characteristics to be evaluated. Before calculating their standard deviation, the scores for a each motion pair were normalised, by dividing them by their mean, to neutralised any bias from a motion set’s overall similarity.

There was a weak correlation (0.32) between the length of the motions in the motion pair and the deviation between scores, suggesting that the length of a motion doesn’t significantly impact the accuracy of these similarity metrics. There was a very strong negative correlation (-0.84) between the average similarity scores obtained by a motion pair and the deviation between those scores, this indicates that the similarity of motions being compared does impact the accuracy of correlation based similarity metrics, with pairs of motions that are less similar having more deviation between their similarity scores.

Figures 4.10 and 4.11 show the motion curves and joint poses of two example motion pairs, one pair has a low deviation between their similarity scores, the other a high deviation. Figure 4.10 shows a pair a walking motions with a high mean similarity score (0.87) and a low standard deviation between those scores (0.086). Figure 4.11 shows a pair of motions containing identical movements performed while sitting in a

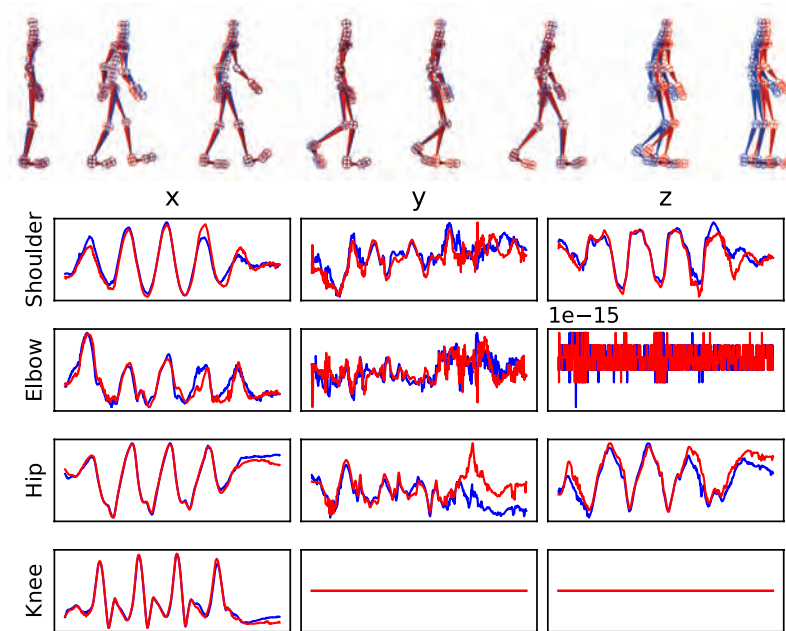


Figure 4.10: The motion curves of right hand joints and character poses, of a pair of walking motions, where the red motion has been aligned to the blue motion. Within the aligned data-set, correlation based similarity metrics scored this pair of motions the highest (μ score of 0.87), with a low deviation between the different metric scores ($\sigma = 0.086$).

high chair, they scored a lower mean similarity score (0.81) with a higher deviation between those scores (0.182).

The form of the motion curves belonging to the sitting motions do not match each other as well as the walking motions. Despite being temporally aligned there are still clear spatial differences between the two sitting motions, which are clearly visible in both the joint poses and the motion curves, particularly in the y axis and particularly in the joints of the lower half of the body (hip and knees).

The flat line at the start of the red motion curves in the sitting motion, is the result of the same frame being duplicated a large number of times during the motion alignment process. This indicates that the recording of two motions, start at different places relative the main substance of the motion.

The nature of the sitting motion causes very little movement in the lower body, and unlike a walking motion, there is limited correspondence between the movements of the upper and lower joints. Any approach to time warping or similarity measurement in which the upper and lower joints are equally weighted will be adversely effected

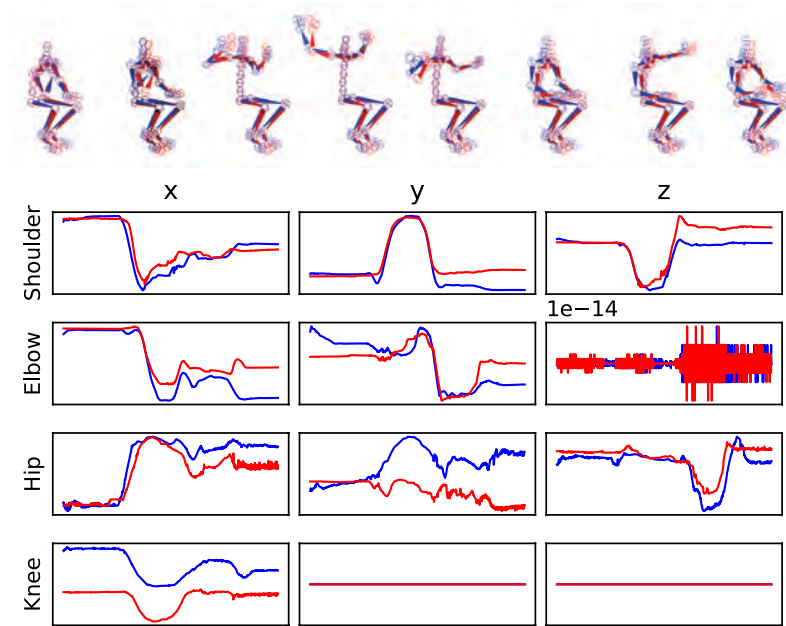


Figure 4.11: The motion curves of right hand joints and character poses, of a pair of motions with identical movements performed while sitting down, where the red motion has been aligned to the blue motion. Within the aligned data-set, correlation based similarity metrics scored this pair of motions the lowest (μ score of 0.81). with a higher deviation between the different metric scores ($\sigma = 0.182$).

by motions such as this, particularly with correlation based metrics that normalise the results for every joint regardless of the extent of their movement.

There are also spatial differences in the gestures made by the upper body of the sitting motion. The shoulder is positioned slightly further back in the blue motion and the hands are in different positions at the extreme reaches of the gestures motion.

Unlike the sitting movements, walking is a motion we constantly use and are likely to be able to repeat with greater accuracy than the upper body gestures in the sitting motion. However, Table 4.8 demonstrates the dangers of making generalisations across motion features, as unexpected factors can have a significant impact on similarity of motions. For example stylised walks such as sneaky and stiff walks can have a significantly lower similarity score compared to more natural walks, while the movements performed on the low chair have a very high similarity because the legs were firmly planted on the floor and did not move between each recording of the motion.

Table 4.8: The mean similarity scores for each motion set. The ticks indicate the features contained within each motion. The mean similarity is based on the scores attained from motion pairs in the aligned data-set.

	Walking	Sitting	Walking and Sitting	Dynamic	Rhythmic	Object Handling	Score μ Score
Walk	✓						0.848
Stepping stones walk	✓						0.846
Dance					✓		0.881
High Chair Sit Walk			✓				0.861
Walk Sit High Chair			✓				0.867
Gestures on High Chair		✓					0.831
Gestures on Low Chair		✓					0.879
Jaunty Style Walk	✓						0.830
3 Jumps				✓			0.841
Long jump				✓			0.831
One Hand Pickup						✓	0.882
Two Hand Pickup						✓	0.859
Punch Kick				✓			0.884
Stacking shelves						✓	0.825
Sit Walk			✓				0.884
Walk Sit			✓				0.838
Slow walk	✓						0.847
Sneaky Style Walk	✓						0.813
Stiff Style Walk	✓						0.824
Twirling				✓			0.822
μ Score	0.835	0.855	0.863	0.845	0.881	0.855	

The range of movement in a motion can also have a significant impact on motion similarity. Motions with a greater range of motion (i.e. have a greater variation in joint angles) have the potential to score more highly in similarity metrics, than motions with limited movement. Many of the movements that have a high similarity in Table 4.8, contain the most movement, examples include: punching and kicking; dancing and transitioning between walking and sitting. Note that a motion such as walking may appear to have a lot of movement, however, while walking creates considerable translation in global space, it is the amount joint rotation in local space that is relevant here. This phenomena can be expressed as the ratio between the spatial deviation between two motions and the joint movement inherent in the motions being performed. This ratio is likely to be higher for motions that require joints to rotate more and lower for those that do not.

4.6 Conclusions

This study compared a variety of similarity metrics based on both distance and correlation. Different methods of representing angular data and measuring correlation were evaluated, including novel approaches such as the use of displacement vectors, logarithmic maps and rank correlation. The tests revealed that in general correlation based metrics are better for measuring the alignment, while distance based metrics are better for measuring similarity. The results also showed that the alignment and similarity of two motions should not be considered the same, with different similarity metrics performing best in each use case. For example applications such as motion blending and motion graphing, are more concerned about motion alignment and should therefore use correlation based metrics, while applications which recognise or classify motions are concerned about motion similarity, and therefore, should use distance based metrics.

The results showed a correlation metric based in joints parameterised using displacement vectors and correlation measured using Kendall Tau rank correlation, to be the best method for measuring the alignment of two motions. They also showed that distance based metrics based on angular or positional distance should be used to measure the similarity of two motions. Depending on the application, angular distance can be used to compare the overall pose of a motion, while positional distance can be used to compare the position of end effectors.

The optimal approaches to measuring alignment identified in this study, will be particularly useful in identifying candidate motions in motion synthesis and accurately measuring the performance of time warping algorithms. However, it is important to keep in mind that no matter how accurately a similarity metric measures alignment, it will not consider factors such how much a motion is distorted or the physical plausibility of the output motion, other metrics could be used to do this ([Etemad and Arya, 2015](#); [Reitsma and Pollard, 2003](#)).

This study showed the important of using a variety of different methods when evaluating the performance of similarity metrics. Each test in this study evaluated subtly contrasting aspects of a metric’s performance. While the overlap and MAP

test are objective tests, care should be taken when implementing the correlation test and interpreting its results, as the test will bias itself towards the approaches that are most predominant in a given study and data-sets that generate more widely distributed scores. A robust approach that allows a direct comparison of different similarity metrics is important, as choosing an optimal method measuring similarity is crucial decision for researchers working with large data-sets of motions in variety of disciplines and contexts including: motion synthesis; identifying and classifying motions; and training AI algorithms.

A novel approach was used, combining sets of ($n > 2$) similar motions to efficiently create larger numbers of motion pairs of different types. This approach could be applied to other data-sets such as the HDM05 data-set ([Müller et al., 2007](#)).

Other than choosing a relevant subset of joints, joint weightings were not explored in detail within this investigation. It is expected that joints weighting would have a limited impact on the fundamental findings of this study, with more optimal joint weighting configurations, focused on the movement being performed, expected to re-enforce these findings.

The results of this study provide a clear optimal approach to measuring alignment within future chapters, in which studies into on-line time warping algorithms are presented.

Despite the novel approach used to build the data-sets used in this study, there is a clear need to increase the size of the data-set for future studies. A power analysis can be performed on the similarity scores attained in this study to determine a minimum sample size to be used.

Chapter 5

Online Warping of Human Motion Using Windowing

This work was published in the proceedings of the 2023 CASA conference on Computer Animation and Social Agents, then subsequently in the journal of Computer Animation and Virtual Worlds as: "Online alignment of human motion using forward plotting-dynamic time warping" (Randall, Harvey and Williams, 2023b).

5.1 Introduction

Chapter 3 introduced the concept of using time warping approaches, such as DTW, to align the temporal features of one motion to another, as well as presenting on-line approaches to time warping. As discussed there are significant shortcomings and barriers to using techniques such as DTW in on-line applications. However, there a variety of potential applications for an on-line algorithm capable of aligning a prerecorded human motion to an incomplete human motion as it is being captured, such as real-time visual effects production or a virtual dance trainer (Chan et al., 2010). Accurate alignment is a key element of applications such as motion style transfer (Xia et al., 2015), and on-line alignment algorithms allow these functions to be performed in real-time.

Building on the knowledge developed in previous chapters, this chapter will propose

and test a number of novel approaches to on-line time warping of human motion, which plot alignments in a forward direction. This study will utilise techniques established in the previous chapter, for measuring the alignment two human motions.

A distinction should be made between real-time and on-line time warping. Real-time and on-line time warping both require fast and efficient methods for warping time series data to align temporal features, and therefore share many characteristics and use similar approaches. The key distinction is that real-time warping can refer to warping with two completely known time series in real-time, while on-line time warping refers to warping in real-time while one or both of the time series are incomplete and only partially known. Typically on-line time warping is use in data capture scenarios, such as speech recognition in which temporal features of word sounds are being aligned to live audio capture to recognise words as they are spoken.

This study is focused on aligning a complete prerecorded motion, referred to as the input motion, with an incomplete or partially captured motion, referred to as the target motion. Within this paper the terms *input* and *target* correspond to the alignment process, where input motion is being aligned or warped to fit the target motion. Frames of the complete input motion must be mapped monotonically to every frame of the incomplete target motion in real-time as it is being captured. The mapping needs to be performed in such a way as to result in an optimal alignment of the input motion.

As discussed in Section 3.3.7, previously proposed approaches to on-line time warping typically plot an alignment backwards, aligning the best fitting part of a complete time series with the known part of a partial time series. However, this approach requires the mapping of the last known frame of a live target motion to be unconstrained, allowing it to be mapped to any frame of the input time series that is determined to be most optimal. This means that continuity can not easily be enforced between the mapping of sequential frames in a live target motion. Figure 5.1 shows an example in which inconsistent starting points have been determined from the backwards alignment of two sequential frames (15 and 16) of a target motion. This will cause the playback of the input motion to snap backwards, between the

start points of each alignment on frames 15 and 16 of the target motion, resulting in a jump in the time warp and breaking the monotonic constraint. This lack of continuity does not impact applications such as recognising or classifying motions or time series data, however, it is undesirable in live performance scenarios, where input frames are being played back for visualisation purposes as they are being aligned.

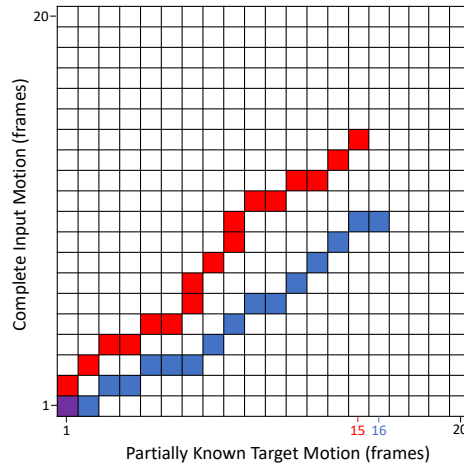


Figure 5.1: An example of the lack of continuity between the alignment paths plotted for sequential frames of a target motion, when the mapping of the last incoming target frame is unconstrained. The input frames aligned to frames 15 and 16 break the monotonic constraint resulting in backwards jumps in the playback of the input motion.

In this chapter, two novel approaches to on-line time warping are proposed and evaluated, that impose a monotonic constraint. Both approaches plot temporal alignments in a forwards direction, but utilise a smoothing algorithm to predict the future frames of the incomplete motion over a small window of time. This allows the next step in the alignment path to be determined by comparing the predicted frames of the incomplete motion with a window of frames within the complete motion. One approach uses DTW to plot an alignment between the predicted frames of target motion and the select frames of the input motion, using the first steps in the plotted alignment to determine the next step in the alignment path. The other approach determines the next step in the alignment, directly from an accumulated cost matrix. The total costs are accumulated starting at frame (m, n) and ending at frame $(0, 0)$, the opposite direction to that used by DTW. An optimal alignment point, for the incoming target frame, is then selected from the first column of the accumulated cost matrix.

The performance of the on-line time warping algorithms is to be tested using a variety of robust objective techniques, to measure the accuracy and quality of alignment paths created. The results of the tests are to be evaluated and discussed, to explore the characteristics of these algorithms and determine if they can support on-line applications. Specifically the algorithms will need to produce alignments which satisfy two key benchmarks for a majority of the test performed: i) produce a more accurate alignment than can be achieved using UTW; and ii) produce an alignment so close to that achieved using DTW that the difference would typically not be perceptible.

UTW is a uniform time warp that can be achieved by simply stretching or shrinking the duration of a motion. Although it is an offline process that requires complete knowledge of both motions, to be considered useful a time warping algorithm should be able to align a motion more accurately than UTW.

5.2 Background

This section provides key background information that has informed this study in addition to the background information already presented in Chapters 2 and 3. In this section the concept of forward plotting is outlined and existing methods for forecasting time series data using smoothing and measuring the accuracy of temporal alignments are discussed.

5.2.1 Forward Plotting

Applications which need to visualise character's performance as it is being aligned, such as: training; motor rehabilitation; live performance; and virtual production scenarios, require a smooth on-line alignment without the previously mentioned temporal jitter, that can occur when plotting an alignment path backwards. This has motivated the approaches proposed in this chapter, which use forward plotting to achieved a smooth monotonic alignment. In this study, each frame of prerecorded input motion is aligned to an incoming target motion, plotting in a forward direction as the frames of the target motion are captured and without knowledge of future

frames of the target motion.

A consequence of the forward plotting is that an error in one step of the alignment can often lead to additional larger errors in subsequent steps of the alignment. These errors can accumulate leading to an alignment path that diverges significantly from the optimal solution. Figure 3.4 shows how forward plotting determines the next alignment step based on local values within the cost matrix. While DTW which can determine a solution which is optimal for the entire sequence, the on-line warps dependence on local values can lead to errors.

A naive approach to forward plotting of an alignment was presented called OPW (On-line Predictive Warping) (Randall, Williams and Athwal, 2017). Future frames of the incomplete motion were predicted using a smoothing algorithm, then a distance metric based on joint positions was used to decide if the remaining portion of the motion being aligned should be warped to make it shorter (sped up) or longer (slowed down) in length. The approach worked and demonstrated the feasibility of using predicted frames to reduce latency in an on-line alignment. The approach of warping the remaining motion rather than mapping specific frames had weaknesses, as the latency between changing the speed of the motions playback and achieving the correct alignment often led to over compensation with unnecessary extra warps being applied between these points.

The approach was tested using synthesised misalignments, aligning two copies of the same recorded motion, with a warp applied to one version. While this approach simulated temporal offsets between motions, it did not simulate positional or spatial offsets that naturally occur between the motions. For example two separate captures of a walking motion, starting on the same foot, with a matching number of steps and recorded on the same actor, will still have naturally occurring temporal and spatial deviations between the two captures. As time warping methods use the positional and rotational distances between joints to determine an optimal alignment, the spatial differences will impact the performance of a time warping algorithm. Therefore a test data set comprising of only temporal deviations between motions, would not be relevant to real world applications in which spatial differences also occur.

5.2.2 Smoothing Methods

The approaches to on-line time warping proposed in this study utilise a smoothing algorithm to predict frame values in the future. The main smoothing algorithms available for this purpose are as follows:

- **Dead reckoning:** Assumes that the current rate of change in value x_t is going to continue, creating an estimated prediction without using smoothing.

$$\hat{x}_{t+1} = x_t + (x_t - x_{t-1}) \quad (5.1)$$

This approach can be adapted to be based on a moving average of the last k values.

$$\hat{x}_{t+1} = x_t + \frac{x_t - x_{t-k}}{k} \quad (5.2)$$

- **Weighted moving average:** Uses a weighed moving average of the most recent values to determine a smoothed value.

$$\hat{x}_{t+1} = \sum_{n=1}^k w_n x_{t+1-n} \quad (5.3)$$

The sum of the of weight factor $\{w_1, w_2, \dots, w_k\}$ must equal 1 such that:

$$\sum_{n=1}^k w_n = 1 \quad (5.4)$$

- **Exponential smoothing:** Uses a weighted average between the current value x_t and previous predicted values \hat{x}_{t-1} , to smooth a time series.

$$\hat{x}_t = \alpha x_t + (1 - \alpha)\hat{x}_{t-1} \quad (5.5)$$

An α value of near 0 is weighted towards previously predicted values and greater smoothing, while and an α near 1 is weighted towards original time

series without any smoothing applied. α must satisfy $0 < \alpha < 1$.

- **Holt’s Exponential Smoothing (HES):** Extends exponential smoothing, by also applying it to 1st order derivative values, also taking account of changes in velocity [Kalekar et al. \(2004\)](#). Both value (position) and its derivative (velocity) have their own respective smoothing parameters, α and β .

$$\hat{p}_t = \alpha p_t + (1 - \alpha)(\hat{p}_{t-1} + \hat{v}_{t-1}) \quad (5.6)$$

$$\hat{v}_t = \beta(\hat{p}_t - \hat{p}_{t-1}) + (1 - \beta)\hat{v}_{t-1} \quad (5.7)$$

([Stakem and AlRegib, 2009](#)) compared the performance of different predictive smoothing algorithms as predicting the movement of a user’s hand, to facilitate user interaction with virtual objects. The study showed that exponential smoothing performed similarly to dead reckoning, but also proposed a new technique called AHES (Adaptive Exponential Smoothing), in which the smoothing parameters α and β used in double exponential smoothing automatically adapt themselves. This approach was shown to cope better with sudden twitch movements than dead reckoning and exponential smoothing, but HES was only more accurate than dead reckoning periods less than 40 milliseconds, when forecasting for periods over 40 milliseconds all the methods performed equally well and dead reckoning out performs HES and AHES when predicting further than 200 milliseconds. This is due to samples predicted further into the future being less likely to be correlated with the samples used to make the prediction, a phenomena which is more likely to effect more optimised techniques such as HES and AHES. It should also be noted that approaches such as double exponential smoothing should be used with time series that exhibit an underlying trend, this is often not the case with motion data, especially with cyclic motions.

5.2.3 Measuring Alignment

Many on-line time warping algorithms are evaluated based on their ability to accurately recognise a sequence of data, rather than their ability to align a data sequence.

However, the accuracy of the temporal alignment resulting from the time warp is more important to this study. Therefore this section reviews existing approaches to measuring the alignment of human motion.

Any meaningful comparison of time warping methods requires robust and relevant testing methods. [Hülsmann et al. \(2017\)](#) identified and labelled segments within each motion of the data set, then measured the percentage of corresponding frames in the aligned and reference motion which were in matching segments. This approach is particularly appropriate to studies using on-line time warping to classify or recognise motions, but does not measure the overall quality or accuracy of the alignment. This approach requires motions to be segmented and labelled in a consistent manner which is not feasible for the large data-sets.

Similarity metrics are commonly used to evaluate alignments by measuring the difference between the input motion and the target motion. The warped input motion with the smallest deviation from the target motion, and therefore greatest similarity, is considered better aligned. As discussed in a previous chapter there are a wide variety of distance based similarity metrics to choose from ([Yang and Guan, 2005](#); [Arikan and Forsyth, 2002](#); [Kovar, Gleicher and Pighin, 2002](#)).

The results in chapter 4 highlighted issues with using distance metrics as they are overly effected by global differences across all the data points of a motion or differences in the amplitude or exaggeration of a motion. This motivated [Etemad and Arya \(2015\)](#) to propose using correlation as a similarity metric, with optimal approaches for implementing this explored within the study presented in the previous chapter.

[Folgado et al. \(2018\)](#) proposed a distance metric for measuring the temporal alignment of two motions, called TAM (Time Alignment Measurement), based on the alignment path required to align one motion to the other. TAM measures the ratio between the number of out of phase alignment points, which appear as vertical or horizontal segments within the alignment path, and in phase alignment points, which appear as diagonal segments within the alignment path.

Zalkow et al. (2017) used a triple-based transfer approach to measure the accuracy of transferring music annotations, by temporally aligning audio recordings. Annotations from recording a are transferred in a ring through three different recordings $a \rightarrow b \rightarrow c \rightarrow a$ creating \hat{a} , the position of the annotations in \hat{a} and a are then compared giving an error value. This approach could be adapted to test time warping algorithms, aligning motion a to b to create a^b , then aligning a^b to c to create a^{bc} , and finally aligning a^{bc} back to a to create a^{bca} . This approach, however, is ultimately aligning a warped version of a time series, \hat{a} , back to the original version of the same time series a . It therefore falls into the same pitfall as the naive study discussed in Chapter 1, in which the final motions being aligned are only temporally displaced, and do not simulate the spatial displacements that would naturally occur between two motions. In addition, to facilitate a relevant real world evaluation of the performance of time warping algorithms, alignments resulting from single time warps need to be evaluated rather than those culminating from multiple time warps.

As well as the accuracy of the alignment, other qualities can be measured such as distortion (Etemad and Arya, 2015). This considers how efficiently a motion has been manipulated during the alignment process, as well as the overall smoothness of the warp. For example sudden or unnecessary changes in the direction of the alignment path, will result in higher distortion.

To contextualise and add meaning to these measurements, it is desirable to understand how they relate to human perception. Hoyet, McDonnell and O’Sullivan (2012) studied the ability for viewers to perceive timing errors between the cause and effect of character interactions in computer animated scenes. The study determined that on average viewers are unable to perceive timing errors of less than 150ms. This is significantly longer than the average perception of interaction delays in rigid body objects, which is 60ms (Reitsma and O’Sullivan, 2009). These findings suggest that a time warp that deviates less than 150ms, at any given point, from the time warp determined using DTW, would be imperceptible from the DTW alignment when applied to a human motion. This would form a useful benchmark to assess a time warping algorithm against.

5.3 Methodology

5.3.1 Overview

The purpose of this study is evaluate and compare the performance of different approaches to on-line time warping of human motion, with the aim of understanding how they would potentially behave in given applications. As with prior studies focused on the time-warping of human motion data, there is particular interest in evaluating how optimal the resulting aligned motions are, both in terms of the accuracy of the alignment, and the impact of any distortion on the quality of the resulting aligned motion. As this study is evaluating on-line approaches there is an additional need to measure the computational performance of the algorithms being used within each approach.

There are three key elements to the study’s design. First, a data-set of human motions needs to be compiled. It is important for the motions in the data-set to be free of capture errors (such as: dropped frames; foot skate; or jittery motion), represent a suitable range of human motions and be large enough to produce results that are statistically reliable. Second, once the data-set is compiled, each alignment algorithm will be applied to this same set of motions. The algorithms will need to be implemented consistently and in a manner that is relevant to how they might be used in practical applications. Third the resulting aligned motions outputted from each algorithm need to be analysed, as discussed above, to evaluate different aspects of their performance. Each of these elements are described in more detail in the sections below.

5.3.2 Sourcing and preparing the data-set

Sourcing the data-set

The human motions within the data-set are recordings of human movement obtained using a motion capture system. These motions can be obtained by recording a subject or subjects with a motion capture system; or sourced from an existing publicly available data-set of human motions.

Two human motion data-sets that are widely used are the HDM05 database ([Müller et al., 2007](#)) and the CMU (Carnegie Mellon University) Graphics Lab Motion Capture Database ([CMU Graphics Lab, 2001](#)). They have been used to support the study of a variety of topics such as: human action recognition ([Yan, Xiong and Lin, 2018](#)), motion synthesis ([Holden, Saito and Komura, 2016](#)) and the representation of human motion ([Han et al., 2017](#)).

The CMU and HDM05 data-sets were published in 2001 and 2007 respectively and were both recorded using a Vicon MX system with twelve cameras, at a sample rate of 120Hz. The Vicon MX system was considered the industry standard benchmark at the time. The data-sets both have issues with missing data where frames are occasionally dropped, an issue that was not uncommon with motion capture systems of that era, where the lower powered computers often struggled to keep up with the processing requirements of the 120Hz sample rate.

The data-set to be created for this study requires many repeated recordings, sometimes referred to as takes, of the same motion, which are to be aligned to each other during the study. While the CMU data-set is well documented and contains a broad range motions, only a small number of them are repeated. The HDM05 data-set contains a more limited set of scripted motions which have been recorded many times by five different actors. This makes the HDM05 data-set more suitable for this study.

To diversify the range of sources used to create the data-set, it will also include motions recorded within BCU in 2018, which can be accessed through [Randall \(2022a\)](#). These motions were captured using a more recent iteration of the Vicon motion capture system. These are the same set of motions used the previous study, recorded on a system consisting of eight 2.2 megapixel Vero 2.2 cameras, recording at a sample rate of 120fps and configured into a 7x7 metre capture volume. The process of capturing these motion is presented in more detail in Section [4.3.2](#). Unlike the CMU and HDM05 datasets, these motions do not contain any dropped frames as the use of a high specification workstation based on a Xeon processor prevented this from occurring.

Preparing and verifying the data-set

To prepare the recorded motions for use in this study they are all re-targeted to a single skeletal joint system, using Autodesk MotionBuilder and then saved to .fbx format. While the motions recorded within BCU were all recorded on the same actor and joint system, within the HDM05 data-set, discrete recordings of the same scripted motions were recorded on five different actors, each with a separate joint system. Re-targeting all motions in the data-set to the same joint system ensures that they are consistently represented across all motions, and avoids problems and errors caused by differences in joint names or joint lengths. It also ensures that the respective joints of every motion in the data set match the same orientations shown in Figure 5.2, when their rotational transforms are set to zero, an essential prerequisite for many of the algorithms to be used in this study.

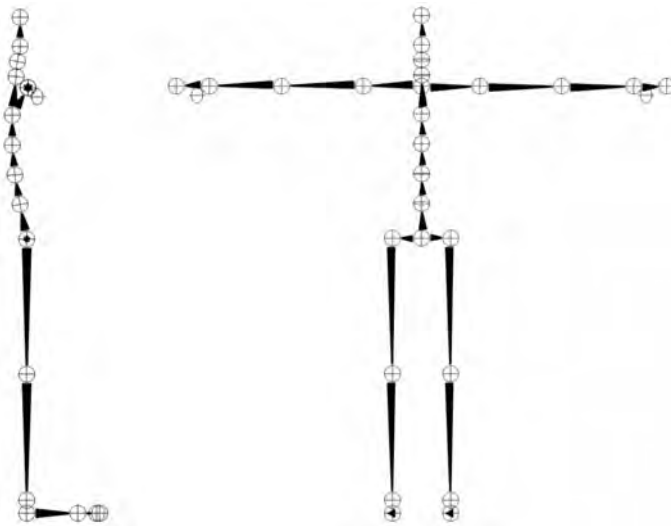


Figure 5.2: The skeletal joint rig used in this study, showing the 'T' pose the joints form when their orientations are set to zero.

During the re-targeting process redundant axes in the elbow and knee joints, which are not being used, are constrained to 0° . This prevents these axes from being used when solving the IK to fit joint system to the motion being re-targeted. It also removes any potential low level residual noise that is often left over from the motion capture process.

The motions were stored in the .fbx format. FBX is a open format developed by Autodesk for exchanging 3D assets between its own and third party software, and

has been adopted as a standard in a variety of industries. The FBX format is an ideal choice for this study, it can be accessed by both Autodesk MotionBuilder ([Autodesk, 2021](#)) and Python, using the FBX SDK ([Autodesk, 2020](#)) available from Autodesk.

All motions used in the data-set must be checked for any errors such as dropped frames, significant errors in recorded motion or joints moving beyond the range of typical human motion. Depending on the configuration of the joint system, the axis of some joints, typically in the elbows and knees, become redundant. This means that these axes are not required to encode the motion data and tend not to contain any key-frame data. To allow each frame to be processed independently, streamlining the processing of the motion data, all the motions in the data-set were re-sampled, to fill in drop frames and add key-frames to redundant axes. Every motion was re-sampled at 120Hz, using the Python FBX SDK to interpolate between frames of motion data.

All motions in the data-set were visually checked for obvious errors such as: footskate, poses with a high degree of abnormality and sudden popping of joints. Motions were visually assessed by playing them back at normal speed. Any motions exhibiting errors were removed from the data-set. The purpose of this check was to identify motions with obvious errors, not to assess their perceptual realism or levels of noise in capture data. Therefore, the check did not involve a detailed inspection of the motion curves.

Verifying range of motion

The motions within the data-set need to be checked to make sure that joints are not being articulated outside of the possible Range of Motion (ROM) for human movement. The process for checking the ROM of each motion in the data set is described in more detail in this section. To prevent this task from becoming infeasible, only the joints used in this study are checked. As with the previous study, only a subset of joints which are considered most pertinent to everyday motions are used in this study. This subset consists of both left and right: shoulders, elbows,

hips and knees.

When determining the parameters of acceptable ROMs for this study, it is important to avoid specifying overly constrained ROMs. There are plenty of reliable sources which specify the average ROM for various human joints (Norkin and White, 2016), however, the motions in this study need to be checked against the maximum possible ROM, only eliminating motions which are clear outliers. The acceptable ROMs for this study are based on Boone and Azen (1979), as the study published the standard deviations of the sampled ROMs as well as the averages, allowing a total variance to be determined. More specifically the results for the males over 19 years old group were used, as this matches the age range and gender of the actors used to record the motions within the data-sets used. To avoid specifying an overly constrained set of ROMs, a generous interpretation was made of variance in the data using Equation 5.8, where J_{max} is the maximum possible range of movement for joint J .

$$J_{max} = J_{\mu} + J_{\sigma}^2 \quad (5.8)$$

The ROMs used to check the motions against are specified in table in Appendix A, along with the corresponding joints and axis these limits are applied to. Any motions with joints rotating beyond these limits were removed from the data-set.

Although the joint system used in this study is fairly typical, the way the ROMs relate to joints and axis may vary when applied to other joint systems.

Relative to other joints fewer ROMs were specified for the shoulder. This is due to the ROM in shoulder being particularly complex to model, with the ROM in one axis being dependent on the orientation of the joint in another axis. It was also unnecessary to check the ROMs of redundant axes in the elbow and knee joints.

The Final Data-set

The motions used in this study's data-set can be seen in Table 5.1. The number of takes refers the number of unique records of each motion. Each permutation of two takes of the same motion will be used to test the alignment algorithms in this study,

providing a total of 3248 tests. For each test one motion, referred to as the input, will be warped to be temporally aligned with the other motion, referred to as the target. Only the input motion is manipulated in the time warping process.

The table also identifies which of the following types of movement occur within each motion: Cyclic; Non-Cyclic; Ballistic; Lower Body and Upper Body. The Cyclic and non-cyclic movement types, identify if periodic repetitions of movement exists in the motion, such as a walk cycle or multiple kicks. The interest in cyclic movement is motivated by problems that can occur when time warping cyclic motions, as the algorithm can get confused between which cycles in an input motion to align with which cycles in the target motion. Ballistic movement refers to motions which contain moments where there is no contact with the ground, such as jumps. These are of interest, as at these moments, the motion is largely determined by the physical forces acting on the actor, rather than the actions of the actor themselves. The temporal distortion caused by time warping, will potentially have a bigger negative impact on the perceived realism of a motion during these ballistic non-contact moments, than at other moments. Lower body and upper body movement, identifies if a motion contains movement in upper and/or lower body. Identifying this will allow the relationship between the joints used in motion and joint weightings to be explored. The representation of each movement type within the data-set can also be seen within the table.

A power analysis was used to ensure that a sufficient number of tests are performed to have confidence that differences in results produced by each alignment algorithm can be relied on. The power analysis was based on the results of the previous study. The two closest sets of results, across all of the various motion alignment algorithms tested, were identified using the Mann-Whitney U test with SciPy (Virtanen et al., 2020) for comparing two non-parametric sample groups (Haslwanter, 2016). The means (μ_1 and μ_2) and standard deviations (σ_1 and σ_2) of the of the two sample groups were used with Cohen's d Equation 5.9 to determine the effect size, which was used along with standard values for $power = 0.8$ and $\alpha = 0.05$ to determine the required sample size using Statsmodel (Seabold and Perktold, 2010). This power analysis suggested a sample size of 2599, which the study's data-set exceeds.

$$d = \frac{\mu_2 - \mu_1}{\sqrt{\frac{\sigma_1^2 + \sigma_2^2}{2}}} \quad (5.9)$$

5.3.3 On-line time-warping algorithms

Within this study on-line time-warping of human motion refers to aligning a prerecorded input motion to a live streamed target motion. This process monotonically matches every streamed target frame to a prerecorded input frame, resulting in frames from the prerecorded input motion being inserted and deleted as necessary, while the incoming target frames are unaffected.

There are two main challenges to on-line time-warping of human motion. First, aligning frames of motion data in real-time, processing each frame faster than the motion's frame rate. Second, aligning the frames of an input motion to a target motion, when during the time-warping process there is only knowledge of the target frames that have already been aligned and no knowledge of the remaining unaligned frames of the motion, that are yet to be streamed into the algorithm.

Three on-line time-warping algorithms are implemented and tested in this study, each of which use different methods to select an input frame to align to a given incoming target frame:

- Method A - Frame Matching: The optimal input frame to align with an incoming target frame, is selected from a window starting at the input frame selected for the previous target frame.
- Method B - Predictive Windowing Using DTW: The next n incoming target frames are predicted. The optimal input frame is found by plotting an alignment between the predicted target frames and a window of the input motion, using DTW.
- Method C - Predictive Windowing Using Cost Matrix: Also based on predicting the next n frames in the target motion, but with the optimal input frame being selected directly the accumulated cost matrix.

Table 5.1: The composition of motions within the data-set used for this study.

Source	Motion	No.	No. of 2 Take	Non			Lower	Upper
		Takes	Permutations	Cyclic	Cyclic	Ballistic	Body	Body
BCU	3 Jumps	3	6	x		x	x	x
BCU	Stepping Stones Walk	3	6	x			x	
BCU	Dance	3	6		x			x
BCU	High Chair Sit Walk	3	6	x	x		x	x
BCU	Jaunty Style Walk	3	6	x			x	x
BCU	Long Jump	3	6		x	x	x	x
BCU	High Chair Movment	3	6		x			x
BCU	Low Chair Movement	3	6		x			x
BCU	1 Punch 1 Kick	3	6	x			x	x
BCU	One Hand Pickup	3	6		x		x	x
BCU	Shelf Stack	3	6	x				x
BCU	Sit Walk	3	6	x	x		x	x
BCU	Slow Walk	3	6	x			x	
BCU	Sneaky Style Walk	3	6	x			x	x
BCU	Stiff Style Walk	3	6	x			x	
BCU	Walk	3	6	x			x	
BCU	Walk On All Fours	3	6	x			x	x
BCU	Walk Sit On High Chair	3	6	x	x		x	x
BCU	Walk Two Hand Pickup	3	6	x	x		x	x
HDM05	Clap 5 Reps	9	72	x				x
HDM05	Deposit Floor R Hand	7	42		x		x	x
HDM05	Elbow To Knee 3 Reps L Start	11	110	x			x	x
HDM05	Grab Middle Shelf R	13	156		x			x
HDM05	3 Jumps	12	132	x		x	x	x
HDM05	Hop L Leg 3 Hops	14	182	x		x	x	x
HDM05	Jog Left Circle 6 Steps R Start	15	210	x			x	
HDM05	Jog On Place 4 Steps R Start	14	182	x			x	x
HDM05	Jump Down	11	110		x	x	x	x
HDM05	Jumping Jack 3 Reps	13	156	x		x	x	x
HDM05	Kick R Side 2 Reps	5	20	x			x	
HDM05	Punch L Side 2 Reps	11	110	x				x
HDM05	Rotate Arms Forward 3 Reps	13	156	x				x
HDM05	Shuffle 4 Steps L Start	11	110	x			x	
HDM05	Sit DownChair	14	182		x		x	x
HDM05	Sit DownFloor	5	20		x		x	x
HDM05	Skier 3 Reps L Start	3	6	x			x	x
HDM05	Sneak 4 Steps R Start	16	240	x			x	x
HDM05	Staircase Up 3 Rstart	16	240	x			x	x
HDM05	Stand Up Sit Chair	14	182		x		x	x
HDM05	Stand Up Sit Floor	3	6		x		x	x
HDM05	Throw Far R	7	42		x	x		x
HDM05	Turn Right	13	156		x		x	
HDM05	Walk 4 Steps L Start	13	156	x			x	
HDM05	Walk Circle 6 Steps R Start	13	156	x			x	
		Totals	3248	2322	950	634	2688	2416

Before presenting each algorithm in detail, some common terms and concepts need to be defined. Let $I = \{I_0, I_1, \dots, I_m\}$ be the frames of prerecorded input motion and $T = \{T_0, T_1, \dots, T_n\}$ be frames of a live target motion which have already been received and mapped to an input frame. The incoming target frame being processed is T_{n+1} . Note that set T only consists of frames in the target motion that are already known, it is not the complete target motion. Set $M = \{M_0, M_1, \dots, M_n\}$ maps each received target frame to an input frame. Alignment of the input motion to the target motion must be monotonic and is achieved by either: mapping the same input frame to adjacent target frames (slowing the input motion down); mapping adjacent input frames to adjacent target frames (maintaining the normal speed of the input motion); or skipping frames in the input motion (deleting frames and speeding up the input motion).

Method A - Frame Matching

Frame matching is the most straight forward and naive approach to on-line time-warping. While it is expected to demonstrate some of the short-comings discussed in earlier in this chapter, it will act as a baseline with which to compare the other time warping methods.

Using Algorithm 4, Frame Matching uses a distance based cost function, based on the rotational distance of corresponding joints in each motion frame, to find the frame within a window of the prerecorded input motion $\{I_{M_n} \dots I_{M_n+w}\}$ that most closely matches an incoming target frame T_{n+1} . This search window is visualised in Figure 5.3.

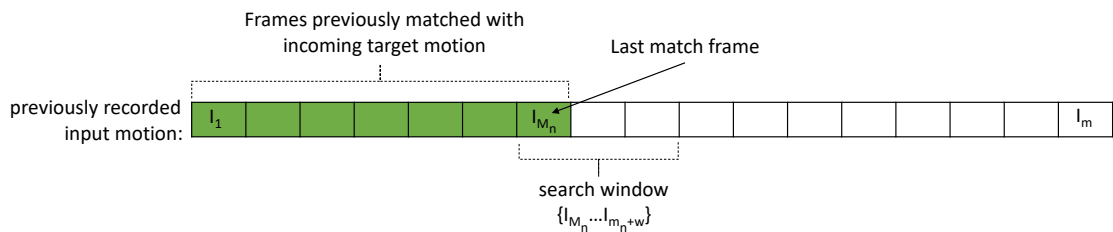


Figure 5.3: Search window used in Frame Matching algorithm, to find the pre-recorded input frame that best matches the incoming target frame

Before evaluating each of the input frames within the window, the size of the window

Algorithm 4: The Method A: Frame Matching time warping algorithm. The algorithm is applied to each captured frame of a target motion, mapping it to the best fitting frame within a specified window of the input motion.

input : Window size w
New incoming target motion frame T_{n+1}
Set of prerecorded input frames $I = \{I_0, I_1 \dots I_m\}$
Set of existing maps $M = \{M_0, M_1 \dots M_n\}$ between target frames $\{0..n\}$ and input frames in I

output: Map M_{n+1} mapping target frame T_{n+1} to an input frame in I

```

1  $C \leftarrow$  array of 3 dimensional Euler rotations
2 if  $M_n + w > m$  then
3    $w = m - M_n$ ;
4 if  $w = 0$  then
5    $M_{n+1} = M_n$ ;
6 else
7   for  $i \leftarrow 0$  to  $w$  do
8      $C_i = \text{geoDistance}(I_{M_n+i}, T_{n+1})$ ;
9    $M_{n+1} = M_n + \text{arg min}\{C_0, C_1 \dots C_w\}$ ;
```

is checked at Line 2 to ensure that there are still enough frames remaining in the input motion to cover the specified window size, if not, the window size is reduced accordingly. If the window size becomes zero the search for the best matching input frame is skipped altogether and the incoming frame is matched to the same input frame as the previous target frame.

A key consideration is the optimal size window w to use. Consideration needs be given to how this will constrain the solution. A smaller window reduces the number of input frames that can be skipped when mapping them to adjacent frames in the target motion, effectively allowing fewer frames to be deleted between each frame of the input motion when aligning it. A smaller search window, therefore, effectively reduces the number of possible alignment solutions which could be used. While this constraint prevents two motions with more extreme differences in temporal alignment from being properly aligned, overly large windows result in an under constrained search, which can cause errors such as entire motion cycles being skipped or more commonly allowing the alignment to deviate so far from the optimal solution that it not able to find its way back again.

For this study a small window size of two was selected. This approach avoids the issues of alignments deviating too far from an optimal solution in a single alignment step, mimicking a Type I local continuity constraint, which is commonly used in time warping ([Rabiner and Juang, 1993](#)). A small pilot study was used to validate this approach.

Method B - Predictive Windowing Using DTW

The nature of on-line time-warping means that incoming target sequence T is only partially known. This means that there is no way of knowing when determining an alignment point for the partial sequence T , if that point is on the optimal path for aligning the entire sequence. While method A does not try to mitigate against this problem, Methods B and C propose a windowing technique to do this, inspired by [Macrae and Dixon \(2010\)](#).

This method applies a smoothing function to sequence T to predict a set of future frames over a small forecast window w . A slight variation of the standard DTW algorithm is then used to plot an alignment path within the window, with an alignment point at the start of the path being used to align the incoming frame T_{n+1} . By choosing an alignment point for T_{n+1} based on some limited knowledge of the future of sequence T , rather than none at all, it is expected that this approach will avoid some the pitfalls of method A such as: the alignment path getting stuck and aligning all remaining frames T to the same frame in I , or plotting alignment points which are far from the optimal alignment path for the entire sequence.

The method functions by applying [Algorithm 5](#) to every incoming frame $T_n + 1$ in target sequence T . The algorithm performs the following steps: (i) check if enough target frames have been captured to forecast the motion and that there are still prerecorded input frames remaining to align, reducing window size if required; (ii) determine predictions for future frames in the target motion F ; (iii) calculate a cost matrix C based on the similarity between every frame within a window of the input sequence and the predicted frames of the incoming target motion; (iv) accumulate the costs in the cost matrix to create an accumulated cost matrix D ; (v) plot an

alignment path P through the cost matrix, and (vi) map the incoming target frame T_{n+1} using the second alignment point on the alignment path P_1 .

Algorithm 5: The Method B time warping algorithm. Determines the best input frame to map to a target frame by forecasting w frames of the target motion, then plotting an alignment between predicted target frames and a subset of frames from the input motion. The algorithm is applied to each captured frame of a target motion.

input : Set of prerecorded input frames $I = \{I_0, I_1 \dots I_m\}$
Set of already received target motion frames $T = \{T_0, T_1 \dots T_n\}$
New incoming target motion frame T_{n+1}
Set of existing maps $M = \{M_0, M_1 \dots M_n\}$ between target frames $\{0..n\}$ and input frames in I
Forecasting window size w
Smoothing window size s
Set of joints $J = \{J_0, J_1 \dots J_k\}$

output: Map M_{n+1} mapping target frame T_{n+1} to an input frame in I

- 1 $F \leftarrow$ 2D array of 3D vectors of size k (number of joints) x w (window size);
- 2 $C, D \leftarrow$ 2D array of floats size w_i (size of input motion window) x w (size of target motion window);
- 3 $P \leftarrow \{\}$ integers ;
- 4 **if** $n < s$ **then**
- 5 $M_{n+1} = n + 1$;
- 6 **else if** $M_n \geq m$ **then**
- 7 $M_{n+1} = m$;
- 8 **else**
- 9 $w_i = w$;
- 10 **if** $M_n + w_i > m$ **then**
- 11 $w_i = m - M_n$
- 12 $F_0 = T_n$;
- 13 $F = F + \text{forecastJoints}(\{T_n, T_{n+1}\}, J, w)$ // Algorithm 6 ;
- 14 $C = \text{getCostMatrix}(\{I_{M_n} \dots I_{M_n + w_i}\}, F)$ // Algorithm 1 ;
- 15 $D = \text{getTotalCostMatrix}(C)$ // Algorithm 2;
- 16 $D_{0,0} = 10000$;
- 17 $P = \text{plotPath}(D)$ // Algorithm 7;
- 18 $M_{n+1} = M_n + P_1$;

Before the DTW function is performed, checks are made to ensure a number of conditions are satisfied. First a sufficient number of frames in the target sequence need to have been processed in order to forecast the sequence. If this is not satisfied then the DTW is skipped and the incoming target frame T_{n+1} is mapped to the next unmapped frame in the input motion I_{n+1} . Second are there enough unmapped frames remaining in the input sequence to fulfill the window size being used, if this is

not the case, the window size for the input sequence w_i is adjusted accordingly and the DTW is performed on the smaller window. If the last input frame has already been mapped, then T_{n+1} is also mapped to this frame and the DTW is skipped.

Forecasting Motion Data

Method B uses a smoothing algorithm to forecast motion frames in a target motion, this can be seen in Algorithm 6. There are a number of different smoothing algorithms that can be used to predict samples in a sequence. [Stakem and AlRegib \(2009\)](#) compared a number of smoothing algorithms including dead reckoning and Holt’s Exponential Smoothing, measuring the forecasting error of each algorithm when applied to predicting the motion of a human hand. The study established that more complex smoothing algorithms, such as Holt’s Exponential Smoothing, require fine tuning of smoothing parameters to optimise them, and are only more accurate when forecasting the first 40ms of motion. Given that the motions used in this study have been sampled at 120Hz, 40ms equates to just under five frames of motion. The limited benefit of using HES, suggest that a more straight forward approach such as dead reckoning, that does not require tuning, would be more appropriate for this study.

Algorithm 6: The `forecastJoints()` Function. Given two sequential frames of a motion, this function uses dead reckoning to predict the poses of a set of joints J over the next f frames.

input : Set of two motion frames $A = \{A_0, A_1\}$
Set of joints to forecast $J = \{J_0, J_1, \dots, J_k\}$
Forecasting window size f

output: $F \leftarrow$ Predicted joint orientations as 2D Array of 3D vectors size $k \times f$

```

1  $V \leftarrow$  3D vector;
2 for  $i \leftarrow 0$  to  $k$  do
3    $V = A_{0,i} - A_{1,i}$ ;
4   for  $j \leftarrow 0$  to  $f$  do
5      $F_{i,j} = A_{1,i} + (V * (j - 1))$ ;

```

Smoothing algorithms need to be applied to an Euler representation of the joint angle. Although quaternions are used later in the Method B algorithm, for measuring the distance between angles, depending on how they are implemented, quaternions

can be non-linear relative to changes in orientation and representations of very similar angles can look very different if they are expressed on different hemispheres. This makes quaternions unsuitable for use with smoothing algorithms.

Two decisions that need to be made regarding the implementation of the dead reckoning smoothing algorithm are the number of frames that are to be predicted (i.e. the window size) and the number of frames on which to base these predictions (i.e. the smoothing level).

The optimal choice for the first parameter, window size, is determined by testing different window sizes within the study. The window sizes selected for this study were mostly informed by the typical period of a walk cycle, a frequently reoccurring cycle in human motions. Given that a normal walking gait cycle has a frequency of between 0.82 and 0.9Hz (Ekimov and Sabatier, 2011), and the motions in this study are sampled at 120Hz. The period of a typical single walking gait cycle in frames, t , can be determined using Equation 5.10, where f_w and f_s are the frequencies of the walk cycle and the sample rate of the motion capture respectively. This gives a typical walk cycle period of between 133 and 146 frames. Window sizes of more than half a walk cycle are unlikely to be optimal, as beyond this point the potential for the incoming frame of the target motion to have multiple local minima points with frames within the input motion window, will increase, also increasing the potential for incorrect alignments. Taking the slowest typical walk cycle, it is expected that an optimal window size will be below 73 frames. To cover the range of potential window sizes more efficiently, the following non linear distribution of window sizes are tested in this study: 10, 20, 40 and 80 frames. Including a window size of more than 73 frames will allow the hypothesis that windows sizes larger than half a walk cycle are less optimal, to be tested.

$$t = \frac{1}{f_w} f_s \quad (5.10)$$

The optimal choice of smoothing level can be determined by testing them on the motion data-set, and comparing the predicted motion frames with their respective

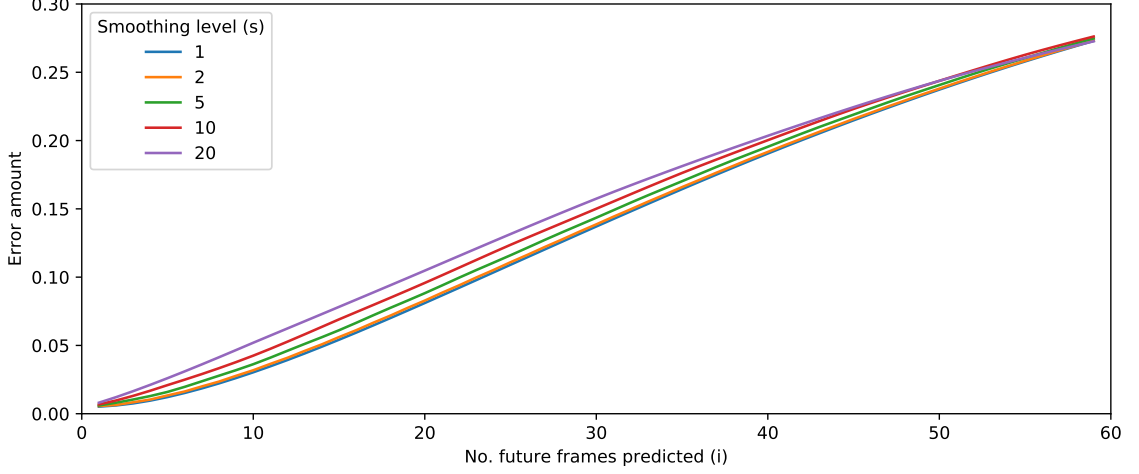


Figure 5.4: The performance of different smoothing levels at predicting human motion. Shows the error resulting from a given number of frames used to predict a motion (smoothing level s) i frames in to the future.

frames in original recorded motions. Equation 5.11 was applied to the entire data-set to evaluate the performance of different smoothing levels s at predicting i frames into the future, where D is a data-set of p motions, each containing q frames, with r joints being evaluated. The results of the test in Figure 5.4 shows that the smaller smoothing levels perform best for predicting up to 50 frames, after this all the smoothing levels perform equally. Again this shows that more optimised, smaller, smoothing levels become less effective as the motion is predicted further into the future, where frames are less likely to correlate with the frames used to make the prediction. Based on this test, the forecasting of motion frames in this study will be based on a smoothing level of one, with the differential between the two most recently captured target frames forming the basis for forecasting frames using dead reckoning.

$$e_{s,i} = \frac{\sum_{m=1}^p \sum_{f=s+1}^{q-i} \sum_{j=1}^r \text{geoDist}(D_{m,f+i,j}, D_{m,f,j} + (v'i))}{\left(\sum_{m=1}^p q - i - s - 1 \right) r} \quad (5.11)$$

$$v' = \frac{D_{m,f,j} - D_{m,f-s,j}}{s} \quad (5.12)$$

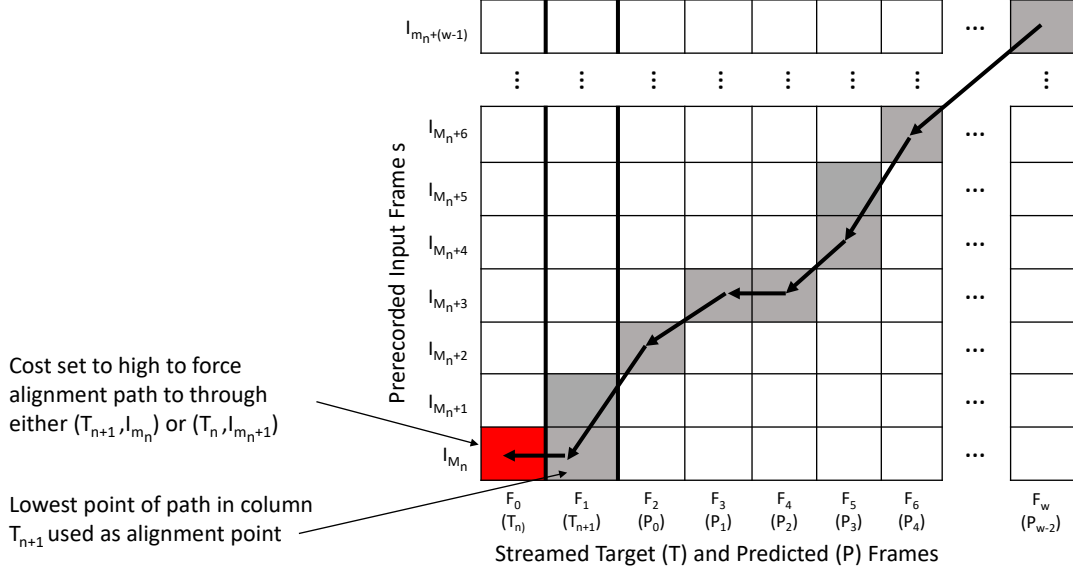


Figure 5.5: Plotting of path aligning prerecorded frames $\{I_{M_n} \dots I_{M_n+(w-1)}\}$ against a mixture of existing frames $\{T_n, T_{n+1}\}$ and predicted target frames $\{P_0 \dots P_{w-2}\}$, contained in set F .

Determining Optimal Alignment Within Method B

Following the standard DTW approach, the next steps are to calculate the cost matrix C and total accumulated cost matrix D , comparing the frames of the predicted target motion F against those within the window of prerecorded input motion I . The costs are determined using the sum of rotational distances between corresponding joints in each frame. The calculation is restricted to a subset of joints, which are converted to quaternions to allow geometric distance between them to be calculated more efficiently.

Figure 5.5 shows an alignment path plotted through the accumulated cost matrix D , using the path plotting Algorithm 7. Apart from a minor change this is the same path plotting algorithm used in the standard DTW algorithm. When plotting a path each frame in F is monotonically match to a single frame in I . If the alignment path steps through multiple frames of I within a single frame of F , as seen in F_1 and F_5 , the frame with lowest index is matched. To facilitate this a single line was added to the standard DTW function at Line 22 in Algorithm 7, to overwrite an existing match of F if the path should step down a frame in I .

Algorithm 7: The plotPath() Function. Plots an alignment path through accumulated cost matrix D , using a slightly adapted version of the established approach used in DTW.

input : Matrix D of size $m \times n$

output: Set P alignment map

```

1  $R \leftarrow \{0,0\}$ ;
2  $p \leftarrow$  next step as string;
3  $R = \{m, n\}$ ;
4  $P = P \cup \{R_0\}$ ;
5 while  $R_0 > 0 \vee R_1 > 0$  do
6    $p =$  "match";
7   if  $R_1 = 0$  then
8      $p =$  "delete";
9   else if  $R_0 = 0$  then
10     $p =$  "insert";
11  else
12    if  $\arg \min\{D_{R_0-1, R_1-1}, D_{R_0-1, R_1}, D_{R_0, R_1-1}\} = 1$  then  $p =$  "delete";
13    if  $\arg \min\{D_{R_0-1, R_1-1}, D_{R_0-1, R_1}, D_{R_0, R_1-1}\} = 2$  then  $p =$  "insert";
14  if  $p =$  "match" then
15     $R = \{R_0 - 1, R_1 - 1\}$ ;
16     $P = \{R_0\} \cup P$ ;
17  if  $p =$  "insert" then
18     $R_1 = R_1 - 1$ ;
19     $P = \{R_0\} \cup P$ ;
20  if  $p =$  "delete" then
21     $R_0 = R_0 - 1$ ;
22     $P_0 = R_0$ ;

```

The frame in I that is aligned to F_1 in the alignment plot gives the optimal frame to be mapped to T_{n+1} . Remember T_{n+1} was assigned to F_1 earlier in Algorithm 7. Figure 5.6 shows how various alignment paths result in different frames in I being mapped to T_{n+1} , allowing frames in I to be inserted, matched and deleted as required to align it with the target motion T .

The path plotting algorithm will always create a path that ends at $D_{(0,0)}$ of the accumulated cost matrix. This problem motivated the addition of the last mapped frame T_n into the cost matrix. If it was not included then T_{n+1} would always be mapped to I_{M_n} , in the same way that T_n is always mapped to I_{M_n} in Figure 5.6. Adding frame T_n , allows frame T_{n+1} to be mapped to the lowest point of the

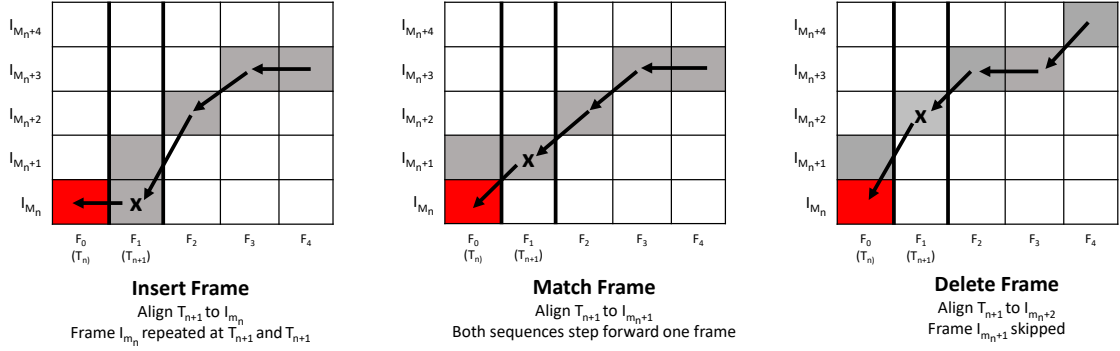


Figure 5.6: How the alignment path is used to determine if frames in input motion I are inserted, matched or deleted to align it with a target motion T , using the frame in I that is aligned to T_{n+1}

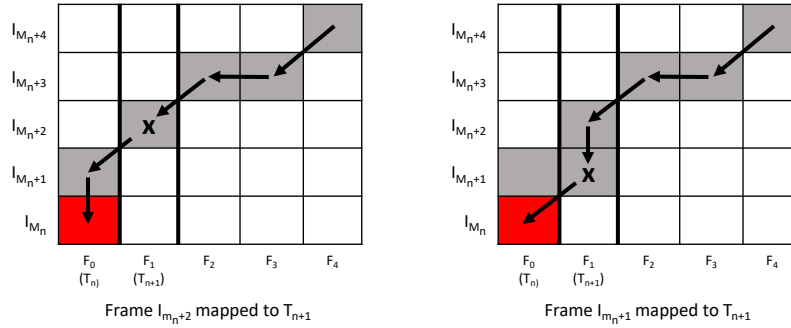


Figure 5.7: Determining which frame in input motion I to map to target motion T based the lowest cell in frame T_{n+1} the path goes through. After the path plots through $(T_{n+1}, I_{M_{n+2}})$, it moves to frame, T_n , in the left example and continuous downwards in the same frame T_{n+1} in the right example. As only one frame of I can be mapped to frame T_{n+1} , in the right example the mapping of frame T_{n+1} is replaced as the plot continuous downwards in that frame, eventually resulting in the lowest point the the path plots in frame T_{n+1} being mapped to that frame.

alignment path in frame T_{n+1} . Without the inclusion frame T_n this would always be I_{M_n} , however, with inclusion of T_n the alignment path can move to this frame as it completes it journey to (T_n, I_{M_n}) , when the frames in (T_{n+1}) become less optimal, as shown in Figure 5.7, allowing the most optimal alignment of frame (T_{n+1}) to be identified.

Another path plotting requirement, which is specific to the Method B algorithm, is the need to prevent the alignment path from stepping directly from cell $(T_{n+1}, I_{M_{n+1}})$ to cell (T_n, I_{M_n}) . Due the manner in which the costs in the accumulated cost matrix are determined, cells $(T_n, I_{M_{n+1}})$ and (T_{n+1}, I_{M_n}) will always be more expensive than cell (T_n, I_{M_n}) , preventing an alignment path from being plotted through them.

Uncontrolled this would prevent a path being plotted through $(T_{n+1}, I_{M_{n+1}})$ and a frame being inserted into the time warp alignment as shown on the left most example in Figure 5.6. To mitigate this, after the accumulated cost matrix is determined, the cost of cell (T_n, I_{M_n}) is set to a very high value in Line 16 of Algorithm 5, forcing the path to go through either cell $(T_n, I_{M_{n+1}})$ or (T_{n+1}, I_{M_n}) before going to cell (T_n, I_{M_n}) .

Method C - Predictive Windowing Using Cost Matrix

This method is a modification of method B, which removes the need to plot an alignment path through the accumulated cost matrix D . By reversing the direction in which the costs are accumulated, the optimal frame within input motion I to map to target motion T_{n+1} , can be determined directly from the accumulated cost matrix. The motivation for this approach was to bypass the complexities associated with plotting and interpreting alignment paths in method B, to create a more streamlined solution which potentially requires less computation.

The Method C Algorithm 8, performs initial checks, predicts joint positions F and calculates a cost matrix C in the same manner as Method B. The only difference is that T_n no longer needs to be added as the first element of F .

At Line 13, Algorithm 8 calculates an accumulated cost matrix, D , in a reverse direction using Algorithm 9. Unlike the established approach of Algorithm 2, utilised in Method B, in which the costs are accumulated from $D_{0,0}$ to $D_{m,n}$, Algorithm 9 accumulates the costs in the opposite direction starting at $D_{m,n}$ and ending at $D_{0,0}$.

The optimal choice of input frame to map to the target frame T_{n+1} can be determined using the first column of the accumulated cost matrix, as this represents the cost of matching each frame in input window $\{I_{M_n}..I_{M_n+w}\}$ to the incoming target frame T_{n+1} . Therefore, the input frame corresponding to the cell with the least accumulated cost in first column is mapped to T_{n+1} .

This method will also be tested with window sizes 10, 20, 40, and 80, the same as Method B, to determine the optimal window size to use for this method.

Algorithm 8: The Method C time warping algorithm. Determines the optimal input frame to map to a given target frame, by forecasting w frames of the target motion, then using an accumulated cost matrix between the predicted frames and a subset of frames from the input motion, to determine the optimal input frame to map. The algorithm is applied to each captured frame of a target motion.

input : Set of prerecorded input frames $I = \{I_0, I_1 \dots I_m\}$
Set of already received target motion frames $T = \{T_0, T_1 \dots T_n\}$
New incoming target motion frame T_{n+1}
Set of existing maps $M = \{M_0, M_1 \dots M_n\}$ between target frames $\{0..n\}$ and input frames in I
Forecasting window size w
Smoothing window size s
Set of joints $J = \{J_0, J_1 \dots J_k\}$

output: Map M_{n+1} mapping target frame T_{n+1} to an input frame in I

```

1  $F \leftarrow$  2D array of 3D vectors of size  $k \times w$ ;
2  $C, D \leftarrow$  2D array of floats size  $w_i \times w$ ;
3 if  $n < s$  then
4   |  $M_{n+1} = n + 1$ ;
5 else if  $M_n \geq m$  then
6   |  $M_{n+1} = m$ ;
7 else
8   |  $w_i = w$ ;
9   | if  $M_n + w_i > m$  then
10  |   |  $w_i = m - M_n$ ;
11  |   |  $F = \text{forecastJoints}(\{T_n, T_{n+1}\}, J, w)$ ;
12  |   |  $C = \text{getCostMatrix}(\{I_{M_n} \dots I_{M_n + w_i}\}, F)$ ;
13  |   |  $D = \text{getReverseTotalCostMatrix}(C)$ ;
14  |   |  $M_{n+1} = M_n + \text{arg min}\{D_{0,0} \dots D_{m,0}\}$ ;

```

5.3.4 Evaluating the performance of time-warping algorithms

The performance of the three on-line time warping methods, and the impact of different windows sizes on the predictive window based methods, Methods B and C, need to be measured to allow them to be appropriately compared and evaluated. The following four approaches will be used to measure contrasting aspects of each algorithms performance: (i) Alignment, a similarity metric will be used to measure how similar the resulting aligned motion is to the target motion it was aligned to; (ii) Accuracy, how similar is the on-line alignment solution to that produced by the standard offline DTW algorithm; (iii) Distortion, how much was the input motion distorted by the time-warping algorithm; and (iv) Computation, how much processing power does each algorithm require to match an input frame to each frame

Algorithm 9: The getReverseTotalCostMatix() Function. Accumulates the costs in cost matrix C starting at (m, n) and ending at $(0, 0)$. The opposite direction to that used within a standard DTW algorithm.

input : Matrix C of size $m \times n$
output: Matrix D of size $m \times n$

```

1 for  $i \leftarrow m - 1$  to 0 by  $-1$  do
2   for  $j \leftarrow n - 1$  to 0 by  $-1$  do
3     if  $i = m - 1 \wedge j = n - 1$  then
4        $D_{m-1, n-1} = C_{m-1, n-1}$ ;
5     else
6        $D_{i, j} = C_{i, j} + \min\{D_{i+1, j}, D_{i, j+1}, D_{i+1, j+1}\}$ ;

```

in the target motion.

Measuring Alignment

Using the findings of Chapter 4 which showed Kendall Tau correlation applied to joints parameterised as trajectories to be the most accurate method of measuring alignment, the alignment of each aligned motion A is measured based on its correlation to the target motion T , to which it was aligned. The correlation between the corresponding motion curves of the joint axis in motions A and T are measured using Equation 5.13 where j is a joint in subset of joints $J = \{J_0, J_1 \dots J_k\}$ and a is an axis in $\{x, y, z\}$.

$$r^t(A, T) = \frac{\sum_{j=1}^k \sum_{a=1}^3 r^t(A_{j,a}, T_{j,a})}{3k} \quad (5.13)$$

Note that the alignment process will cause the aligned motions A to have same number of frames as their respective target motions T , eliminating the need to match the lengths of the motions to facilitate the correlation.

Measuring Accuracy

Alignment accuracy is measured by comparing the mapping, M , of input frames, I , to target frames, T , produced from the on-line alignment algorithm to that produced by the standard offline DTW process. This approach treats the mapping produced by standard offline DTW as the most optimal alignment solution possible. Measuring difference between the corresponding maps plotted by the offline P^a and on-line P^b algorithms for the same input and target motions, allows an A/B testing approach to be taken, in which the on-line solution can be evaluated based on how close it is to the offline solution.

Three measures of the difference between maps M^a and M^b are used: the average difference 5.14, measures the bias of the algorithm towards warping ahead or behind the optimal solution; the absolute difference, 5.15, measures how close the algorithm was to the optimal solution; and the maximum difference, 5.16, measures how far the algorithm was from the optimal solution at its furthest point. Each of the equations normalise the results to account for maps M of different lengths n .

$$D_{avg}(P^a, P^b) = \frac{\sum_{q=1}^n P_q^a - P_q^b}{n^2} \quad (5.14)$$

$$D_{abs\ avg}(P^a, P^b) = \frac{\sum_{q=1}^n |P_q^a - P_q^b|}{n^2} \quad (5.15)$$

$$D_{max}(P^a, P^b) = \frac{\max\{|P_1^a - P_1^b|, |P_2^a - P_2^b|, \dots, |P_n^a - P_n^b|\}}{n} \quad (5.16)$$

Measuring Distortion

Distortion measures the amount a motion has been warped or distorted in order in order to align it with a target motion. Achieving an alignment through a small number of smooth changes to an input motion is more desirable than using sharp

changes or unnecessary or overly aggressive warps which have to be compensated for with further warping later in the motion. Overlay or unnecessarily distorted motions are undesirable as they can potentially be perceived as unrealistic. The distortion measure in Equation 5.17, inspired by Etemad and Arya (2015), uses a signal to noise ratio approach where $SNR = \frac{|noise|}{|signal|}$. In this case the noise is the difference between the original input motion I and the input motion after alignment I' , and the signal is the original input motion. j is a joint in the subset of joints $J = \{J_0, J_1 \dots J_k\}$, a is an axis in $\{x, y, z\}$, and f is a frame of the motion.

To facilitate this, the warped version of the input motion is uniformly time-warped using UTW and re-sampled at 120Hz so that it has the same number of frames as the original input motion.

The SNR is based on the slopes of the motion curves, there are two reasons for not basing it directly on the motion curve values. First this would bias a warped motion with no alignment, as would be obtained using UTW rather than DTW. Secondly, motion curves do not naturally centre themselves around the zero value, as the difference between the two values is divided by the value from the original motion, motion curves closer to zero would have a larger influence on the results than those further away.

$$d(I', I) = \frac{\sum_{j=1}^k \sum_{a=1}^3 \sum_{f=1}^m \frac{|\Delta I'_{j,a,f} - \Delta I_{j,a,f}|}{|\Delta I_{j,a,f}|}}{k3m} \quad (5.17)$$

Measuring Computation

The computational efficiency of each algorithm will be measured based on the average time taken to align each frame of a given motion, with the same motion being used on each algorithm.

The algorithms were ran on a processor with a base speed of 2.21GHz but capable of up to 3.9GHz. The algorithms have not been implemented to take advantage

of multi-threading. When running the test, checks will be made to minimise background processes and notes will be made of the processing speed used.

5.4 Results

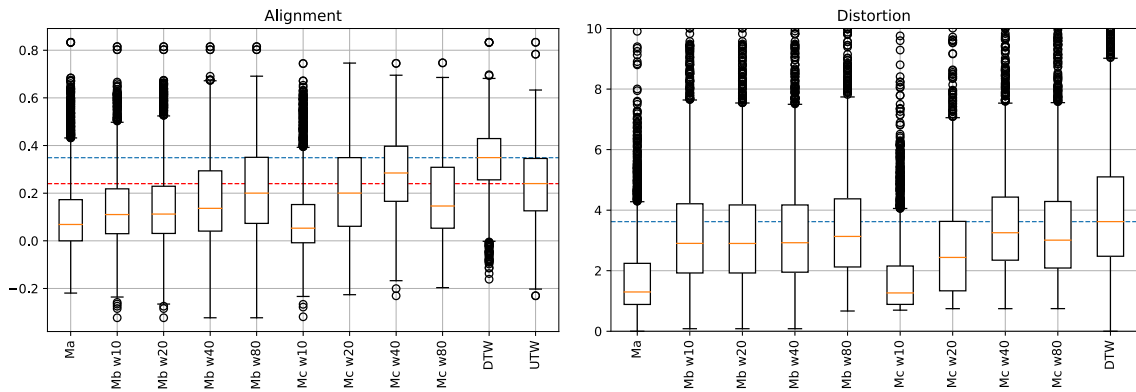
5.4.1 Interpreting the results

The results of the alignment, distortion and accuracy performance tests can be seen in Figure 5.8. The tests were used to measure the performance of three different alignment methods. The results are shown as box plots with each sample group representing aligned motions produced by a specified combination of alignment method and window size. Throughout this section Ma, Mb and Mc refer to methods A, B and C respectively, while w specifies the window size used.

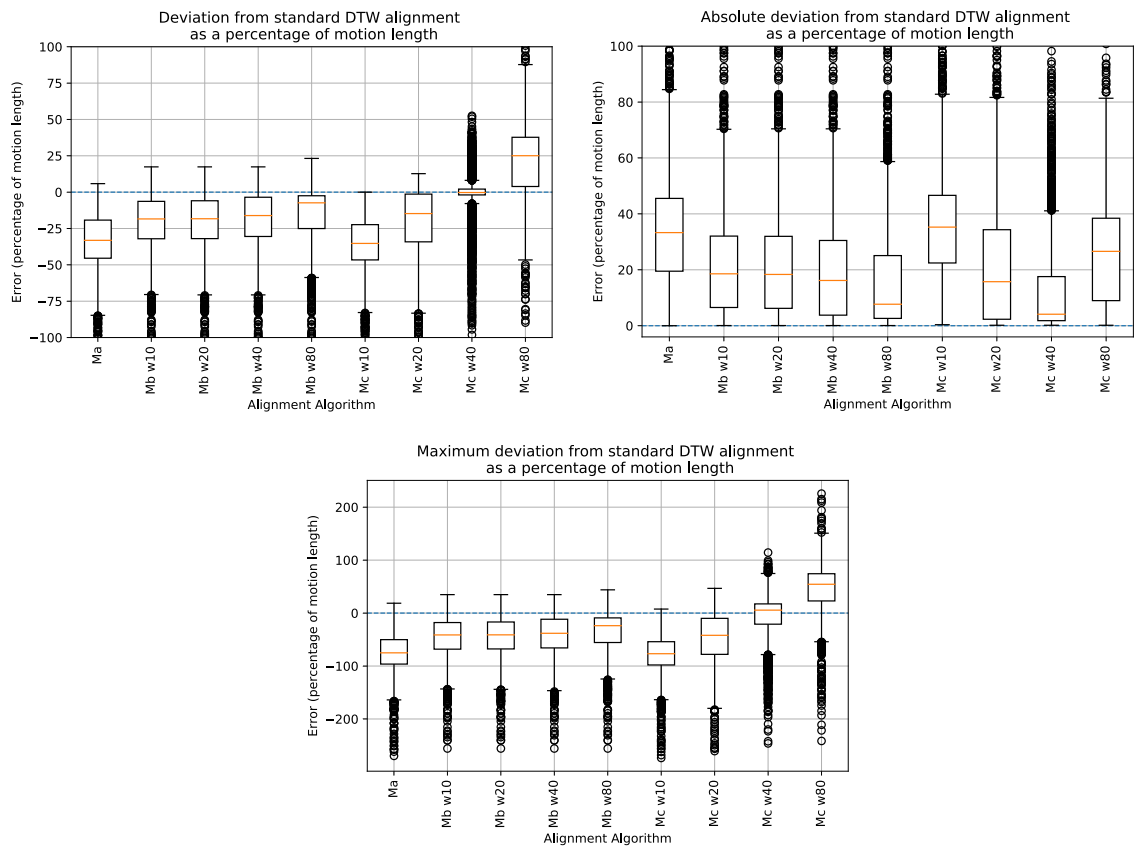
The results obtained from the established offline time warping methods DTW and UTW have also been included to some of the graphs for comparison purposes. As an established and optimal approach to offline time warping, DTW provides an upper performance threshold to compare the on-line time warping algorithms in this study against, demonstrating the best potential performance a time warping algorithm could achieve. UTW shows the alignment achieved by uniformly stretching and squashing the motion with no alignment of individual frames. In this study UTW is used to establish a threshold which on-line time warping algorithms should meet, if they are to be considered effective enough for real world applications. Blue and red lines have been plotted at the median point of the DTW and UTW results respectively as a visual aid to help comparison.

5.4.2 Confidence in Results

A power analysis was performed on the alignment test results, which can be seen in Table 5.2. For any combination of two alignment algorithms tested, the table shows the number of samples required to be confident that a difference between their performance does exist. In line with standard practice, a statistical power of 80% was used, allowing a 20% chance of a type II error, while the significance level



(a) Results of the alignment and distortion performance tests. In both tests the higher result is best. The blue line represents the alignment and distortion achieved by the standard offline DTW algorithm, and the red line represents the alignment achieved using UTW with no alignment.



(b) Results of the accuracy performance tests. Results closest to zero are best.

Figure 5.8: The results of the performance tests. Ma, Mb and Mc refer to methods A, B and C respectively, while w specifies the window size used.

was set to 5%, allowing 5% chance for a type I error.

Table 5.2: A power analysis of the alignment test results in Figure 5.8a, showing the number of samples required to have confidence in a performance difference existing between any two time warping algorithms. The analysis was conducted with a statistical power of 80% and significance level of 5%. Each algorithm was tested with 3248 samples, giving confidence that differences do exist between the performance of almost all the time warping algorithms tested. A small number of outlying pairs of algorithms, require more than 3248 samples, as they have a similar distribution of results in the alignment test.

	Mb w10	Mb w20	Mb w40	Mb w80	Mc w10	Mc w20	Mc w40	Mc w80
Ma	205	131	47	20	523	21	8	35
Mb w10		2388	138	36	78	40	11	89
Mb w20			245	48	61	55	14	143
Mb w40				162	30	209	24	3389
Mb w80					15	9610	63	247
Mc w10						16	7	23
Mc w20							53	344
Mc w40								27

The power analysis was ran on the alignment test results as this test is most pertinent to the potential performance of an algorithm when used in real-world applications. It was considered unnecessary to run the power analysis on all the tests.

As documented previously, the number of samples used on each alignment algorithm (i.e. the sample group size) was 3248. The power analysis shows that significantly fewer samples are required to be confident that differences exist between the performance of most pairs of algorithms. Where pairs of algorithms do require more samples (for example *Mb w80* and *Mc w20*), consideration will be given as to whether these impact on the recommendations and conclusion of this study.

5.4.3 Alignment and Distortion Results

The results of the offline DTW method act as a gold standard exhibiting the best alignment and distortion results that could potentially be achieved, with a median score of 0.349 and 3.621 for the alignment and distortion tests respectively. As anticipated the offline DTW method achieved the best results for both alignment and distortion in Figure 5.8a.

While no on-line warping method is expected to perform better than the DTW

method, an ideal on-line time warping method would perform better than UTW. The results of the alignment test in Figure 5.8a, shows that only Method C with a window size of 40 (*Mc w40*) achieved this with UTW and *Mc w40* achieving median scores of 0.240 and 0.285 respectively in the alignment test. Although all the other approaches failed to beat the UTW method, this should not be considered a disappointment. The UTW method itself, although basic, is also dependent on knowledge of the entire input and target motions, unlike the on-line approaches tested.

As expected, method A (*Ma*) was one of the worst performers in both the alignment and distortion tests, with median scores of 0.069 and 1.298 respectively. However, it set a useful benchmark, providing a straightforward naive approach to evaluate the window based methods, A and C, against. The results of the distortion test matched closely with those of the alignment test, with the poor performances of Method A and Method C with a window size of 10 (*Mc w10*), being more exaggerated in the distortion tests.

There was an interesting contrast in the relationship of window size to performance between Methods B and C. Figure 5.8a shows as the window size increases with Method B, the results of alignment tests, and to a lesser extent the distortion test, improved in a predictable and stable manner. The improvement in the performance of Method B, in alignment test, between each window size, increased as the window sizes being tested increased, with improvements of: 0.002 (2%) between *Mb w10* and *Mb w20*; 0.024 (18%) between *Mb w20* and *Mb w40*; and 0.064 (17%) between *Mb w40* and *Mb w80*.

The improvement in the performance of Method C in relation to window size was less consistent. The alignment tests show a clear peak in performance at a window size of 40, dropping significantly by 0.139 to 0.146 with window size of 80.

As suggested previously in Section 5.3.3, when discussing appropriate window sizes to test within this study, a window size of 80 is less optimal as this covers over half the typical period of a walk cycle, which is between 133 and 146 frames, resulting in multiple local minima points occurring between the incoming target motion frame

and the subset of input frames within the window. This suggests a potential relationship between optimal window size for method C, the sample rate at which the motions are recorded and the frequency of cycles within motion. This issue does not occur with Method B as an alignment path is plotted starting at (n, m) and end at $(0, 0)$. The path plotting combined with the approach used by Method B to selecting the optimal input frame based on the path, prevents large numbers of input frames from being skipped in a single step of the alignment.

Compared with the results for the DTW method, the best performing on-line method, *Mc w40*, exhibited a wider spread of results. This would suggest that this method does not perform as consistently, across different motions as the DTW method. Another indicator of the potentially inconsistent or unpredictable nature of Method C, is how much the performance of the algorithm drops off outside of the optimal window size in comparison to Method B.

In order to consider how each time warping algorithm performs with motions that contain the different types of movement identified in 5.1, the performance of each algorithm in the alignment test was broken down into motions containing a specific types of movement in Table 5.3. Despite concerns about consistency of the *Mc w40* algorithm, these results show it consistently performing the best of all the online alignment algorithms, regardless of the type of movements contained within the motion. The ranked order of the mean alignment performance test results (μ), for motions containing each of the different types of movement are the same for both the *Mc w40* and DTW algorithms. Ordered from the best to worse, the rank motions containing each type of movement are in the following order: Lower Body; Cyclic; Ballistic; Upper Body; and Non-Cyclic.

5.4.4 Accuracy Results

Three accuracy tests were carried out comparing the mapping of input frames to target frames produced by each on-line algorithm with that of the DTW algorithm, treating the map from the DTW algorithm as the optimal alignment. The results of these tests can be seen in Figure 5.8b. The deviations between the maps have been

Table 5.3: The means μ and standard deviations σ of the alignment test scores achieved by each time warping algorithm, for motions containing different types of movement as identified in Table 5.1

Method	Cyclic		Non Cyclic		Ballistic		Lower Body		Upper Body	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Ma	0.103	0.149	0.114	0.157	0.105	0.127	0.108	0.153	0.104	0.147
Mb w10	0.139	0.146	0.134	0.177	0.125	0.124	0.145	0.156	0.133	0.155
Mb w20	0.150	0.161	0.140	0.184	0.137	0.139	0.155	0.169	0.140	0.165
Mb w40	0.187	0.182	0.158	0.196	0.185	0.155	0.191	0.188	0.170	0.184
Mb w80	0.231	0.181	0.195	0.205	0.233	0.147	0.238	0.187	0.211	0.188
Mc w10	0.089	0.144	0.084	0.152	0.112	0.134	0.088	0.147	0.091	0.143
Mc w20	0.225	0.184	0.189	0.186	0.270	0.136	0.228	0.186	0.202	0.180
Mc w40	0.297	0.157	0.249	0.182	0.270	0.122	0.304	0.159	0.262	0.167
Mc w80	0.172	0.172	0.225	0.185	0.107	0.093	0.207	0.176	0.186	0.177
DTW	0.356	0.137	0.320	0.152	0.348	0.098	0.373	0.122	0.329	0.141

normalised and expressed as a percentage of the overall motion length. The closer the median of each sample group is to zero, the better the algorithm has performed.

Again, as expected Method A (*Ma*) is consistently one of the worst performing methods, setting a lower benchmark of a 33.3% error, in the average absolute deviation test, to compare other methods against.

As with the alignment results, the performance of Method B in all the accuracy tests improved consistently as the window size increased, with increasingly larger improvements in performance as the window sizes increased. The performance of Method B in the average absolute deviation tests improved by: 1.6% between *Mb w10* and *Mb w20*; 11.5% between *Mb w20* and *Mb w40*; and 52.5% between *Mb w40* and *Mb w80*.

Method C has also produced some interesting results for the accuracy tests, as it did with the alignment tests. Each of the accuracy tests show Method C working at its most optimal when configured with a window size 40, with a median 4.1% error in the average absolute deviation test. The results of the average deviation tests in Table 5.8b, both show the alignment maps in Method C falling behind the optimal alignment, by 18.4% and 18.2% with a window size of 10 and 20 respectively, but being ahead of the optimal alignment by 25.1% when a window size of 80 is used.

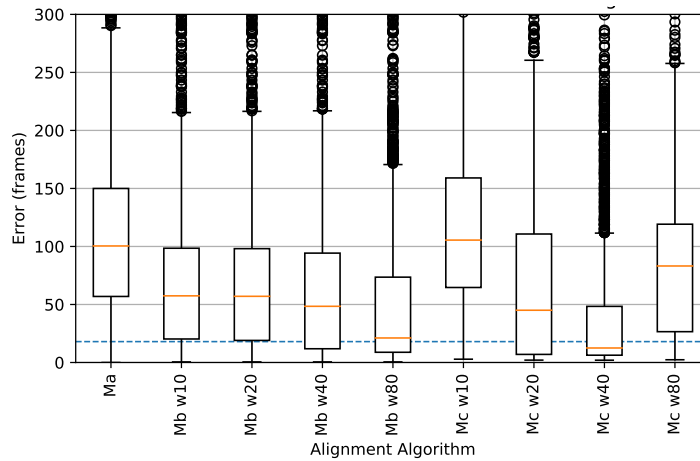


Figure 5.9: Absolute deviation from the standard DTW alignment in frames. The blue line represents the threshold at which timing errors in character interactions can be perceived (Hoyet, McDonnell and O’Sullivan, 2012).

This gives an indication of how the different windows sizes in Method C are effecting the performance of the algorithm.

Each of the tests have outliers with deviations of more then 100%. This is possible when the input motion being aligned is longer than the target motion.

To understand how the accuracy of a time warp relates to human perception, the average threshold for viewers being able to perceive timing errors of 150ms, established by Hoyet, McDonnell and O’Sullivan (2012), was used. As the data-set used in this study was recorded at 120Hz, the number of motion frames that 150ms represents was determined to be 18 using $150/(1000/f)$ where f is the sample frequency. The deviation from the standard DTW alignment shown in Figure 5.8b is normalised, so does not allow direct comparison to the 150ms threshold. Figure 5.9 shows the deviation in frames for the a more direct comparison. A blue line has been added to represent the 150ms perception threshold, so performance of the algorithms can be compared against this. The only algorithm with a median below this was *Mc w40* with a median error of 13 frames, although *Mb w80* does come close with a median error of 21 frames.

5.4.5 Computational Performance

An on-line time-warping algorithm must be able to process frames faster than the rate they are being sampled to avoid dropping frames. Table 5.4 shows the average time each algorithm took to process a single frame in microseconds (μs). The algorithms were ran on a single processing core, with the processor running between 3.58GHz and 3.95GHz during the time warp. Given a sample rate of 120Hz, a process time significantly less than $(1 \times 10^6)/120$ or 8333 μs is required.

Table 5.4: The average time each algorithm takes to process a single frame of motion data in microseconds.

	Window Size (w)			
	10	20	40	80
Method B	856 μs	1654 μs	4806 μs	16945 μs
Method C	782 μs	1583 μs	4764 μs	10106 μs

The results show that a window size of 80 would not be feasible for use with a sample rate of 120Hz. The quadratic relationship between windows size and processing time, that is characteristic of DTW, can be clearly seen in these results. The lower than expected processing time of Method C with a window size of 80 (*Mc w80*), is an anomaly, due to the algorithm running out of input frames to match to, early in the alignment process, therefore requiring any attempt to align to the remaining target frames to be skipped. Ignoring the anomaly of *Mc w80*, although Method C is slightly more computationally efficient than Method B, as it does not need to plot an alignment path, the reduction in computation time is insignificant, only reducing it by 74 μs , 71 μs , and 42 μs for window sizes 10, 20, and 40 respectively.

Potential adaptations could be made to the processing of cost matrices in both methods to allow larger windows sizes such as 80 frames to be viable options. One approach would to determine the initial alignment costs of each cell in the cost matrix in parallel, using a multi threaded process. Another approach would be to down sample the cost matrix as proposed by (Salvador and Chan, 2004).

While motion performances are typically captured at a sample rate of 120Hz, visualisations do not need to be played back or displayed at that rate. Visualisations

could be played back at 60fps or 30fps, with little to no impact, potentially allowing intermittent alignment of capture frames (i.e. aligning every other frame) and interpolation of others to reduce the computational load. In fact on-set previsualizations of virtual characters during visual effects production would need to be time locked with the capture rate of the camera being used, which would typically be 24fps.

5.4.6 Visualising Aggregated Plots

Figure 5.10 visualises the alignment paths plotted by each time warping algorithm, by aggregating the paths plotted for each sample into a single heat-map. This allows the behaviour of different alignment algorithms to be visualised. The alignment paths determined using the standard DTW approach have also been plotted for comparison.

The heat-maps were created by accumulating all the alignment paths M , plotted by a given time warping algorithm into a single 100 x 100 matrix $A_{100,100}$, using equation 5.18 on each alignment point in M of every aligned motion. M^t and M^i are the positions of the alignment point in the target and input motions respectively and n and m are the number of frames in the input and target motion respectively.

The heat-map for Method A shows how the algorithm often sticks on a particular input frame during alignment. These stuck alignments appear as horizontal lines in the heat-map. Rather than the alignment paths clustering around a diagonal line going from (0,0) to (100, 100) as with the DTW alignment, the stuck alignments of Ma create a hot area under the diagonal. Once stuck, algorithm Ma is rarely able to continue moving forward through input frames, much less find its way back onto the alignment path. The issue of getting stuck on an input frame also affects Methods B and C, although to a lesser extent.

$$A_{p,q} = A_{p,q} + 1$$

$$\text{where } p = \frac{M^t}{n} 100 \text{ and } q = \frac{M^i}{m} 100 \tag{5.18}$$

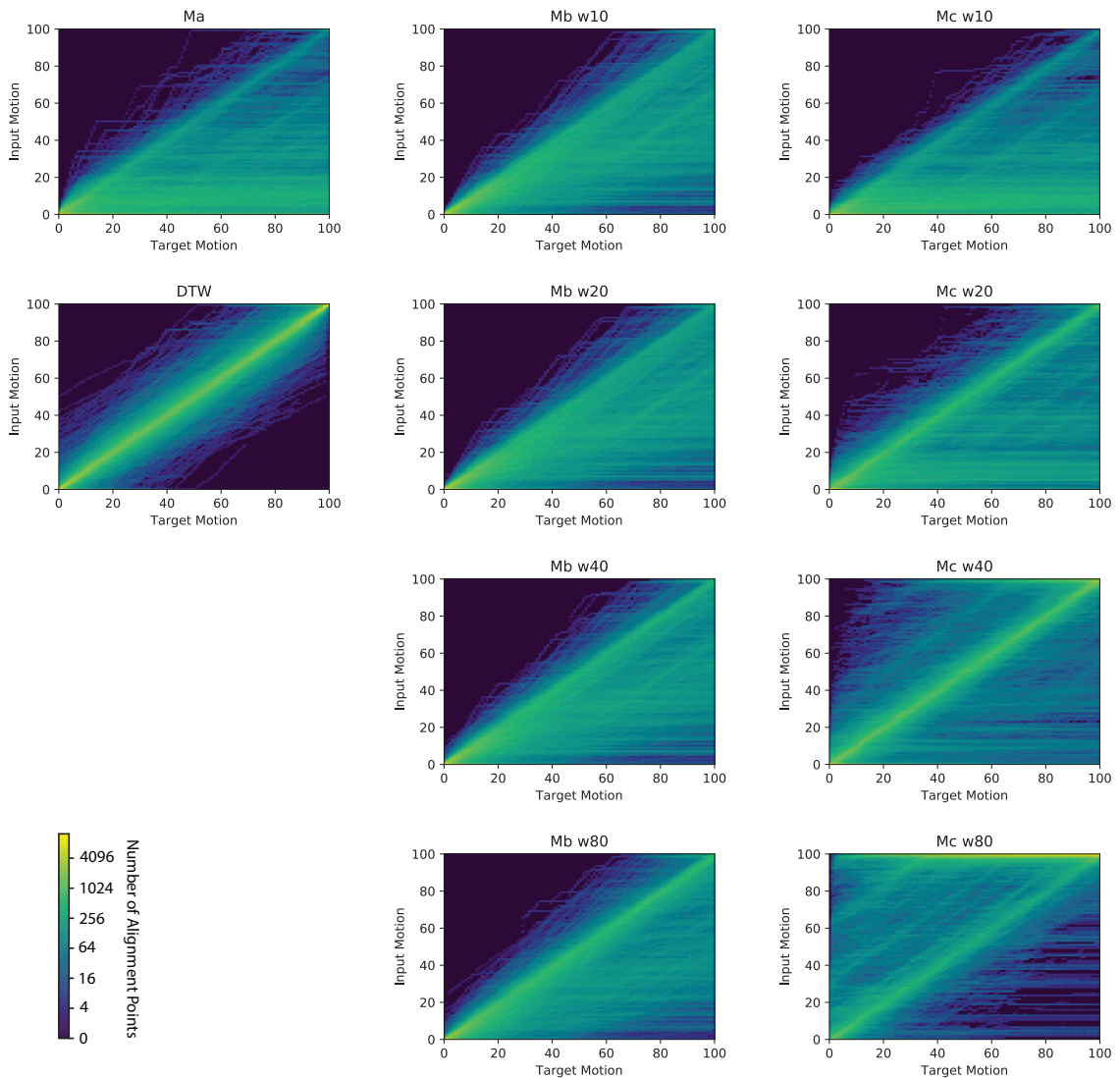


Figure 5.10: Heat maps visualising the alignment paths plotted by each algorithm. Each heat-map shows an aggregate of the alignment paths plotted by all 3248 samples in the data-set for a given time warping algorithm. Every alignment path has been normalised to a size of 100 by 100, each cell in the heat map shows the number of alignment points plotted within that cell. An optimal alignment algorithm will cluster alignment paths around the diagonal as shown with the DTW algorithm. Hot areas under the diagonal indicate alignment paths incorrectly sticking on frames, while hot areas above the diagonal indicate alignment paths incorrectly skipping frames.

The gradual improvement in Method B as the window size increases can also be seen. As the window size increases, the alignment path is less likely to stick or remain stuck on an input frame, this can be seen by the slight reduction in the hot area under the diagonal. The consistently dark area above the diagonal, shows that this improvement does not result in significantly more alignment paths incorrectly skipping input frames.

As the window size increased for Method C, the plots visually show the bias moving from getting stuck on an input frame (hot below the diagonal) to skipping input frames (hot above the diagonal). While window size 40 is clearly the best of the window sizes tested, the hot area at the top of plot *Mc w40* (at input frame 100), shows there are a lot of alignments where too many input frames are being skipped, causing the alignment path to run out of input frames to align to. The spread of alignment paths on this plot also suggests that this algorithm does not work consistently across all samples. The hot area above the diagonal within the *Mc w80* plot, shows many alignment paths incorrectly skipping frames, corroborating the results of average absolute deviation tests in Figure 5.8b.

5.4.7 Visualising Individual Alignments

To gain a deeper understanding of the quality of alignment and resulting motions produced by each time warping algorithm. The individual alignments produced by different online time warping algorithms, for selected pairs of input and target motions were plotted and visualised in more detail in Figure 5.11.

Three pairs of motions were selected that demonstrate issues and characteristics that occur in time warped motions. The top example shows alignments resulting from time warping a **Walk Circle 6 Steps R Start** motion, that contains a cyclic walking motion with movement in both the upper and lower body. The bottom two examples show alignments resulting from time warping **Turn Right** motions, in which the performer turns their whole body in one spot, it contains a non cyclic motion with most of the movement in the lower body. Neither motions contain a any ballistic movement.

Three different forms of visualisation have been produced for each motion pair to visualise different aspects of the alignments created for them:

1. **Alignment Paths:** Shows the alignment paths produced by each on-line time warping algorithm with the offline DTW algorithm included for reference. The alignment paths are overlaid over a heat map of the accumulated cost matrix of the pair of motions used in the alignment. The dark blue area of the heat-map indicates a low cost with minimal difference between the poses input and target frames, the bright yellow area indicates an area of high cost with a substantial difference between the input and target frames. The alignment algorithms' objective is to plot a path through the dark blue, low cost, areas of the matrix, to determine the most cost effective alignment.
2. **Motion Curves:** Plots of the motion data associated with a single joint axis. The purpose of these is not visualise the entire motion data but to visualise the alignment of a single joint axis in more detail, which is particularly pertinent to the motions being aligned.
3. **Views:** Orthographic renders of characters poses from individual frames within the motion, from a specified elevation. The elevations plotted are those most appropriate for displaying the motion being performed.

The colour coding of the different on-line time warping algorithms is same across all of three visualisation. While the light blue in the alignment paths, denotes the offline DTW algorithm, the darker blue in the motion curves and views, refers to the target motion to which the input motion was being aligned to. In both the motion curves and the view, the bottom three aligned inputs are being compared against the target motion at the top.

The sample number next to each example is a unique ID that was given to every pair of motions in the data set. The sample number can be used to look up a corresponding video of the alignment within a GitHub repository ([Randall, 2022b](#)).

In both the views and videos the root translation or rotation of the hips been removed in some cases, to make it easier to compare the motions. An example of this is the

Walk Circle 6 Steps R Start where the character turning obscured a clear view of how well the walk cycle was being aligned.

Incorrectly Sticking on Input Frames

A clear example of an alignment incorrectly sticking on an input frame can be seen in Figure 5.11, within the alignment produced by the red *Mc w20* algorithm in the top example alignment. The alignment path sticks early in the alignment and then proceeds horizontally, mapping the same input frame to the remaining frames of the target motions. The stuck alignment manifests itself as a flat line in the motion curve, as repetition of the same input frame causes a repetition of the same joint orientations. The views show the stuck alignment resulting in the aligned motion freezing early in motion.

Another example of an alignment sticking can be seen with the grey *Mb w40* algorithm at the bottom of Figure 5.11. Rather than sticking on a single input frame for the remainder of the motion, it stick for short periods creating plateaus within the alignment path. These manifest themselves as flat spots in the motion curve and as matching poses in frames 132 and 176. The video shows the alignment pausing in a number of places, including before frame 132, indicating that small stick will not always show up in the views.

Incorrectly Skipping Input Frames

The top of Figure 5.11 also contains an example of an alignment incorrectly skipping, with algorithm *Mc w80*. The alignment skips approximately 132 input frames, between aligning to frames 180 and 182 of the target motion. This causes the aligned motion to skip an entire walk cycle and run out of input frames to align before the alignment is completed. In the alignment path there is a sudden vertical step between frames 180 and 182, as the alignment path crosses a brighter high cost island in the heat map and to an input frame in the following walk cycle, which contains a similar pose to the target frame being aligned. The skip also appears in the respective motion curve which shows only two peaks in the rotation of the hip, when there should be three, one for every two steps in the motion. At frame 180

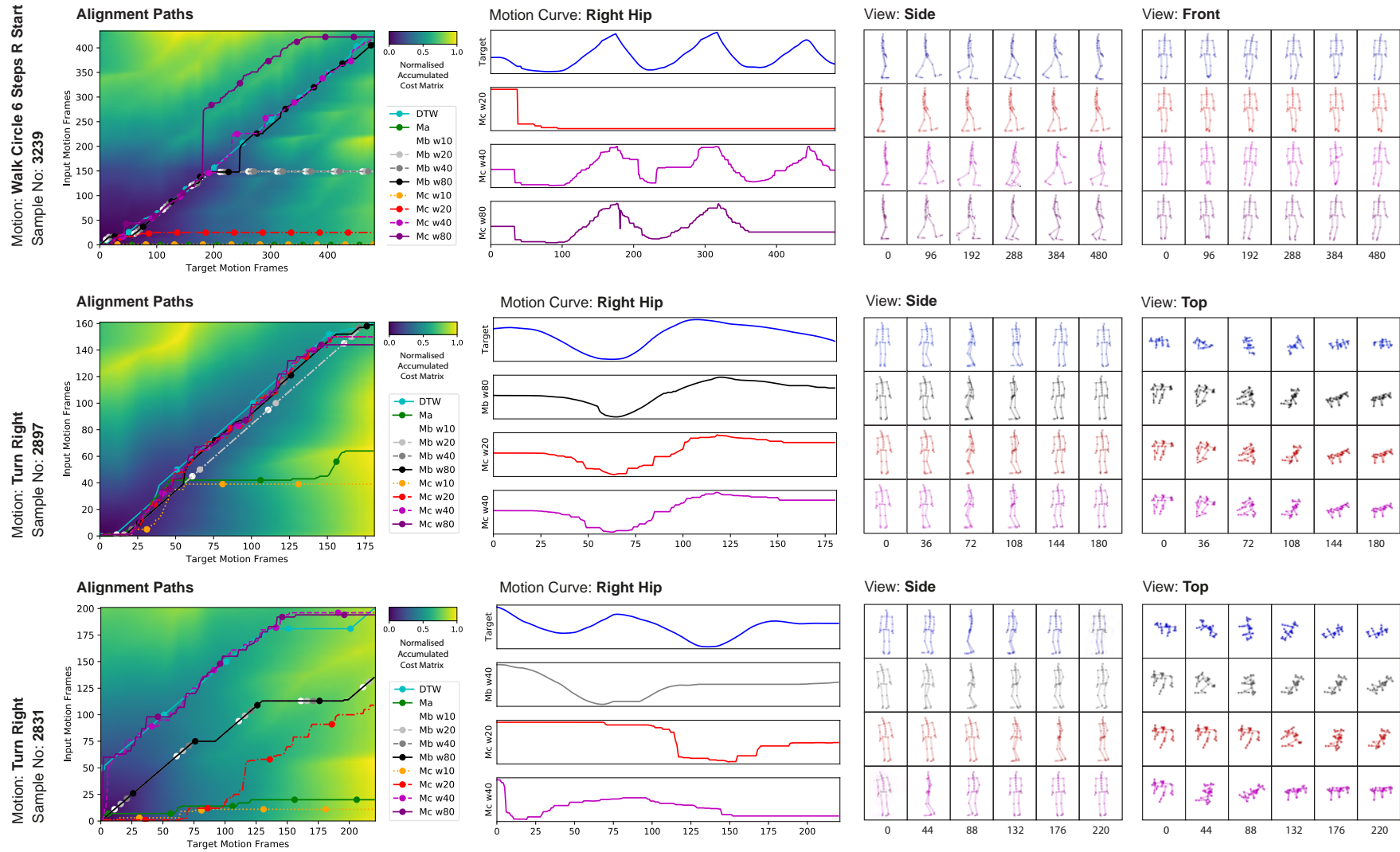


Figure 5.11: A detailed presentation of the alignments produced by online time warping algorithms, showing: the alignment paths plotted by each time warping algorithm, aligned motion curves in comparison to the target motion, and rendered character posed from different elevations.

there is also a glitch in the motion curve where the rotation in the x axis of the right hip, suddenly drops and then comes back up. The glitch is caused by a skip occurring at target frame 180, where 78 frames of the input motion are skipped, almost the maximum number the window the algorithms window size allows, during which the value of x axis drops from 22° to -11° . On the following target frame, 181, another 54 input frames are skipped, during which θ rises back to 20° . When the alignment runs out of frames towards the end of the motion, a flat line occurs in the motion curve and the motion freezes as shown in the video and last two poses of the motion shown in the view. When watching a video of the alignment, the skip in the alignment is not visible, with the first two walk cycles aligning accurately and smoothly. It is only when the motion stops prematurely that it is clear something went wrong.

Jittery Alignments

The plots for Sample number 2897, in the middle of Figure 5.11, show a number of time warps that have produced good alignments, allowing a comparison of the alignment achieved by different time warping algorithms, when they have not been affected by incorrectly skipping or sticking on input frames. While all three algorithms, whose motion curves have been plotted (*Mb w80*, *Mc w20* and *Mc w40*), have produced a good alignment, the motion curves resulting from algorithms based on Method C are jittery, while those resulting from Method B are smooth. These different characteristics between Methods C and B, can also be seen in the bottom example. Algorithm *Mc w40* has produce an accurate alignment that follows the DTW alignment, but a jittery motion curve, while *Mb w40* has produced a less accurate alignment, but with a smother motion curve.

The cause of the jittery motion curves produced by Method C can be seen in alignment paths produce by its respective algorithms, which contain frequent changes in direction with lots of small vertical cliffs and horizontal plateaus. In comparison the alignment paths plotted by algorithms based on Method B, contain only a few infrequent changes of direction. Despite how pronounced these jitters appear in both the alignment paths and motion curves, the video of Sample number 2897 shows

that they are not as pronounced in the playback of the motion, with limited impact on the fidelity of the motion.

Interestingly, the jittery alignments produced by algorithm *Mc w40* do not appear to overtly impacted the results of the algorithms distortion test in Figure 5.8a. However, when the improvement in performance from algorithm *Mb w80*, an algorithm that produced less accurate alignments without jittery alignment paths, to *Mc w40* is compared to across both the alignment and distortion tests in Figure 5.8a, the impact of the jittery alignments produced by algorithm *Mc w40* becomes clearer. While *Mc w40* achieves a 42% improvement over algorithm *Mb w80* in the alignment test, it only achieves a 4% improvement over algorithm *Mb w80* in the distortion test.

Impact of Window Size

In comparison to algorithms based on Method B, the performance and behaviour of algorithms base on Method C varies more significantly between different window sizes. This can be seen in both the aggregate plots in Section 5.4.6 and the alignment paths in Figure 5.11. There is a clear divergence between the alignment paths plotted by algorithms based on Method C, with algorithms using smaller windows consistently sticking on input frames incorrectly, and those using larger windows sizes having a tendency to skip frames. In contrast the algorithms based on Method B often followed similar and often almost identical paths.

5.5 Discussion

5.5.1 Impact of Sample Rate on Window Size

The choice of window size has a considerable impact on how both Methods B and C perform, however, the results showed a clear peak in performance for Method C at window size 40 dropping significantly with a larger window size of 80. The purpose of this section is to determine if the optimal window size for Method C relates to frequencies of reoccurring cycles within movement of the motions being aligned.

The occurrence periodic cycles, such a walk cycles, in motion data was discussed in Section 5.3.3, when considering which size of forecasting windows to test within this study. In that section, it was anticipated that a window size with a duration of more than half a walk cycle (73 frames) would not be optimal. The more constrained approach of Method B, which has to plot an alignment path through the forecast window, appears to prevent this issue from occurring, maintain performance improvements at larger window sizes, with *Mb w80* performing the best of the Method B algorithms tested. However, with its less constrained approach, this issue did occur with Method C, with a significant drop in performance between algorithms *Mc w40* and *Mc w80*.

Any potential relationship between the frequency of movement cycles and the optimal window size, will also have implications on the relationship between the sample frequency at which motions are recorded and the optimal window size. To explore this, the data-set was down sampled to 60Hz, effectively halving the number of frames over which any motion cycles occurred within the data-set. Method C was applied to this data-set using the same previously used set of windows sizes. The performance of these algorithms on the down sampled data-set can be seen in Figure 5.12.

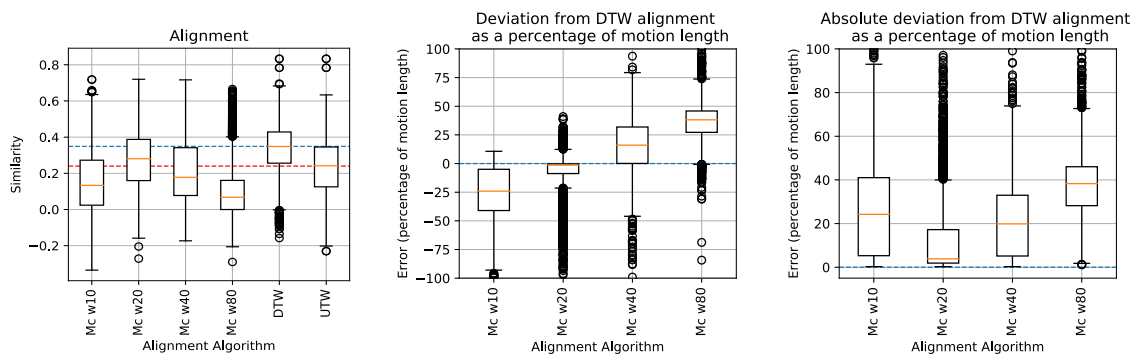


Figure 5.12: Performance results for Method C when applied to motions sampled at 60Hz.

As anticipated the optimal window size has shifted downwards, along with the sample rate, from 40 to 20. Both the alignment and accuracy test results show the same characteristics as the test results from the data-set samples at 120Hz, but shifted down a window size. The window size of 80 now represents more than an entire motion cycle and perform worse in the absolute deviation accuracy test than the

Ma algorithm.

5.5.2 Joint Weighting

Within this study, a subset of joints considered most pertinent to everyday motions was used, consisting of both left and right: shoulders, elbows, hips and knees. Each of these joints were equally weighted when calculating the similarity of two motions or calculating the cost or distance between the character poses of two given frames within the alignment algorithm. The joints were equally weighted regardless of which joints might be most pertinent to a particular movement, for example upper body joints were not given more weighting in motions that make particular use of those joints.

Although there is no knowledge of the incoming target motion, an analysis of the pre-recorded input motion could be performed to automatically determine the optimal weighting of each joint. [Patrona et al. \(2018\)](#) proposes an approach to automatically weighting joints based on the amount each joint moves from a given neutral position. Although implementing automatic joint weighting is outside of the scope of this study, it could potentially further optimise the performance of an online time warping algorithm.

Another approach to joint weighing is the use of dimensional reduction techniques such as PCA analysis. [Johnson \(2003\)](#) proposes a technique for representing the orientations of all the joints of a character in single multidimensional PCA space. The aim was to reduce the amount of data required to store a character's pose by storing only the eigenvectors within the PCA space with the largest eigenvalue, as these are the principle components of the pose. This same approach could be used as a pseudo automated joint weighting approach by basing the distance calculations on the principle eigenvectors in PCA space. This approach may also reduce computation time, however, a distance metric based on distance in PCA space could have implications on accuracy which would need to be explored.

5.5.3 Jittery Alignments

In section 5.4.7, it was found that time warping algorithms based on Method C produced jittery alignments, with frequent changes of direction in the alignment paths and jagged motion curves containing lots of small cliffs and plateaus. This is in contrast to the smoother alignments produced by time warping algorithms based on Method B.

There is a need establish if the jitter in the Method C alignments is perceptible or acceptable. To support this a method a measuring the magnitude of jitter is required, along with human perception tests to establish that magnitude of jitter which is perceptible or acceptable for a given task.

The magnitude of the jitter could be objectively evaluated by measuring the size of the cliffs and plateaus in the alignment paths, an approach that is quite specific to the style of jitter that Method C produces. Alternatively a more general approach would be to apply a smoothing algorithm to the alignment path, then used SNR to evaluate the difference between the smoothed and none-smoothed alignment paths.

It is important to consider that any testing of what is an acceptable magnitude of jitter, would be dependent on the task being performed. For example, certain tasks such as real-time transfer of motion styles, maybe a lot more sensitive to jittery alignments than visualisations motion at low frame rates.

5.5.4 Optimizing Alignment Accuracy

Constraints

Time warping techniques such as DTW are often implemented with constraints. However, as the aim of this study was to measure the raw performance of each method, no constraints have been applied to any of the methods used in this study, beyond the number of input frames in the search window.

Constraints have the potential to reduce the problem of algorithms skipping too many input frames or sticking on an input frame. [Rabiner and Juang \(1993\)](#) propose a local inter-frame constraint method, in which the options available for an alignment

path to take on a given frame are dependent on the path taken in the previous frames. For example, if the last two target frames have been mapped to the same input frame, then the next target frame cannot be mapped to the same input frame, therefore preventing the alignment path from sticking on the same input frame for too many consecutive frames.

Smoothing the Motion Data

A number of techniques could be explored to potentially improve the accuracy of the alignment paths determined by the on-line time warping algorithms tested in this study.

Noise and jitter are by-products of the inaccuracies of motion capture process, resulting in small frequent errors occurring in the motion capture data. For this reason it is often desirable to apply a smoothing filter to remove noise from the motion data, although no smoothing filters were applied to the motions used in this study. Smoothing could reduce the potential for an error in the motion capture data to cause an error in the alignment. Although Figure 5.4 shows using two consecutive frames, is the most accurate approach to using dead reckoning to predict target frames, using more frames could potentially smooth out errors in the target motion.

Another approach is to use a low pass filter, such as a Butterworth filter which is commonly used on motion data. Yu et al. (1999) used regression techniques to determine Equation 5.19 which provides the optimal cut off frequency f_c to use when applying a Butterworth filter to data sample at frequency f_s . In the case of the data-set used in this study which was sampled at 120Hz, this gives an optimal cut off frequency of 8.088Hz.

$$f_c = 0.071f_s - 0.00003f_s^2 \quad (5.19)$$

Pre-Analysis of Motion

While the target motion is unknown, the pre-recorded input motion could still be analysed before the alignment process to optimise the configuration of the alignment

algorithm. For example an analysis of movement frequencies within the motion would allow a more optimal window size to be chosen.

Blended Approaches

There is an interesting contrast between the characteristics of Methods B and C. Method B is more likely to incorrectly stick on an input frame, while Method C is more likely to incorrectly skip input frames. Selecting an input frame based on a weighted blend of the input frames selected by the each of the two methods may be an optimal approach. This approach would be computationally more expensive, but they would both make use of the same initial cost matrix, which is the most computationally expensive part of the process, and the subsequent calculations of the accumulated cost matrix for each method could be performed on different processing threads.

Algorithms could be blending or switched automatically in response to detecting frame sticking or skipping in the time warp. For example switching from a Method B algorithm to a Method C algorithm when frame sticking is detected.

5.5.5 Optimizing Computational Performance

The algorithms in this study were all ran as a single threaded operation. There is significant potential to use a multi-threaded approach to reduce the time required to process each frame of motion. A significant amount of the processing time is spent calculating the initial cost matrix, for which the processing time increases quadratically, not linearly, as the window size increases. The cost matrix, however, easily lends itself to multi-threading, each cell of the matrix can be calculated independently. Unfortunately this is not the case when calculating the accumulated cost matrix or plotting a path through the accumulated cost matrix, as a programmatic approach is taken in which each step is dependent on the previous step.

A common optimisation applied to DTW is to avoid calculating the entire cost matrix and ignore the area around extreme corners of $(0, m)$ and $(n, 0)$ of the matrix which an alignment path rarely needs to go through. [Sakoe and Chiba \(1978\)](#) and

Rabiner and Juang (1993) both propose different approaches for determining the area of the cost matrix to be calculated, which are discussed in more detail in the next chapter, however the aggregate paths plotted by the DTW algorithm in Figure 5.10 clearly indicate which areas need to be calculated. The dark blue areas around (0, 100) and (100, 0) are not used by any of the alignment paths so do not need to be calculated, however, the area which all the paths are plotted through going diagonally across the center of the matrix, does need to be calculated.

5.5.6 Frame Dropping

Although this is a study into on-line time warping algorithms, a deliberate decision was made not to consider dropped frames when testing the algorithms, to ensure the tests were consistent.

In the case of this study dropped frames occur when the algorithm takes longer to process a single frame than the time available between incoming target frames. A sample rate of 120Hz allows 8.333ms between frames, to process each frame.

When the frames are dropped there is still a requirement to map the dropped target frame to a frame in the in the input motion. The path plotting approach of method B provides a graceful way of dealing with this. Any dropped target frames can be appended in front of the current incoming and predicted target frames, resulting in the alignment path also being plotted through the dropped frames, selecting appropriate input frames for them.

As Method C does not plot a path through the accumulated cost matrix, the options for dealing with dropped frames are more clumsy. The input frame to which the dropped frame is mapped to could be interpolated based on the input frame the current incoming target frame was matched to. Another approach is to again append the dropped frames to the front of the target frames, then use the first n columns of an accumulated cost matrix to select input frames for n dropped frames.

5.5.7 Performance Tests

Most of the performance tests used in this study produced useful and contrasting views of the performance of the online alignment algorithms. Some tests, however, did establish themselves to be more useful than others. The core performance tests that should be used in this type of study are the alignment and the average difference and absolute average difference accuracy tests. The alignment tests provide a fundamental measure of the quality of the alignment solution provided by each algorithm, while the two accuracy tests help to develop insight into how the algorithm behaves. The average difference test was particularly useful showing the bias each algorithm has towards skipping input frames or sticking on input frames.

The distortion and maximum deviation tests didn't appear to inform the findings of this study as much as other tests. Although the distortion test largely corroborated the results of the alignment tests and provided little additional insight themselves, it is useful to have a secondary test method to confirm the results of the alignment test. The maximum deviation test felt unnecessary and provided little information beyond what was provided by the average difference test.

5.6 Conclusions

In this chapter two novel approaches to on-line time warping of human motion, methods B and C, were tested and compared to offline DTW and UTW approaches and a naive on-line approach, Method A, using a variety of performance tests.

The results of all the performance tests showed Method C with a window size of 40 to be the best performing algorithm, when working with motions captured at 120Hz. The window size needs to be appropriately adjusted, when working with motion capture data sampled at different frequencies, for example a window size of 20 should be used with a sample rate of 60Hz. Despite concerns about how consistently the *Mc w40* algorithm would perform across motions containing different types of movement, Table 5.3 showed it was the best performing online algorithm regardless of the types of movement contained in the algorithm.

Mc w40 was the only algorithm in which the majority of the alignments produced were close enough to the alignments produced using standard DTW that the difference would not be humanly perceptible. It was also the only algorithm to perform better than UTW. As the only algorithm to pass these two thresholds it is the only one that could be recommended for use in real world applications.

As expected the naive approach, Method A, performed worst, with alignments often sticking on input frames.

Method B has the potential to perform well, but needs large search windows (size 80) to work well, creating significant computational work for real-time applications as shown in Table 5.4. At lower size search windows, Method B would often incorrectly stick on an input frame and did not cope well with delays to the start of the target motion.

Method C performed best and showed optimal performance at smaller window sizes (40), making it ideal for real-time applications. On average the algorithm was shown to perform better than an offline UTW alignment, and able to align to within 150ms of the gold standard offline DTW solution. This demonstrates that the algorithm can produce high quality alignments required for real-world applications.

A relationship was established between the optimal window size for Method C, the sample rate used to record the motion, and the frequency of cyclic movements within the motion. For the data-set used in this study, which contained a variety of cyclic and non-cyclic motions with walking making up the predominant cyclic movement, window sizes of 40 and 20 were found to be the optimal size to use with sample rates of 120Hz and 60Hz respectively. In each case the duration of the window equates to 333ms, suggesting that Method C would struggle to handle misalignment's greater than this.

A number of ideas have been presented for further optimising the algorithm, the most promising and established of which is applying constraints to the alignment path being plotted and penalties for warping a motion (i.e skipping or sticking on an input frame). These approaches will be implemented and tested in the next chapter.

Chapter 6

Constraints and Penalties

6.1 Introduction

A number of different time warping algorithms were implemented within the study presented in Chapter 5 and their performance evaluated. The algorithms in the study were implemented without the incorporation of any optimisation techniques, to allow a direct comparison. Within the study, one algorithm performed well enough to be used for real world applications. However, the algorithm appeared to be sensitive to ordinary errors in the motion data, such as starting points of the motion being misaligned or lack of joint weightings to reflect the joints which are most pertinent to a particular motion. There is, therefore, a need to explore some of the optimisation techniques outlined within the discussion in the previous chapter.

The motivation of the study in this chapter is to:

1. Further optimise the accuracy of the best performing algorithm.
2. Determine if the performance of other algorithms can be improved to a point where they could also be considered for use in real world applications. To support this motivation, it is necessary to test the impact of optimisation techniques on a selection of the time warping algorithms presented in the previous chapter, not just the best performing one.

There are three established techniques for optimising time warping algorithms: constraints, penalties and weighted joints. This study will focus on constraints and penalties, as these target the issues identified in the previous chapter, and there is more potential for the different algorithms to respond quite differently to these techniques.

In this chapter, established approaches to implementing constraints and penalties are adapted to fit characteristics and requirements of plotting an alignment forward in real-time. To support this, in contrast to previous work, penalties will be implemented to constrain alignment paths and sets of constraints paths will start at a single point, rather than converge at a single point. Additionally an approach to implementing constraints using a ‘constraint state table’ is proposed, as a generalised approach to maintaining the continuity of constraints, when sequentially aligning frames.

The performance of different on-line time warping algorithms, with either constraints or penalties incorporated, will be tested using the same techniques and data-set used in the previous chapter. This will allow:

1. A direct comparison of the algorithms’ performance, with and without these optimisation techniques applied.
2. An independent evaluation of the impact of each technique.

6.2 Background

6.2.1 Constraints

In addition to the monotonic and boundary constraints, which a time warping algorithm must satisfy, additional constraints can also be imposed to reduce the number of possible alignment paths that can be plotted. Two commonly used additional constraints are:

- Global path constraints, which define the areas of the cost matrix that an

alignment path must stay within, and are used to reduce computational requirements of the time warp.

- Local continuity constraints, which define the local step sequences allowed with an alignment path, and prevent a time series or motion from being overly distorted by excessive expansion or compression.

Global Path Constraints

Two established approaches to defining global path constraints are the Sakoe-Chuba Band (Sakoe and Chiba, 1978) and the Itakura Parallelogram (Itakura, 1975), shown in Figure 3.3. By constraining the area of the cost matrix within which an alignment path is searched, they reduce the computation required to solve a time warp from being a quadratic function of the length of the motions to towards a linear function.

The boundaries of both global path constraints are defined using slope equations as seen in Figure 6.1. The Sakoe-Chuba band constraint can be simplified to the condition defined in Equation 6.1. For a point to be within Itakura Parallelogram it must satisfy the conditions in both Equations 6.2 and 6.3, where q determines the size of the parallelogram.

$$|y - x| \leq r \tag{6.1}$$

$$\frac{x - 1}{q} + 1 \leq y \leq \frac{x - m}{q} + n \tag{6.2}$$

$$q(x - m) + n \leq y \leq q(x - 1) + 1 \tag{6.3}$$

Local Continuity Constraints

It is important for temporal alignment to be achieved without excessively distorting the motion, resulting in an unnatural looking motion, or excessive skipping of input

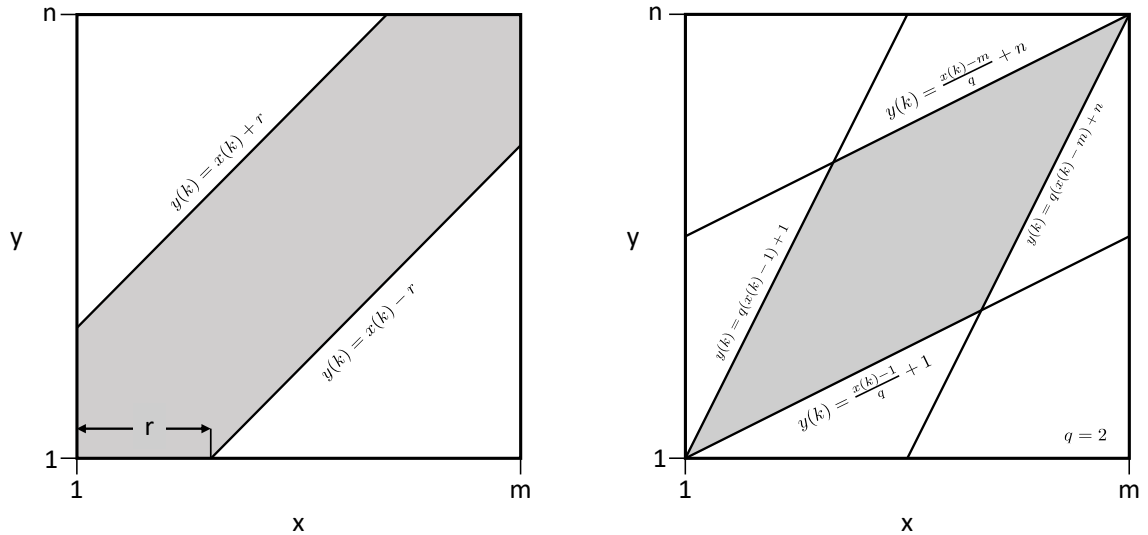


Figure 6.1: Equations for defining slopes to determine the boundaries of global path constraints. Sakoe-Chuba Band (left) and the Itakura Parallelogram (right). In the example Itakura Parallelogram on the right $q = 2$.

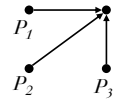
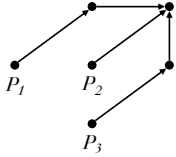
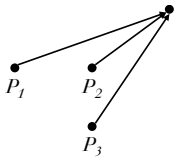
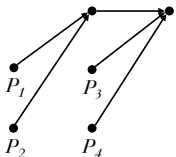
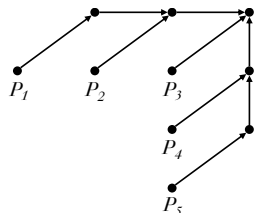
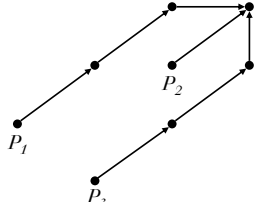
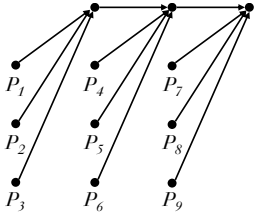
frames resulting in loss of information. Local continuity constraints can be incorporated to restrict the alignment paths that can be plotted to a set number of steps (Rabiner and Juang, 1993). A variety of different rules can be used to express local continuity constraints, such as the constraints proposed by Sakoe and Chiba (1978) in Equations 6.4 and 6.5.

$$x_{k+1} - x_k \leq 1 \tag{6.4}$$

$$y_{k+1} - y_k \leq 1 \tag{6.5}$$

Myers, Rabiner and Rosenberg (1980) proposed a notation system that allowed a wide variety of local continuity constraints to be defined more conveniently, in terms of incremental paths. In Table 6.1, different constraints are defined in terms of a number of allowable incremental paths $\{P_1 \dots P_t\}$, where each path specifies a sequence of moves. When incorporated into a time warping algorithm, a time alignment must be satisfied using only paths specified within the constraint. Note that the Type I constraint in Table 6.1 replicates the condition specified in Equations

Table 6.1: Types of local constraints, expressed as sets of allowable paths $\{P_1 \dots P_t\}$

Type	Allowable Paths	Path Specifications
I		$P_1 \rightarrow (1, 0)$ $P_2 \rightarrow (1, 1)$ $P_3 \rightarrow (0, 1)$
II		$P_1 \rightarrow (1, 1)(1, 0)$ $P_2 \rightarrow (1, 1)$ $P_3 \rightarrow (1, 1)(0, 1)$
III		$P_1 \rightarrow (2, 1)$ $P_2 \rightarrow (1, 1)$ $P_3 \rightarrow (1, 2)$
IV		$P_1 \rightarrow (1, 1)(1, 0)$ $P_2 \rightarrow (1, 2)(1, 0)$ $P_3 \rightarrow (1, 1)$ $P_4 \rightarrow (1, 2)$
V		$P_1 \rightarrow (1, 1)(1, 0)(1, 0)$ $P_2 \rightarrow (1, 1)(1, 0)$ $P_3 \rightarrow (1, 1)$ $P_4 \rightarrow (1, 1)(0, 1)$ $P_5 \rightarrow (1, 1)(0, 1)(0, 1)$
VI		$P_1 \rightarrow (1, 1)(1, 1)(1, 0)$ $P_2 \rightarrow (1, 1)$ $P_3 \rightarrow (1, 1)(1, 1)(0, 1)$
VII		$P_1 \rightarrow (1, 1)(1, 0)(1, 0)$ $P_2 \rightarrow (1, 2)(1, 0)(1, 0)$ $P_3 \rightarrow (1, 3)(1, 0)(1, 0)$ $P_4 \rightarrow (1, 1)(1, 0)$ $P_5 \rightarrow (1, 2)(1, 0)$ $P_6 \rightarrow (1, 3)(1, 0)$ $P_7 \rightarrow (1, 1)$ $P_8 \rightarrow (1, 2)$ $P_9 \rightarrow (1, 3)$

6.4 and 6.5.

The implementation of a local constraint also implies a global constraint, as the allowable incremental paths in set P , can only reach certain areas within the matrix. The reachable area of a constraint can be modelled as an Itakura Parallelogram, using Equation 6.6 to determine the q value from a given set of paths P (Rabiner and Juang, 1993). If path P is made up of points $\{P_1 \dots P_n\}$, each containing coordinates (i, j) , then $l(P)$ is the function applied to each path in which the sum of the j co-ordinates in a path is divided by the sum of the i coordinates. The q value for a constraint is determined by the path P which returns the largest value from the $l(P)$ function. Table 6.2 shows the q value for each of the different type of constraint.

$$q = \max l(P) \left\{ \frac{\sum_{k=1}^{P_t} P_{k,i}}{\sum_{k=1}^{P_t} P_{k,j}} \right\} \quad (6.6)$$

Table 6.2: The q values for different types of constraint. The local constraints result in a search area resembling an Itakura Parallelogram, the smaller the q value the more constrained the alignment and the smaller the search area.

Constraint Type	q Value
I	∞
II	2
III	2
IV	2
V	3
VI	1.5
VII	3

The impact of various motions on the performance of different constraint types cannot be easily analysed, therefore, the impact of constraint types needs to be explored experimentally.

Macrae and Dixon (2010) tested the impact of each type of constraint on the alignment accuracy of an application that aligned musical scores, using a novel windowed time warping approach. The tests found that the Type V constraint performed the

best across a range of window sizes, therefore this was chosen as the local constraint method to implement within this study.

6.2.2 Penalties

Penalties are often incorporated into time warping algorithms to mitigate against biases or some undesirable characteristic within the time warping algorithm or the data being aligned. The two main categories of penalties are:

- Feature based penalties, which penalise or weight alignment paths based on features in the data being aligned.
- Path based penalties, which penalise an alignment path, based on the route it is taking through the cost matrix.

Feature Based Penalties

Feature based penalties are effective for dealing with features and trends in the data being aligned, that can effect the performance of a time warping algorithm. These issues typically stem from the distance based metric used to measure the cost or similarity of points m and n in the input and reference data respectively.

[Jeong, Jeong and Omitaomu \(2011\)](#) proposed a time warping algorithm called WDTW (Weighted Dynamic Time Warping), which applied an additional weighting to the standard distance metric used in DTW, based on how far apart the phases of the two points are. Points that are closely in phase had a small penalty applied, while points which were more out of phase had a higher penalty applied. The motivation for this approach was to reduce alignment errors in cyclic data, preventing alignment from getting confused by points before and after a peak or trough in the signal, which would be very similar distances apart.

Another approach is to preprocess data before the time warp, typically to normalise a feature in the data which may effect or bias a time warping algorithm. This approach has been used to automatically weight joints which are more pertinent to a gesture or movement being aligned ([Arici et al., 2014](#); [Celebi et al., 2013](#)). The

contribution of each joint to a gesture is quantified based on the total movement of joints in an example motion. [Choi, Cho and Kim \(2015\)](#) extended this with an approach that dynamically weights the contribution of joints over time, to reflect the importance of different joints to a motion or gesture at different points in time.

Path Base Penalties

When using time warping to recognise, match or classify a motion, DTW has a bias towards matches with the shortest path, resulting in the shorter motions being matched to a query motion. To tackle this problem, approaches have been proposed to normalise for different path lengths. [Hülsmann et al. \(2017\)](#) and [Muscarillo, Gravier and Bimbot \(2009\)](#) proposed a local normalisation approach in which the lengths of the paths used to determine the values in the accumulated cost matrix D , are stored in a separate matrix L . Alignment paths are then plotted through the matrix based on a weighted value $D(m,n)/L(m,n)$.

[Tormene et al. \(2009\)](#) proposed a global normalisation approach based on the length of the alignment path. This was to facilitate open ended time warps, such as OE-DTW, that allow an entire input (or query) sequence to be aligned to part of a reference sequence. As this approach removes the boundary constraint that exists in standard DTW, which forces the alignment path to end at (m,n) , a normalisation approach was required to stop the warping algorithm from always picking shorter alignment paths, which prefers to match an input to fewer frames in the reference motion.

[Anguera and Ferrarons \(2013\)](#) compared both local and global approaches to normalising alignment paths, to align speech within a speech detection algorithm. Although not conclusive it found that local normalization of path lengths performed better for that use case.

[Sakoe and Chiba \(1978\)](#) and [Dixon \(2005\)](#) both suggest penalizing one to one steps in the alignment path (i.e. steps which move one frame in both input and target sequence), by using a weighting of two for this step. The function for determining the accumulated cost matrix D can be adapted to accommodate this using Equation

6.7.

$$D(m, n) = \min \left\{ \begin{array}{l} D(m, n - 1) + C(m, n) \\ D(m - 1, n) + C(m, n) \\ D(m - 1, n - 1) + 2C(m, n) \end{array} \right\} \quad (6.7)$$

6.3 Constraints Methodology

6.3.1 Adapting Constraints to Support Forward Plotting

As discussed previously there are a variety of continuity constraints that can be applied to a time warp, of which the Type V constraint has been shown to be an optimal choice for real-time scenarios (Macrae and Dixon, 2010). These established constraint types, however, are designed for use when plotting an alignment backwards and need to be adapted to support forwards plotting.

A Type V constraint, as shown on the left of Figure 6.2, is implemented by determining which alignment point out of $\{P_1 \dots P_5\}$ has the least cost, then adding the corresponding steps to the front of the alignment path, this process is repeated, plotting the alignment path backwards until the start of both motions is reached. To implementing the constraint within an on-line forwards plotting time warp, adaptations need to be made to accommodate the following issues: i) alignment points need to be chosen from predicted points in the future, rather than points in the past; ii) only one input frame can be mapped to each target frame; iii) the frame by frame nature of forward plotting, in many cases, means the optimal choice of alignment point (out of $\{P_1 \dots P_5\}$) cannot be chosen in a single step, but instead needs to be determined over a number of frames.

To support forward plotting and allow alignment points to be picked from predicted points in the future, the paths allowed by the constraint must be inverted. Figure 6.2 shows how the paths of Type V constraint were inverted. The order of steps associated with each spot have been reversed, but the steps themselves remain the same.

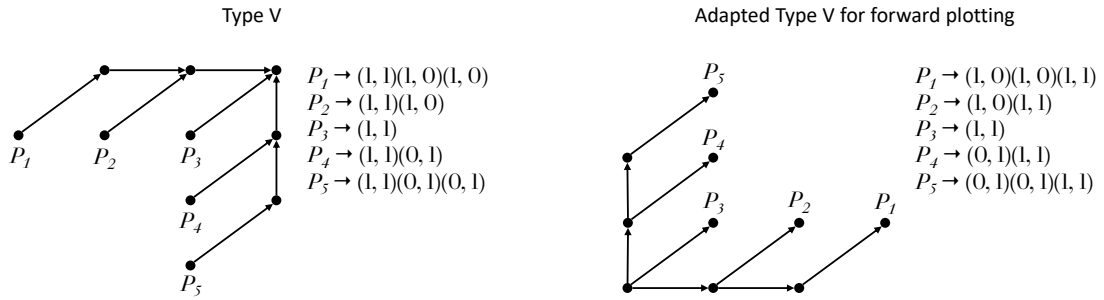


Figure 6.2: Two continuity constraints. On the left is a Type V continuity constraint used when plotting an alignment backwards backwards. On the right is an adapted version of the Type V constraint for plotting an alignment path forwards.

As only one input frame can be mapped to a target frame the constraint steps have to be further adapted as shown in the left of figure 6.3, with points P_5 and P_4 reached in one step. Note that the new incoming target frame automatically forces a step forward in the target motion sequence.

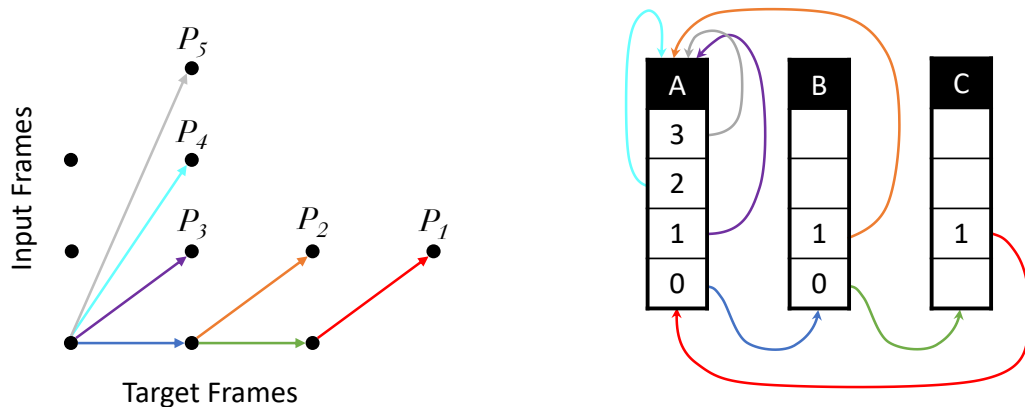


Figure 6.3: The constraint steps used to implement the adapted Type V constraint. The coloured arrows show how each end position in the constraint can be reached.

Unlike points $\{P_3, P_4, P_5\}$, points P_1 and P_2 cannot be reached within a single new target frame. In these cases an alignment point cannot be chosen in a single step, so additional constraints will need to be applied to the preceding step to maintain the constraint. The wider implication is that at any given point in the alignment, the steps available (i.e. input frames which can be mapped to) are dependent on steps performed in the preceding frames. For example if the two preceding steps were both (1,0) (i.e. repeat the same input frame) then the only step possible is (1,1) (step forward 1 input frame).

A novel approach called a constraint state table was implemented, to manage the constraint and maintain its continuity between target frames. The adapted Type V constraint was represented using 3 states (A, B and C) as shown in the right of Figure 6.3. Each state determines which input frames from $I_{M_{n+k}}$ where $k \in \{0..3\}$ can be mapped to the current target frame T_{n+1} . The alignment starts with the constraint in state A, with the current constraint state and selected input frame determining the what state the constraint will be in when processing the next target frame. The coloured arrows in Figure 6.3 show how each end position in the constraint can be reached by stepping through constraint states. Whenever an end position is reached the constraint is set back to state A. The following steps provide an example of constraint states in operation:

1. A time warp begins with the constraint in state A. The time warping algorithm can select to move the alignment of the input motion forward by any number of frames specified in state A, i.e. frames $\{0, 1, 2, 3\}$.
2. The time warp algorithm elects to move the input motion alignment forward zero frames, causing the incoming target frame to mapped to the same input frame as the previous target frame, and the constraint to be set to state B. The time warping algorithm can now only move the alignment of the input motion forward by the number of frames specified in state B, i.e. frames $\{0, 1\}$.
3. For the next target frame the time warping algorithm again elects move the input motion alignment forward zero frames, causing the incoming target frame to mapped once again to the same input frame as the previous target frame. This time the constraint moves to state C which only allows the alignment of the input motion to be moved forward one frame.
4. For the next target frame the time warping algorithm skips any attempt to evaluate alignment options, as it is forced to move the input motion alignment forward one frame. Now the local alignment has reached end point P_1 in the constraint, as shown in Figure 6.3, therefore the constraint resets back to state A.

These states can easily be expressed as a table, as shown in Table 6.3, expressing the constraint logic in a generalised manner that is flexible enough to express other constraint types.

Table 6.3: Constraint state table for the adapted Type V constraint.

		States		
		A	B	C
3	A			
2	A			
1	A	A	A	A
0	B	C		

This approach to adapting local continuity constraints to forward plotting in real-time, does come with a compromise. When implemented in the standard manner, the accumulated cost of each path allowed by the constraint cannot be fully evaluated. The adapted constraints only allows the next step in the path to be evaluated. Using the Type V constraint as an example. If an alignment is in state A, the accumulated cost for paths $\{P_3, P_4, P_5\}$ can be fully evaluated, but only the next step towards paths $\{P_1, P_2\}$ can be considered, as there is no knowledge of the frames these paths go through, as these are outside of the known part of the incoming target motion.

6.3.2 Implementing Constraints

The constraint state table, shown in Table 6.3, was used to implement the adapted Type V constraint within each of the three on-line warping approaches tested in previous chapter, methods A, B and C. For each method the following segments of code were used to implement the constraint:

1. At the start of the alignment the constraint state is set to A.
2. Before processing each frame the constraint table is checked. If the current constraint state has only one possible choice of input frame (i.e. state C), the entire process of determining the optimal input frame is skipped, and the target frame is mapped to the input frame corresponding to the only option available.
3. If the current constraint state provides a selection of input frames to choose

from, the process for determining the optimal input frame is constrained to choose only from that selection. This element was implemented differently for each method to conform to the approach being used to select the next step in the alignment path, within that method.

4. After processing each frame, the constraint state is set to $S(s, k)$, where S is the constraint table, s is the current constrain state and k defines the input frame selected for alignment to the current target frame using $I_{M_{n+k}}$.

For all methods the distance or cost between the incoming target frame and input frames not allowed by the constraint are set to a very high value of 10,000, forcing the input selection algorithm to not pick them. In methods A and C this step is performed just before the input frame with the minimum cost selected. For method B this step is done between calculating the cost matrix and accumulated cost matrix. By setting corresponding cells within the cost matrix to very high values, the alignment path plotted within the forecast window is forced to go through a frame allowed by the constraint. A backup step was added to method B to catch alignment plots that still go through an input frame not allowed by the constraint, if this is the case, the input frame allowed by the constraint which is closest to the alignment point selected by method B is chosen.

Which input frames an alignment point is allowed or not allowed to be placed at, is dependent on the frames allowed by the current constraint state, as specified in the constraint table. Input frames beyond the dimensions of the constraint table or with null values, are considered outside of the constraint and not allowed.

6.4 Results of Type V Constraint

6.4.1 Test Implementation and Interpreting Results

A Type V constraint was implemented with each of the online time warping methods (A, B and C), from chapter 5. The data-set of motions and performance tests, established in the same chapter, were used to test and measure the impact of the

Type V constraint on the performance of each method when aligning humans motions, allowing a direct comparison of the performance of each method before and after the constraint was applied.

The results of the alignment, distortion (Figure 6.4) and accuracy test (Figure 6.5) have been presented and labeled using the same conventions as the previous chapter. To aid comparison, each algorithm’s performance with and without the Type V constraint applied are presented side by side and colour coded.

6.4.2 Alignment and Distortion Results

The impact of the Type V constraint can be seen in Figure 6.4, which shows the results of the alignment and distortion test on each algorithm, with and without the constraint applied. The addition of the constraint improved the performance of most algorithms at aligning motions and reduced the distortion of all algorithms.

While the algorithms using Methods A and B were all improved by the constraint, it attained less consistent results when applied to algorithms using method C. The only algorithm using Method C that was improved by the constraint, had the largest window size (80 frames). The improvement of the *Mc W80 + Type V* algorithm over *Mc w80*, was significantly more substantial, with a increase of 0.149 in the median alignment test result, than that attained for algorithms using Method B, which achieved an average increase of 0.029 in the median of the alignment test results. In addition to *Mc w80* the alignment performance of algorithm *Ma* was also significantly improved by the constraint, with a 0.181 increase in the median alignment test result. The implementation of constraints raised the alignment performance of both the *Ma* and *Mc w80* algorithms above the performance threshold set by UTW.

Table 6.4 presents a different view of the alignment results, breaking down the performance of each algorithm, based on the types of movement contained in the motion. The cells in green indicates a higher mean (μ) or standard deviation (σ) than that of the same algorithm and movement type without the constraint applied, while red indicates it is lower. An algorithm improved by the constraint should exhibit a higher mean alignment score, to indicate better performance, and a lower

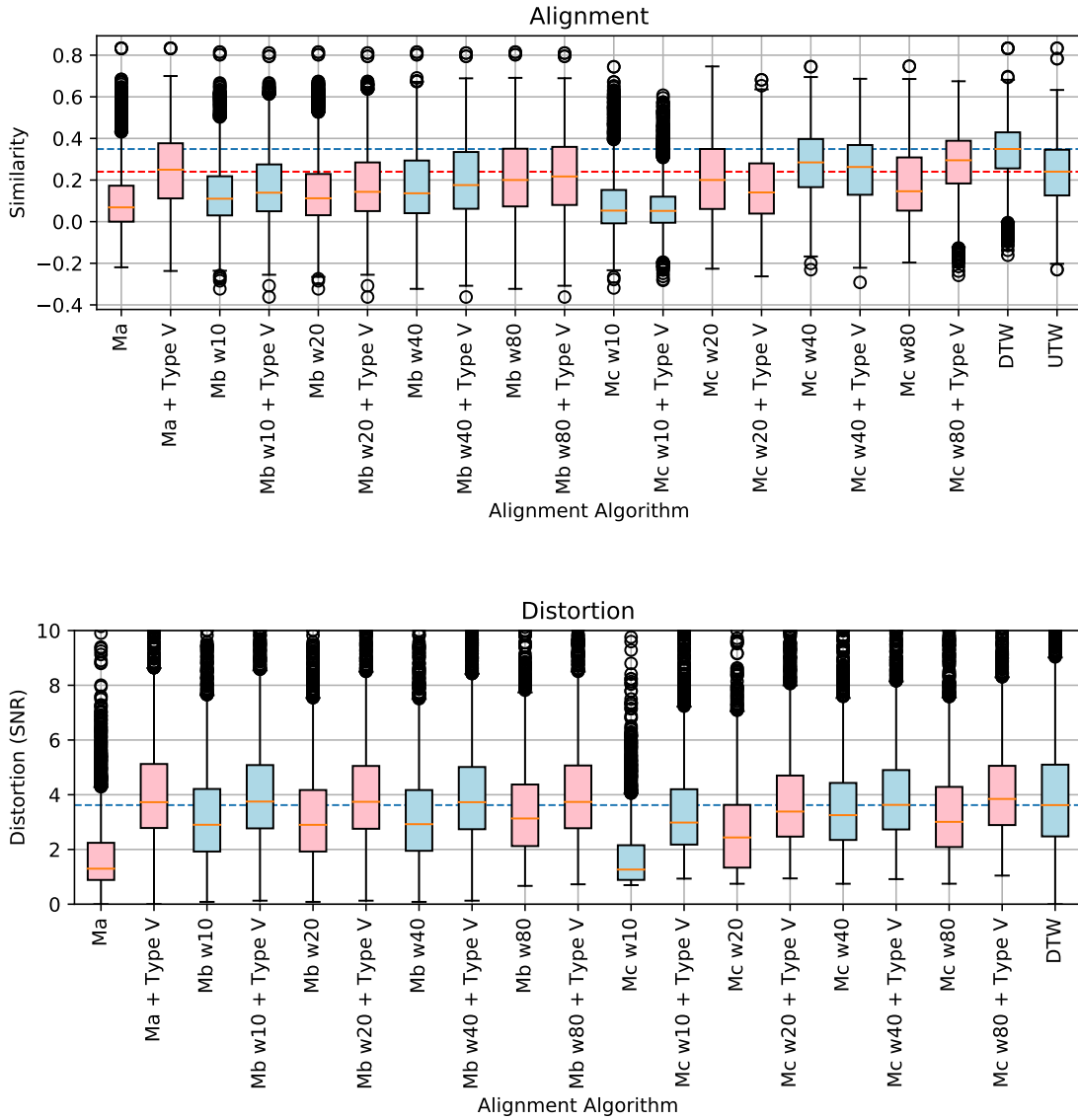


Figure 6.4: The alignment and distortion performance tests results for different time warping algorithms, with and without the Type V constraint applied. In both tests the higher the result the better. The blue line represents the alignment and distortion achieved by the standard offline DTW algorithm, and the red line represents the alignment achieved using UTW with no alignment. Ma, Mb and Mc refer to methods A, B and C respectively, while w specifies the window size used.

standard deviation to indicate a more consistent performance.

Table 6.4: The means and standard deviations of the alignment test scores achieved with the Type V constraint applied to each algorithm, for motions containing different types of movement. Green and red values indicate higher or lower values respectively, than those of the same algorithm and motion type without a constraint applied. The higher values in green indicate an improvement in performance.

Method	Cyclic		Non Cyclic		Ballistic		Lower Body		Upper Body	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Ma + Type V	0.263	0.177	0.226	0.192	0.250	0.148	0.269	0.179	0.237	0.178
Mb w10 + Type V	0.167	0.154	0.164	0.184	0.143	0.130	0.175	0.165	0.161	0.163
Mb w20 + Type V	0.179	0.169	0.169	0.190	0.158	0.145	0.186	0.177	0.170	0.172
Mb w40 + Type V	0.214	0.182	0.185	0.199	0.203	0.152	0.219	0.188	0.196	0.184
Mb w80 + Type V	0.238	0.180	0.211	0.201	0.232	0.147	0.247	0.185	0.223	0.184
Mc w10 + Type V	0.075	0.115	0.060	0.154	0.092	0.102	0.073	0.128	0.077	0.135
Mc w20 + Type V	0.181	0.161	0.130	0.187	0.219	0.136	0.177	0.172	0.166	0.170
Mc w40 + Type V	0.273	0.151	0.202	0.194	0.288	0.117	0.271	0.164	0.239	0.168
Mc w80 + Type V	0.299	0.140	0.249	0.182	0.294	0.112	0.306	0.147	0.272	0.157
DTW	0.356	0.137	0.320	0.152	0.348	0.098	0.373	0.122	0.329	0.141

Changes in the standard deviation, between algorithms with and without the constraint applied, are mainly a function of changes in the mean (i.e. a higher mean score causes a higher standard deviation and visa versa). Where this correspondence between the mean and standard deviation is broken is of particular interest, for example algorithm *Mc w40* performing both worse (lower μ) and less consistently (higher σ) for motions containing: Non-Cyclic; Lower Body; or Upper Body movements, when the constraint is applied to it. Meanwhile, *Mc w80* performing better (higher μ) and more consistently (lower σ) for motions containing all but the Ballistic movement type, when the constraint is applied to it.

Although smaller, the improvements attained by the constraint are slightly more consistent across the different motion types, when applied to Method B than Method C. Out of 20 combinations of windows size and movement type, analysed in Table 6.4, constraints achieved improvements in the mean for 19 out of 20 combinations when applied to Method B and only 6 out of 20, when applied to Method C.

The bottom graph in Figure 6.4, shows that the implementation of the constraint consistently improved the performance of every algorithm in the distortion test, reducing the distortion within the aligned motions, with an average increase of 1.045

in the distortion test score. This is to be expected as the constraint is restricting the choice of possible alignment paths to those closer to the UTW solution. Note that the higher the SNR, the less distortion present in the aligned motion. As with the alignment test, the implementation of a constraint achieved a significant improvement in the performance of algorithm *Ma*, increasing the score in the distortion test by 2.430.

In the distortion test, Figure 6.4, 7 of the 9 time warping algorithms tested beat the benchmark set by the DTW algorithm, when the Type V constraint was applied. It is tempting to see this as an on-line algorithm performing better than off-line DTW, but this potentially indicates the alignment solution being overly constrained, with the constraint itself preventing the algorithm from finding the optimal alignment path, which may be further away from the UTW solution than the constraint allows. It is therefore important to check the results of the distortion test against the corresponding results of the alignment test, to determine if the reduction in distortion is due to better quality or overly constrained alignments. For example in comparison to *Mc w40*, the implementation of the constraint in algorithm *Mc w40 + Type V* reduced distortion, increasing the distortion score by 0.375, but also degraded the alignment, reducing the alignment score by 0.022. This indicated that the improvement in distortion is likely to be the result of the solution being overly constrained, not better aligned. In contrast the implementation of the constraint in the *Ma + Type V* algorithm, resulted in performance improvements in both the alignment and distortion tests, with the scores increasing by 0.181 and 2.430 respectively over the *Ma* algorithm, indicating that the reduced distortion is the result of improvements in the alignment.

6.4.3 Accuracy Results

Figure 6.5 shows the impact of the Type V constraint on the accuracy of the alignment in comparison to an offline DTW warp. Three accuracy tests were performed, each providing a slightly different view of the deviation between the alignment path plotted by the on-line time warping algorithm and the offline DTW algorithm, the smaller the deviation the better the results.

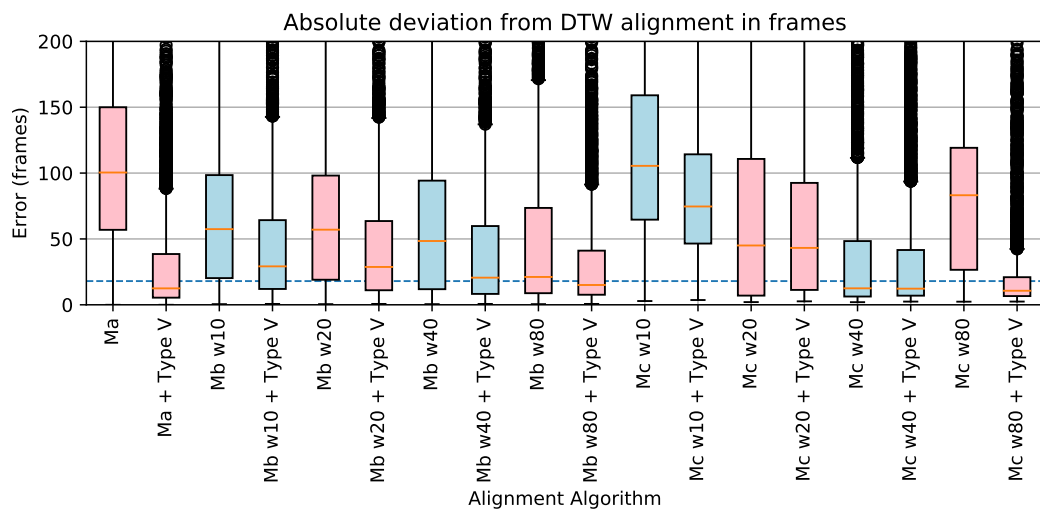
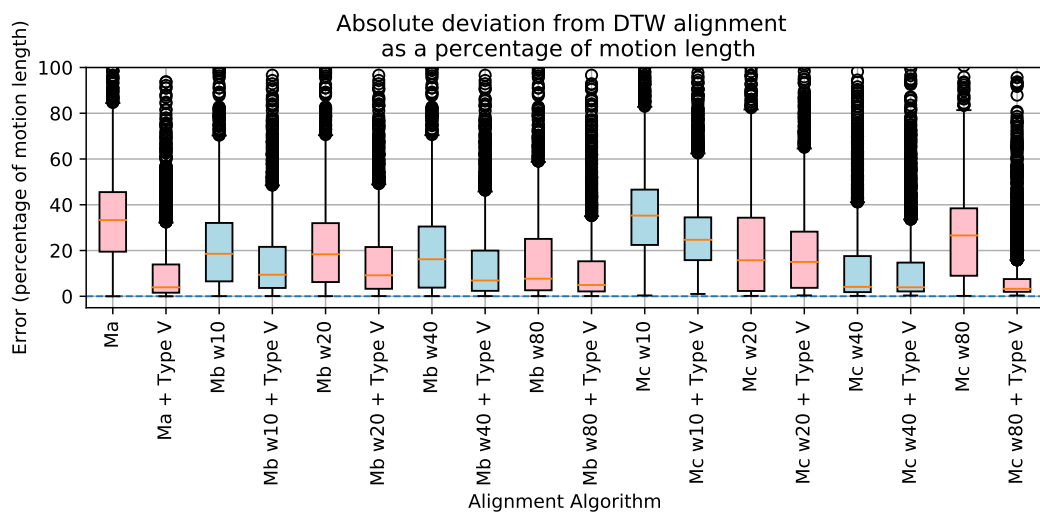
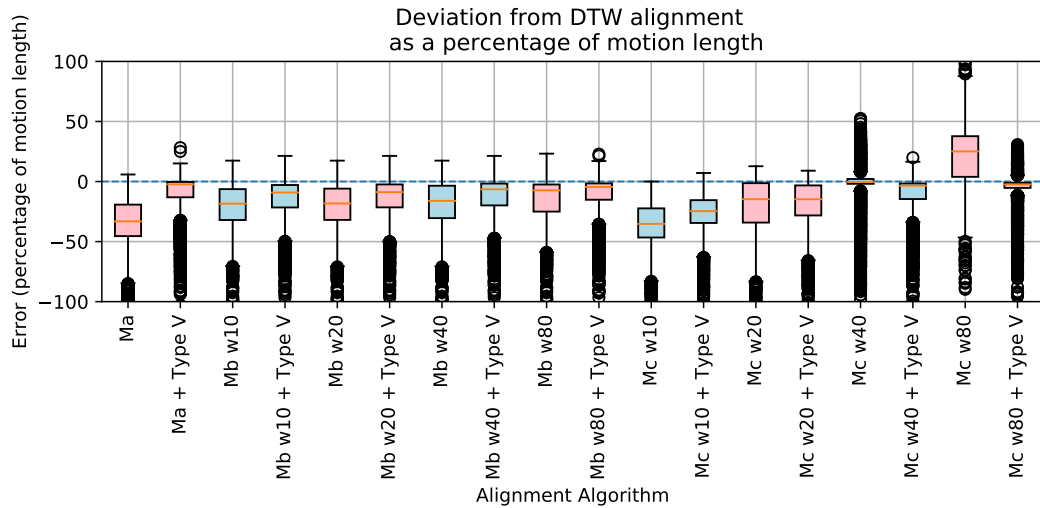


Figure 6.5: The accuracy performance tests results for different time warping algorithms, with and without the Type V constraint applied. Ma, Mb and Mc refer to methods A, B and C respectively, while w specifies the window size used. Results closest to zero are best. The blue line in the bottom chart represents the threshold at which timing errors in character interactions can be perceived (Hoyet, McDonnell and O’Sullivan, 2012).

The results of the three accuracy tests in Figure 6.5, all show the performance of almost every time warping algorithm improving when constraints are applied, apart from algorithms *Mc w20* and *Mc w40*. The constraint also consistently reduced the variation in test results for each algorithm, with the spread of results shown in the box plots of all three graphs in Figure 6.5, being consistently smaller for algorithms with the constraint applied, in comparison to the same algorithm without the constraint applied. However, this is likely to be a function of the mean being moved towards zero, rather than the algorithm with the constraint applied performing more consistently.

The results of the accuracy test largely corroborate the results of the alignment test, with algorithms *Ma* and *Mc w80* again both showing significant improvements in performance when the constraint is applied.

The top chart in Figure 6.5, shows the average deviation and the bias of an algorithm towards warping ahead to behind the optimal alignment. Apart from *Mc w80*, applying the constraint has not changed the bias of any algorithm. Given that the purpose of the constraint is to modify the behaviour of the time warping algorithm, rather than fundamentally change its behaviour, this outcome is to be expected. The exception of *Mc w80 + Type V* is made possible by the mean of the results being very close to zero and limited variation in the results.

The bottom chart in Figure 6.5 shows the average absolute deviation between the alignment paths determined by a given algorithm and DTW, in frames. The implementation of the Type V constraint takes the deviation of an additional three algorithms (*Ma*, *Mb w80* and *Mc w80*) below the threshold of 18 frames, at which timing errors in character interactions can be perceived (Hoyet, McDonnell and O’Sullivan, 2012), as shown by the blue line and established in the previous chapter. The median deviations for algorithms (*Ma*, *Mb w80* and *Mc w80*) were reduced from: 100 to 12, 21 to 15, and 83 to 11 respectively. The *Mc w80 + Type V* algorithm performed particularly well in this test, with 70% of the alignment paths sampled falling below the threshold. Without the constraint, only one algorithm was able to meet this threshold and offer a potentially viable solution to on-line time warping

of human motion. With the constraint applied there are now four algorithms with this potential.

6.4.4 Computational Performance

Table 6.5 shows the impact of the Type V constraint on the computational performance of each algorithm. Although the constraint requires some additional processing steps when added to an algorithm, they are not processor intensive, so have little impact on the overall performance.

Table 6.5: The average time each algorithm takes to process a single frame of motion data in microseconds.

	window size (w)			
	10	20	40	80
Method B	856 μ s	1654 μ s	4806 μ s	16945 μ s
Method B + Type V	813 μ s	1543 μ s	4803 μ s	16928 μ s
Method C	782 μ s	1583 μ s	4764 μ s	10106 μ s
Method C + Type V	649 μ s	1330 μ s	4021 μ s	14099 μ s

Although extra steps are added, the results show that the addition of this type of constraint often reduces the average per frame computation time. Where a constraint state constrains the next step in the alignment path to only one choice of input frame (e.g. State C), the entire process of evaluating which is the optimal input frame is skipped, saving computation time. For Method B the computational time was reduced by: 43 μ s; 111 μ s; 3 μ s; and 17 μ s, for window sizes: 10; 20; 40; and 80 respectively, while for Method C the computational time was reduced by: 133 μ s; 253 μ s; and 743 μ s, for window sizes: 10; 20; and 40 respectively. The larger reductions in the computational times of Method C in comparison to Method B, suggest that the Method C time warping algorithms enter State C during alignments more often than the Method B algorithms. When an algorithm enters state C, no computation is performed to select an input frame, as there is only one input frame to select from. Additional research would be needed to be performed to confirm this, modifying the algorithms to track the number of times each state was entered into during an alignment, however, the frequency at which an algorithm enters State C,

is indicative of how often a constraint has to step in to prevent an alignment from sticking on an input frame.

The only exception to the computational savings attained by the constraint, would appear to be algorithm *Mc w80 + Type V*, where the constraint appears to have significantly increased the computational time. However, this is an anomaly, as the increase is caused by the constraint preventing the algorithm from incorrectly skipping to many input frames, and running out of frames to align the target motion to, early in the alignment process, as discussed in section 5.4.5. The reduction in frame skipping between algorithms *Mc w80* and *Mc w80 + Type V*, results in the computationally expensive process of evaluating choices of input frames to map a target frame to, occurring on more frames during the alignment, therefore increasing the average time each frame takes to process.

6.4.5 Aggregate Plots

A visualisation of the alignment paths plotted by each time warping algorithm, when the Type V constraint is applied, can be seen in Figure 6.6. Using an identical process to that used in the previous chapter in Section 5.4.6, the alignment paths plotted for each combination of motions in the data-set, have been aggregated into a single heat-map, visualising the performance and characteristics of each algorithm.

The addition of the constraint has reduced the overall variation between the aggregate plots produced by each alignment algorithm. This demonstrates that where an algorithm had a tendency towards skipping or sticking on input frames, as shown in plots for *Mc w10* and *Mc w80* respectively of Figure 6.7, this is reduced or more controlled when a constraint is applied, as shown in plots *Mc w10 + Type V* and *Mc w40 + Type V* respectively. Figure 6.7, places the aggregate plots of selected algorithms, with and without the constraint applied, next to each other for direct comparison. Examining the two algorithms which were most enhanced by the constraint *Ma* and *Mb w80*. In the case of the *Ma* algorithm, the hot area below the diagonal, indicating alignment paths incorrectly sticking on input frames, is reduced when a constraint is applied in algorithm *Ma + Type V*. In the case of the *Mc w80*,

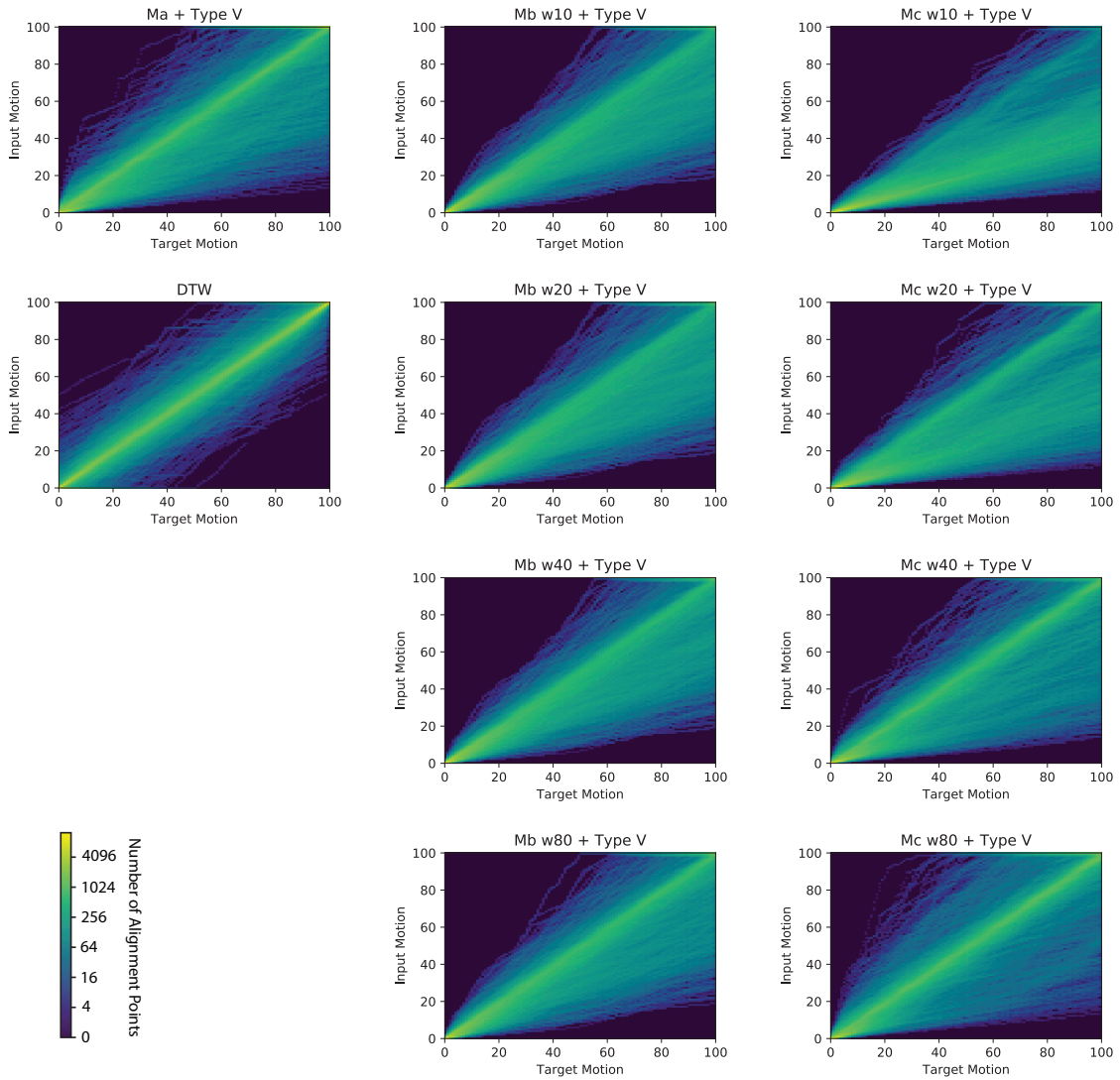


Figure 6.6: Heat maps visualising the alignment paths plotted for all samples in the dataset, by time warping algorithms with the Type V constraint applied.

the hot area above the diagonal, indicating frames being incorrectly skipped, is reduced when a constraint is applied in algorithm $Mc\ w80 + Type\ V$. With the constraint applied, both algorithms are plotting paths within a similar triangular area.

The Type V constraint appears to be much better at preventing algorithms from skipping input frames, than preventing them from sticking on input frames. This can be seen across the plots in Figure 6.6, where the area below the diagonal, from (0,0) to (100, 100), typically contains more paths and is brighter than that above the diagonal. Given that an alignment was plotted in both directions for each pair of

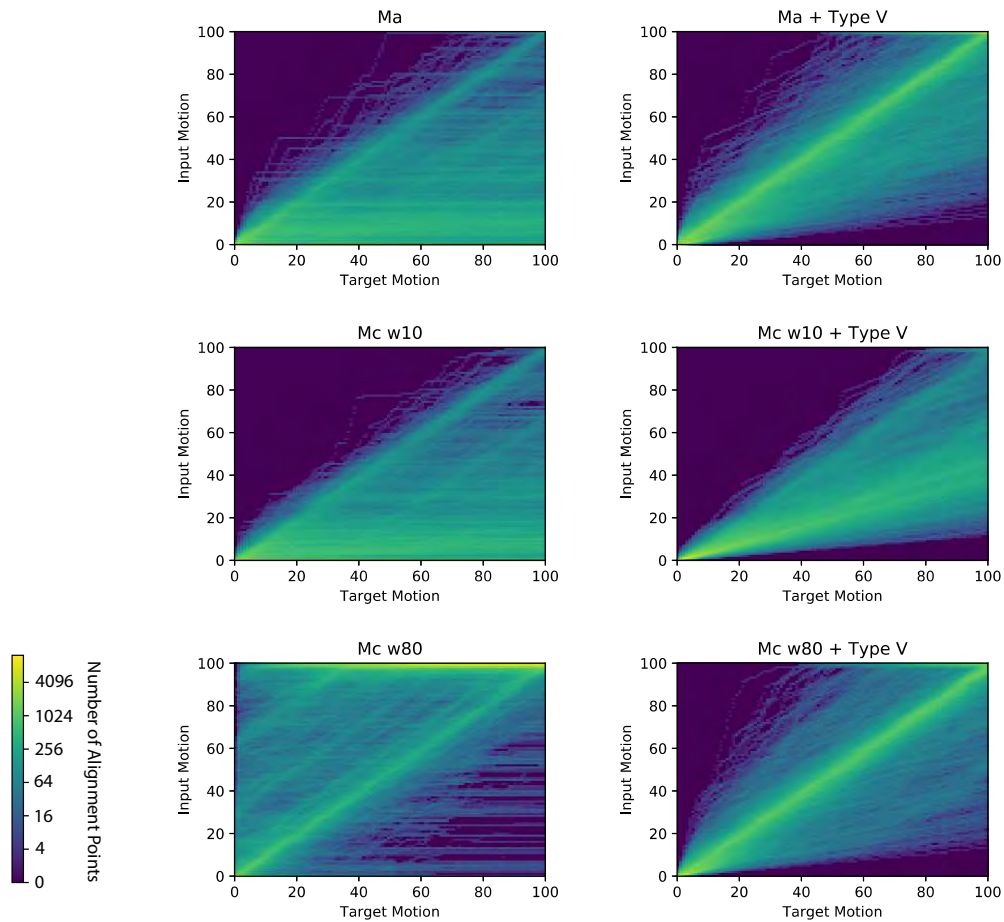


Figure 6.7: A comparison between the aggregate plots of algorithms Ma , $Mc w40$, and $Mc w80$ with and without the Type V constraint applied.

motions within the data-set, aligning motion A to B and visa versa, both areas, above and below the diagonal, should have the same number of paths plotted through them and be equally bright, therefore, if the constraint is successfully controlling any bias towards skipping or sticking on input frames inherent in the time warping algorithm.

In Figure 6.6, the aggregate plot $Mc w80 + Type V$, where algorithm $Mc w80$ has a tendency to incorrectly skip frames, more closely resembles the aggregate plot of the standard DTW algorithm, than plot $Mc w10 + Type V$, where algorithm $Mc w10$ has a tendency to incorrectly stick on a frame. The $Mc w80 + Type V$ plot has clear diagonal hot spot from (0,0) to (100, 100), where most alignments would be expected to be plotted, and a natural scattering of plots along the top edge. The diagonal hot spot in plot $Mc w10 + Type V$, however, is much shallower than where alignment path would be expected to be, and the area where alignment paths are plotted has

a hard edge at the bottom. While the constraint has stopped the algorithm *Mc w10* from sticking on a single input frame, the algorithm is often still failing to correctly return to an appropriate alignment path, instead using the lowest possible number of input frames the constraint allows, resulting in the shallow diagonal highlight.

Although both algorithms *Ma* and *Mc w10* characteristically stick on input frames during alignment, the aggregate plot for algorithm *Ma + Type V* in Figure 6.6 more closely resembled that of the standard DTW algorithm, than the plot for algorithm *Mc w10 + Type V*. Unlike the *Mc w10 + Type V* plot, the gradient and of the diagonal hot spot in plot *Ma + Type V* matches that of the DTW algorithm. This suggest that the constraint is more successful at helping algorithm *Ma* return to an appropriate alignment path, than when applied to algorithm *Mc w10*.

While the constraint struggles to prevent Method C from sticking on an input frame, it is more successful at preventing it from incorrectly skipping input frames. The constraint therefore, works particularly well when method C is implemented with a larger forecast window, where it can control the algorithms tendency to skip frames. The significant performance improvement attained by the constraint when applied to algorithm *Mc w80*, is a good example of this.

6.5 Discussion of Constraint Findings

6.5.1 Key Findings

Out of the nine on-line time warping algorithms proposed in this study, six performed better when they were implemented with a the Type V local continuity constraint. Across the six algorithms that performed better with a constraint, there was an average improvement of 0.074 in the alignment test score of the median sample. Where the constraint did not improve performance, it typically resulted in only small drops in performance, with drops in the median score of: -0.001; -0.060; and -0.022 for algorithms: *Mc w10*; *Mc w20*; and *Mc w40* respectively. This indicates that where an algorithm has not been tested with this constraint, it would still be worth applying it, as it is more likely to improve the performance of the algorithm.

Additionally, whenever the less likely outcome of a reduction in performance occurs, its impact is likely to be limited. Although the implementation of the constraint in algorithm *Mc w40 + Type V* negatively affected the performance of the algorithm, it was still able to meet both the UTW threshold at the top of Figure 6.4 and the threshold of acceptable deviation from the standard DTW solution in Figure 6.5.

The constraint raises the performance of a range of on-line time warping algorithms, which were previously not viable options for real-time alignment of human motion, to a standard where they become viable options. For example the chart at the bottom of Figure 6.5, shows the deviation of algorithms: *Ma + Type V*; *Mb w80 + Type V* and *Mc w80 + Type V*, from the offline DTW algorithm are all be below the threshold at which timing errors in human motion can be perceived. Additionally the chart at the top of Figure 6.4 shows the performance of algorithms *Ma + Type V* and *Mc w80 + Type V*, are above the performance threshold set by the offline UTW algorithm. Importantly this provides a greater range of options to choose from when aligning human motion, removing the reliance on the single algorithm able to meet this threshold without a constraint applied, *Mc w40*.

With constraints applied, the best performing algorithm is *Mc w80 + Type V*, however, it was established in Chapter 5 that a window size of 80 would be computationally too expensive to support real-time applications. Two options for making this window size viable are: i) adapting the algorithm to support parallel processing of the cost matrix, or ii) down sampling the 80 x 80 window to a 40 x 40 window and using interpolation to fill in the missing cells in the cost matrix.

6.5.2 Sticking on Input Frames

The Type V constraint is better at preventing algorithms from incorrectly skipping input frames during alignment, than preventing them from incorrectly sticking on input frames. Figure 6.7 shows that the Type V constraint is better able to reduce the occurrence of frame sticking, when applied to an algorithm based on time warping method A, than method C. This suggests that the cause of this shortcoming stems from the method used by the algorithm rather than the constraint.

The discussion in the previous chapter highlighted how alignment paths determined by method C, tend to stick input frames when the method is applied using smaller window sizes. While the implementation of the constraint prevents alignment paths from sticking on a single input frame, it does not correct the underlying shortcoming within the method. Therefore, rather than the constraint successfully intervening to return the alignment to the optimal path, the problem often continues to persist for the entire time warp, resulting in an alignment path which uses the minimum set of input frames that the constraint will allow.

6.5.3 Impact of Global Constraints

The areas of a cost matrix which can be accessed by a constraint, as determined by their q values in Table 6.2, have been plotted in Figure 6.8, over a heat map showing the alignment paths plotted by DTW for the entire data-set. Note that the forward plotting approach used by the time warping algorithms in this study, means that two slopes passing through (m, n) in the Itakura Parallelogram do not apply. This is reflected in Equation 6.8 which defines the accessible area.

The plots visualises how well suited constraints with a given q value, are to the time warping needs of the data-set being aligned. It shows that a constraint type with a q value of two or three is required, which is satisfied by the Type V constraint.

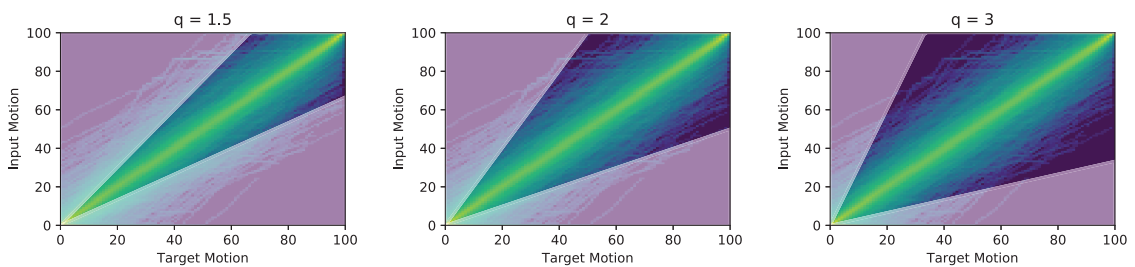


Figure 6.8: A visualisation of how well constraints with a given q value, fit the time warping requirements of the data-set used in this study.

$$\frac{x-1}{q} + 1 \leq y \leq q(x-1) + 1 \quad (6.8)$$

The overall shape of the accessible area defined by a local continuity constraint,

indicates that they can potentially limit the ability for a time warping algorithm to deal with any delay in movement within an input or target motion, as the alignment necessary to deal with these fall outside of the accessible area.

6.5.4 Appraisal of Constraint Study

The performance tests and plots used in this study have provided a robust quantitative assessment of the impact of implementing a constraint on the performance of time warping algorithms. Along with the simulations presented in the previous chapter, they provide valuable insights into how the constraints interact with the characteristics of particular time warping methods.

The distortion performance test, which appeared to be of limited value in the previous chapter, was proven to be of more valuable in this study. The test can potentially identify where alignment algorithms are overly constrained towards the UTW time warp. An algorithm with a good performance in the distortion test, which is not supported with a corresponding score in the alignment test, could potentially be overly constrained.

It should be noted that the study which identified the Type V constraint as the best performing constraint for on-line time warping ([Macrae and Dixon, 2010](#)), applied constraints to an algorithm that plotted alignment paths to align features in a musical score. The differences between these studies suggests that it would be worthwhile evaluating the performance of other types of constraint in this scenario.

Given the compromises of using local continuity constraints in a forward plotting time warp, constraint types that allow more paths to be evaluated in a single step such as Type V and Type VII are better suited to this application. Consideration should also be given to how well the q value of a constraint Type fits the alignment requirements of the data-set.

6.6 Penalties Methodology

6.6.1 Motivation and Overview

The motivation for implementing penalties in this chapter is different to previous studies. [Hülsmann et al. \(2017\)](#) and [Tormene et al. \(2009\)](#) implemented penalties based on path length, to remove bias within the DTW algorithm towards shorter alignment paths, as this creates a positive biasing towards matching a query time series with shortest length reference time series. [Dixon \(2005\)](#) applied penalties to encourage the alignment path to warp the motion, rewarding alignment that moves away from a straight diagonal path. In contrast to these studies, the motivation in this chapter is to use penalties to constrain the amount of warping in an alignment path, potentially reducing the magnitude by which a given algorithm might incorrectly skip to stick on input frames while plotting an alignment. In contrast to previous studies penalties are being used to reduce the amount of warping not encourage it. This contrasting motivation is reflected in the implementation of penalties in this chapter.

On-line time warping requires a penalty system that works with only partial knowledge of one of the motions, and consequently without knowledge of the entire cost matrix or alignment path. This supports an approach in which penalties are determined and applied to each frame independently of each other.

There are two elements to the penalty system: a penalty factor p_f , determined by the position of a potential alignment point in relation to the previous alignment point; and a penalty coefficient p_c , a multiplier applied to the penalty factor which controls the magnitude of the penalty. The penalty applied to a potential alignment point p_i determined by $p(i) = p_f(i) \cdot p_c$.

6.6.2 Determining the Penalty Factor

To support the motivation above, penalties are applied to each time warping method to penalise sticking or skipping of input frames when plotting an alignment path.

For methods A and C alignments penalties are applied to alignment points which

do not step forward one frame in both the target and input motion. Both methods plot a single alignment point for each target frame, imposing a single frame step in the target motion between each alignment point. Therefore, it is the number of input frames between each alignment point that determines the penalty factor, more specifically the further the number is away from one, the larger the penalty factor.

Figure 6.9a shows the penalty factors for each potential alignment point, after the last plotted alignment point in cell (2,2). The monotonic nature of alignment paths means that any input frames before the previously mapped input frame can not be considered. Mapping the next target frame to the same input frame as the previous target frame, incurs a penalty of one, as there will be zero input frames between the alignment points, causing the alignment to stay or stick on the same input frame. Mapping the next target frame to the next input frame, gives a penalty factor of zero, resulting in no penalty. Mapping to the two or more inputs frames ahead of the last alignment point, means the alignment is skipping input frames and therefore incurs a penalty. The penalty factor increases by one for each frame that is skipped.

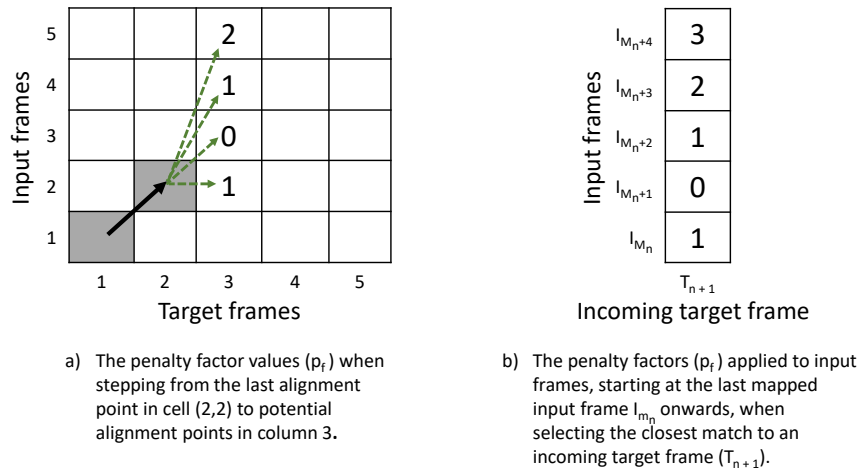


Figure 6.9: Determining the penalty factors to apply to input frames within methods A and C

Given that $M = \{M_0, M_1, \dots, M_n\}$ contains the input frames which have been mapped to each target frame up to the last aligned target frame n . Figure 6.9b shows the penalty factors applied to the previously mapped and subsequent input frames $\{I_{M_n}, I_{M_n+1}, I_{M_n+2}, I_{M_n+3}, I_{M_n+4}\}$. The penalty does not reduce or restrict the number of input frames that a target frame can be aligned to, this is imposed by the

window size of the algorithm and any constraints applied. The penalty factor p_f can be determined for any input frame with index i using equation 6.9.

$$p_f = |i - (M_n + 1)| \quad (6.9)$$

In both methods A and C, penalties are applied after the cost of aligning the target frame to each potential input frame has been determined. In the case of method C this is after the accumulated cost matrix has been calculated. The calculated penalty values are added to the alignment cost of each corresponding input frame. This influences which input frame is considered the optimal choice for the next alignment point, as this is determined by the frame with the lowest cost.

While methods A and C both determine which input frame is the optimal choice for the next alignment point, directly from their alignment costs, method B determines this using an alignment path. Therefore a different approach to determining penalty factors and applying penalties needs to be used with method B, which influences the alignment path that this method plots through the forecast window. To achieve this, equation 6.10 is used to calculate the penalty factor for every cell in the accumulated cost matrix, where Q is a matrix matching the dimensions (m, n) of accumulated cost matrix D . Note that the equation rounds values to the nearest whole number. Examples of the penalty factor matrices generated by equation 6.10 for different values of (i, j) , can be seen in figure 6.10.

The penalty factors Q are used in equation 6.11 to create an accumulated cost matrix that includes penalties D_p , where p_c is a normalisation coefficient and D is the accumulated cost matrix. The alignment path is then plotted through matrix D_p .

$$Q_{(i,j)} = \left[\left| \frac{i}{m} - \frac{j}{n} \right| \cdot \min\{m, n\} \right] \quad (6.10)$$

$$D_p = D + (Q \cdot p_c) \quad (6.11)$$

4	4	3	3	2	2	1	1	0	0
3	3	2	2	1	1	0	0	1	1
2	2	1	1	0	0	1	1	2	2
1	1	0	0	1	1	2	2	3	3
0	0	1	1	2	2	3	3	4	4

9	8	7	6	5	4	3	2	1	0
8	7	6	5	4	3	2	1	0	1
7	6	5	4	3	2	1	0	1	2
6	5	4	3	2	1	0	1	2	3
5	4	3	2	1	0	1	2	3	4
4	3	2	1	0	1	2	3	4	5
3	2	1	0	1	2	3	4	5	6
2	1	0	1	2	3	4	5	6	7
1	0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8	9

4	3	2	1	0
4	3	2	1	0
3	2	1	0	1
3	2	1	0	1
2	1	0	1	2
2	1	0	1	2
1	0	1	2	3
1	0	1	2	3
0	1	2	3	4
0	1	2	3	4

a) 10 x 5 p_f matrix

b) 10 x 10 p_f matrix

c) 5 x 10 p_f matrix

Figure 6.10: Example penalty factor matrices generate by equation 6.10

6.6.3 Determining the Normalisation Coefficient

The magnitude of the penalties applied is controlled by a normalisation coefficient p_c . Applying penalties of an appropriate magnitude is a critical optimisation step, if the penalties are too great then alignment paths will be constrained to a diagonal path, if they are too small they will have a limited or no impact on the alignment.

Given that the penalty factor represents the number of frames the local alignment path is being moved away from a one to one alignment, a logical approach to determining an optimal normalisation coefficient is to find the average difference between the joint poses of two sequential frames of motion data. It also follows that the method used to measure the difference between the joint poses must be the same as that used to measure the alignment cost within the time warping algorithms. Within this study two approaches are used to do this:

1. Global Normalisation: based on the average difference between sequential frames across the entire data-set.
2. Local Normalisation: based on the average difference of sequential frames within the prerecorded input motion being aligned.

Global Normalisation of Penalties

The average difference between sequential frames across the entire data-set D , is calculated using equation 6.12, where data-set D contains p motions, each containing q frames, with r joints being evaluated. As this gives a \bar{d} of 0.034 for the data-set used in this study, a global normalisation around this value should be considered

when implementing penalties in this chapter.

$$\bar{d}(D) = \frac{\sum_{m=1}^p \sum_{f=2}^q \sum_{j=1}^r \text{geoDist}(D_{m,f,j}, D_{m,f-1,j})}{\sum_{m=1}^p \sum_{f=2}^q r} \quad (6.12)$$

Local Normalisation of Penalties

A local normalisation coefficient for motion m can be determined using equation 6.13, where q are the frames contained in motion m and r are the joints being evaluated. The equation calculates the average difference between each pair of sequential frames within the motion.

$$\bar{d}(m) = \frac{\sum_{f=2}^q \sum_{j=1}^r \text{geoDist}(m_{f,j}, m_{f-1,j})}{(q-1)r} \quad (6.13)$$

6.6.4 Implementation of Penalties

To better understand the optimal configuration for penalty coefficients, six different coefficients will be tested as identified in table 6.6. Three global normalisation coefficients will be tested, that surround the $\bar{d}(D)$ value calculated for the data-set. In addition three local normalisation coefficients will be tested, which comprise of different multiples of $\bar{d}(m)$. The motivation is to determine if a coefficient greater or lower than $\bar{d}(D)$ or $\bar{d}(m)$ is a more optimal choice of normalised coefficient. The table also specifies codes that will be used to label the algorithms which have penalties applied, within the results section.

Table 6.6: List of the penalty coefficients to be tested

Test No.	Coefficient Type	Coefficient Value	Test Code
1	Global	0.025	Pf0.025
2	Global	0.050	Pf0.050
3	Global	0.100	Pf0.100
4	Local	$\bar{d}(m)0.5$	Pn0.5
5	Local	$\bar{d}(m)1.0$	Pn1.0
6	Local	$\bar{d}(m)2.0$	Pn2.0

6.7 Results of Implementing Penalties

6.7.1 Test Selection and Implementation

The number of different penalty coefficients being tested, means it would not be practical to test every penalty coefficient on each of the nine algorithms tested in chapter 5. Therefore the worst performing algorithm of each method (A, B and C), along with the best performing algorithm *Mc w40*, were selected to be tested with penalties implemented using the each of the six penalty coefficients in the table 6.6. Algorithms *Ma*, *Mb w10* and *Mc w10*, performed the worst in the accuracy tests in figure 5.8b and showed a tendency to plot alignment paths that stick on input frames. The performance of these algorithms have the most potential to be improved by the implementation of penalties and be influenced different penalty configurations. The best performing algorithm (*Mc w40*) was included in the test to evaluate any negative impact penalties might have on a higher performing algorithm.

Every combination of selected time warping algorithm and penalty configuration in table 6.6 was tested using the data-set motions and performance tests established in chapter 5, to allow a direct comparison of the performance of each algorithm with and without penalties applied. This approach would allow the impact of different penalty configurations on different time warping algorithms to be evaluated.

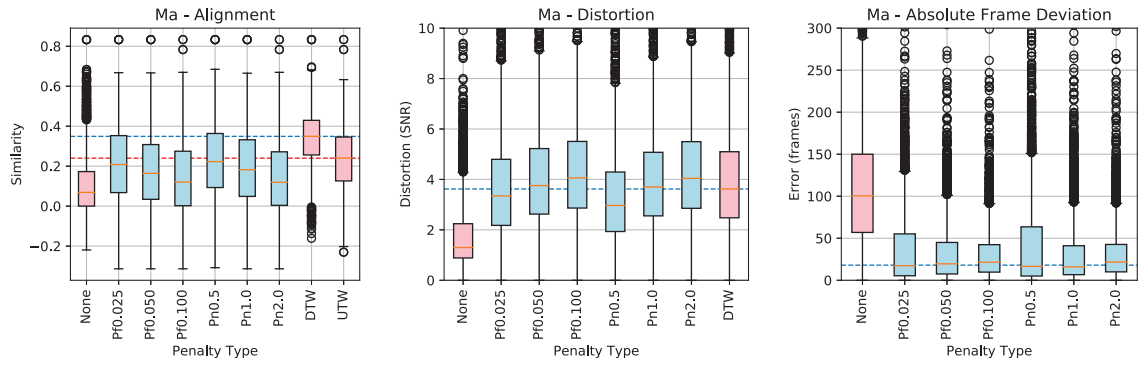
6.7.2 Performance Test Results

The impact of penalties on each of the four time warping algorithms tested, can be seen in figure 6.11. It shows the distribution and median of results for the alignment, distortion and absolute frame deviation performance tests, which were used to evaluate the motion alignments determined by the time warping algorithms, with and without penalties applied. The alignment and performance test are detailed in section 5.3.4, while the frame deviation test, which evaluates how closely an alignment matches that of the offline DTW alignment, is detailed in section 5.4.4. The results shown in blue are algorithms with penalties applied to them, while those in red are algorithms without penalties or the results of the offline DTW and UTW time warping algorithms, which are included for reference. In the alignment and distortion tests, the higher the score the better the performance, with the blue and red lines representing the median score of alignments produced by the offline DTW and UTW algorithms respectively. In the frame deviation test, the lower the score the better the performance, with the blue representing the threshold determined by Hoyet, McDonnell and O’Sullivan (2012), at which a difference between the alignment being tested and the one produced by the offline DTW algorithm would be perceptible.

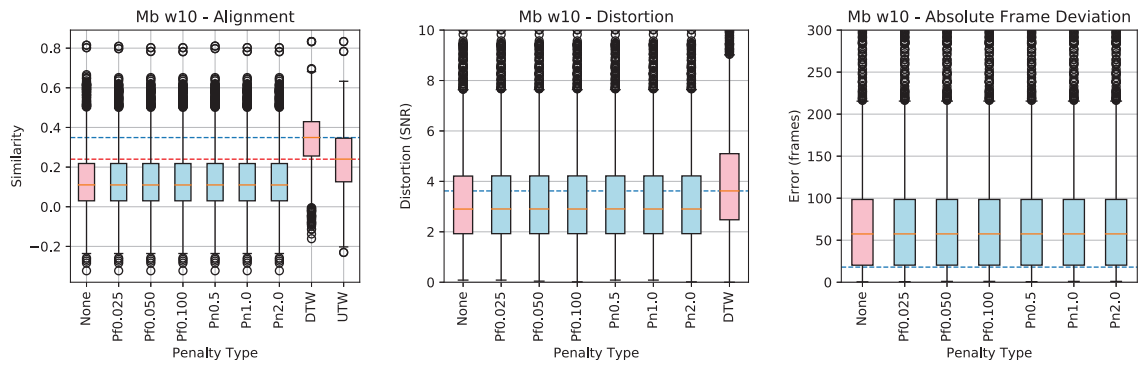
Table 6.7 shows the mean results attained by each algorithm with and without penalties applied across five performance tests, which includes two accuracy tests in addition to the tests described above, which are both described in more detail in section 5.3.4. The result of algorithms with penalties applied have been colour coded green and red to indicate if the performance is better or worse respectively, than the same algorithm without penalties applied.

Note some difference between the mean scores in table 6.7 and the medians shown in figure 6.11. This is due to skews in the distribution of test results within each sample group.

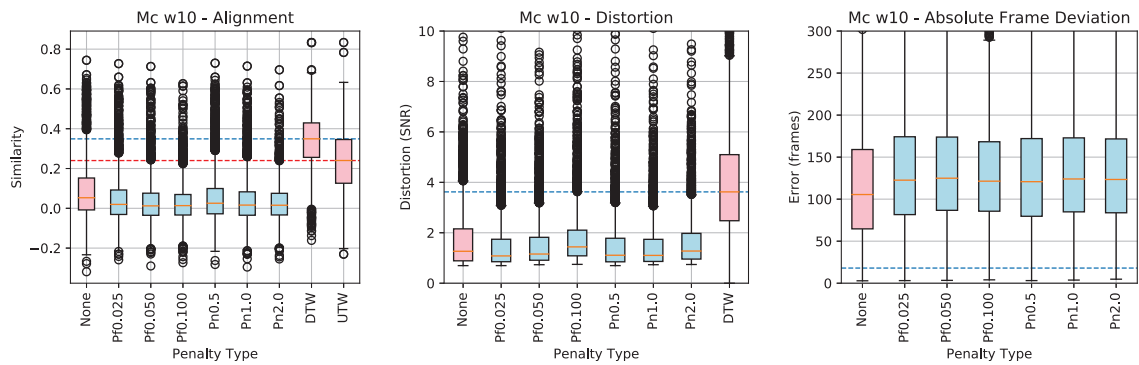
Algorithm *Ma* was the only algorithm to show a significant improvement in performance when penalties were applied. Although the improvement in performance was not enough to take it above the threshold set by the UTW algorithm in the



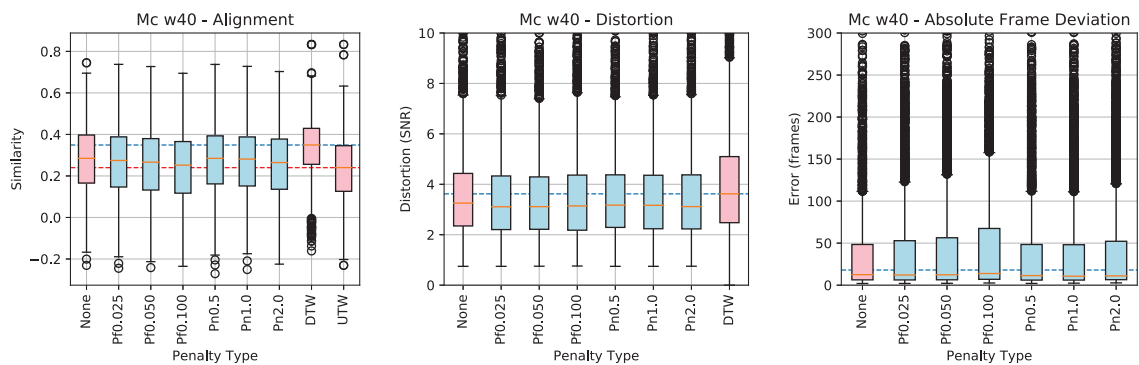
(a) Results for algorithm *Ma*



(b) Results for algorithm *Mb w10*



(c) Results for algorithm *Mc w10*



(d) Results for algorithm *Mc w40*

Figure 6.11: The results for the alignment, distortion and accuracy performance tests for selected time warping algorithms with different penalties of varying strength and penalty coefficient types, applied. The performance of the time warping algorithm with no penalty applied can be seen in red.

Table 6.7: The mean performance test scores attained by time warping algorithms with different penalty configurations applied. Values in green and red indicate algorithms with penalties applied performing better or worse, respectively, than the same algorithm without the penalty applied.

Algorithm	Performance Tests				
	Alignment	Distortion	Average Percent Deviation	Absolute Percent Deviation	Absolute Frame Deviation
Ma	0.106	1.852	-33.684	33.875	117
Ma + Pf0.025	0.216	4.244	-9.220	12.247	37
Ma + Pf0.050	0.176	4.887	-5.393	10.449	32
Ma + Pf0.100	0.145	5.207	-3.578	10.045	31
Ma + Pn0.5	0.233	3.896	-12.666	13.854	43
Ma + Pn1.0	0.197	4.622	-6.332	10.358	32
Ma + Pn2.0	0.144	5.190	-3.723	10.218	32
Mb w10	0.136	3.806	-21.733	21.852	70
Mb w10 + Pf0.025	0.135	3.827	-21.754	21.873	70
Mb w10 + Pf0.050	0.135	3.835	-21.756	21.875	70
Mb w10 + Pf0.100	0.135	3.835	-21.757	21.876	70
Mb w10 + Pn0.5	0.135	3.821	-21.754	21.873	70
Mb w10 + Pn1.0	0.135	3.834	-21.756	21.875	70
Mb w10 + Pn2.0	0.135	3.835	-21.757	21.876	70
Mc w10	0.088	1.814	-35.662	35.707	122
Mc w10 + Pf0.025	0.044	1.567	-40.521	40.543	140
Mc w10 + Pf0.050	0.030	1.635	-41.452	41.473	143
Mc w10 + Pf0.100	0.024	1.927	-40.737	40.768	139
Mc w10 + Pn0.5	0.051	1.572	-39.748	39.771	137
Mc w10 + Pn1.0	0.035	1.567	-41.104	41.122	142
Mc w10 + Pn2.0	0.026	1.739	-41.303	41.323	142
Mc w40	0.282	3.822	-2.871	12.017	36
Mc w40 + Pf0.025	0.268	3.652	-8.366	13.670	43
Mc w40 + Pf0.050	0.259	3.633	-10.934	14.333	46
Mc w40 + Pf0.100	0.244	3.733	-13.764	15.309	50
Mc w40 + Pn0.5	0.276	3.761	-6.746	12.586	39
Mc w40 + Pn1.0	0.271	3.695	-9.033	12.918	41
Mc w40 + Pn2.0	0.258	3.737	-11.738	13.770	44

alignment test, it was enough to reduce the distortion of the alignment to below that of the DTW algorithm in some instances. The algorithm's performance across different penalty configurations, appear to contradict each other in the alignment and distortion tests. This contradiction suggests that higher penalty values are overly restricting the alignment paths plotted by the *Ma* algorithm, forcing it to plot an alignment path that steps forward one input frame for each target frame (i.e. a one to one alignment). While this type of alignment has little distortion it is not an optimal alignment. This is backed up by the frame deviation tests for the *Ma* algorithm, which show the same trend in performance improvement across the different penalty types as that alignment test.

The frame deviation test results for the *Ma* algorithm also show that penalties have reduced the difference between the alignments produced by the *Ma* and DTW algorithms to below the perceptible threshold in some instances. This indicates that penalties can improve the performance of the *Ma* algorithm, to a point where it can produce useful alignments.

In contrast to the *Ma* algorithm, other algorithms did not respond so well to penalties being applied to them. Applying penalties had very limited to no impact on the *Mb w10* algorithm, and caused both algorithms based on method C (*Mc w10* and *Mc w40*) to perform worse. The performance of both algorithm's in the alignment test were similarly affected, both showing a worse performance as the strength of the penalty was increased.

The results of the distortion tests for *Mc w10* and *Mc w40* were more mixed. Algorithm *Mc w10* with the smaller window size, produced alignments with less distortion when penalties were applied, while algorithm *Mc w40* with the larger window size was less effected by application of penalties. The contradicting trends in the results of the alignment and performance test for algorithms *Mc w10* are similar to that observed in the *Ma* algorithm, again suggesting that the stronger penalty coefficients are pushing the alignment path towards a one to one alignment between input and target motion frames.

The frame deviation tests again show the performance of both *Mc w10* and *Mc w40*

algorithms were negatively impacted by the application of penalties. The performance of algorithm *Mc w10* with the smaller window size was more effected than *Mc w40* with the larger window size.

6.7.3 Aggregate Plots

A visualisation of the alignment plots plotted by algorithms *Ma*, *Mc w10* and *Mc w40*, with each of the tested penalty configurations applied and without any penalty applied, can be seen in figure 6.12. Using the same process used in section 5.4.6, the alignment paths plotted for each combination of motions in the data-set, have been aggregated into a single heat-map, to visualise the performance and characteristics of each algorithm with different penalties applied.

The heat maps for the *Ma* algorithm, clearly show the overly constrained alignments resulting from stronger penalties with higher penalty coefficients being applied. This characteristic is the same regardless of whether the penalty is using a global or local normalisation coefficient.

The *Mc w10* and *Mc w40* algorithms also produced worse alignments when higher penalty coefficients were applied. However, the aggregate plots indicate that in the case of these algorithms, this is because higher penalty coefficients are causing the alignment paths to incorrectly stick on input frames. This characteristic is more prominent with Method C algorithms configured with smaller window sizes.

While higher penalty coefficients caused alignment paths to incorrectly stick on frames, when applied to algorithms based on Method C, they did also reduce the occurrence alignment paths incorrectly skipping too many frames.

6.8 Discussion of Penalty Findings

6.8.1 Impact of Penalties on Time Warping Methods

The contrasting approaches used by time warping method A, B and C, has caused them to respond to the implementation of penalties in different ways.

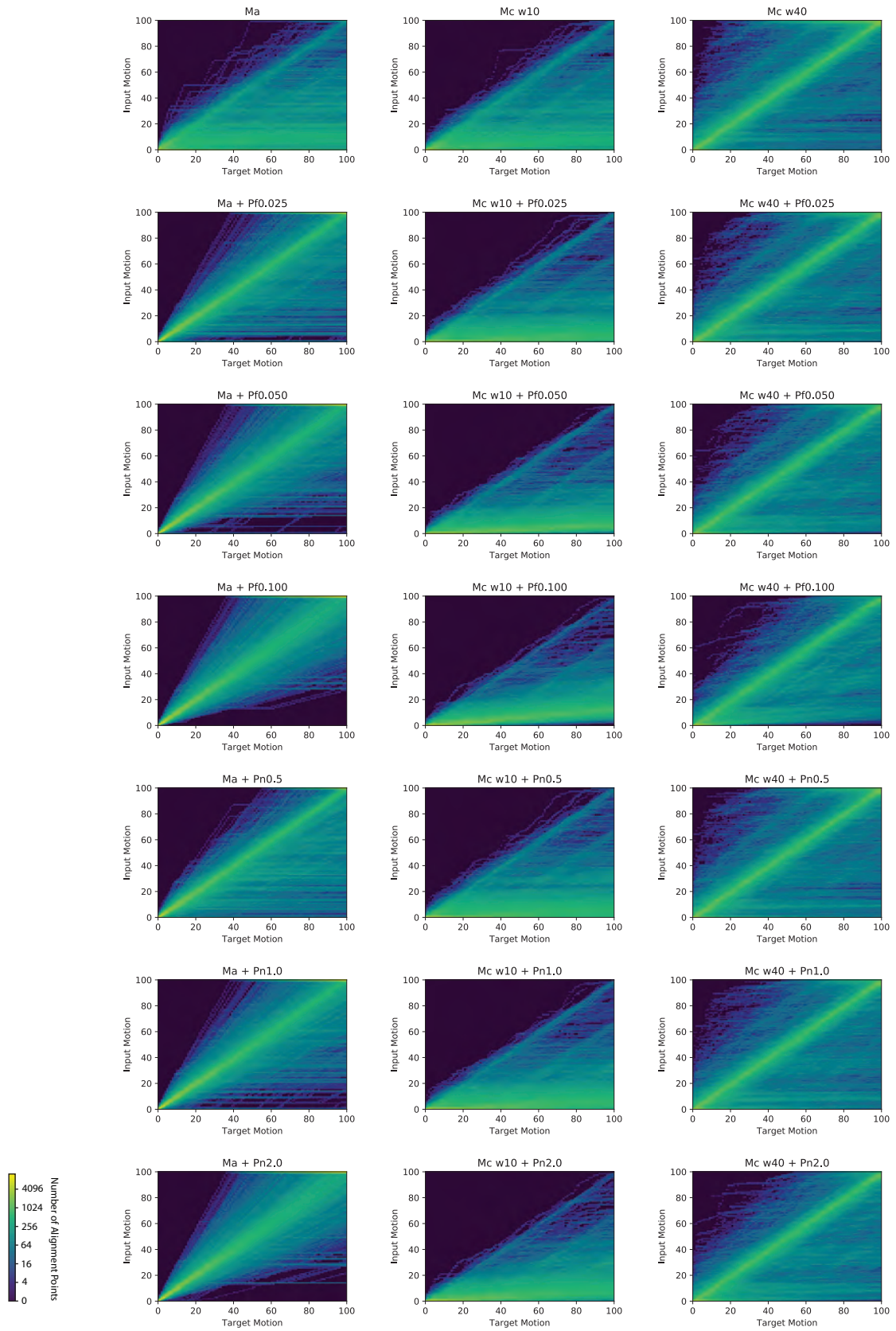


Figure 6.12: Heat maps visualising the alignment paths plotted by algorithms *Ma*, *Mc w10* and *Mc w40*, with different configurations of penalties applied.

Method A showed the most improvement when penalties were applied. This is a combination of the method A algorithm both responding well to the implementation of penalties and having quite a poor initial performance before penalties were applied. Method A was the only time warping method, where the implementation of penalties improved the performance of the algorithm enough, to cross below the threshold at which differences between the alignment produced by on-line time warping method and DTW time warping, are potentially not perceptible.

Algorithms based on Method B do not respond to implementation of penalties, showing little to no improvement or deterioration in performance as a result of penalties being implemented. This is likely due to the approach used by Method B, of plotting a path within the forecast window. As a path is plotted backwards through an accumulated cost matrix, which contains costs accumulated from the bottom left to the top right of the initial cost matrix, the relatively small penalty factor values in matrix Q are likely to have very little impact on the large accumulated costs in the accumulated cost matrix. Therefore, penalties implemented in this way are probably only going to effect one or two of the left most alignment points plotted through the forecast window. Applying the penalty factors in matrix Q to the initial cost matrix, before the accumulated cost matrix is calculated would have a bigger impact.

The performance of both algorithms based on Method C deteriorated when penalties were applied to them. The penalties appeared to exacerbate the problem method C has with alignments sticking on input frames. The performance of algorithm *Mc w10*, with a smaller window, appeared to deteriorate more than algorithm *Mc w40* with the larger window. Despite the performance of the *Mc w40* algorithm deteriorating when penalties were applied, none of the penalties reduced the algorithm's performance enough to take it either below the UTW threshold in the alignment test or above the perceptible error threshold in frame deviation test.

The contrasting impact of penalties on different time warping methods, means a decision on whether to implement penalties is dependent on the method being used. It is recommended that penalties are only applied to time warping algorithms based

on Method A.

6.8.2 Normalisation Coefficients

Across the different performance tests conducted there was not a significant difference between the performance of global and local normalisation of penalties.

When applied to algorithm *Ma*, locally normalised penalties performed better in the alignment test, while globally normalised penalties performed better in the frame deviation test.

In the case of algorithms based on both methods A and C, stronger penalty coefficients resulted in consistently poorer performances in the alignment test, when compared to the same algorithm with smaller coefficients. For any group of time warps using the same type of penalty coefficient, the smallest penalty coefficient performed the best. This suggests that smaller penalty coefficients should be tested and explored to find potentially more optimal coefficient values.

6.8.3 Penalty Implementation

The manner in which penalties were implemented was potentially better optimised for preventing alignment paths from skipping input frames, than preventing them from sticking on input frames.

The penalty factors for determining the optimal alignment point for each target frame, were determined independently each other. While this allowed the penalty factor to be increased for skipped input frames as required, it does not allow the penalty factor to be increased by more than one for sticking on an input frame, regardless of how many consecutive frames the alignment has remained stuck on a particular input frame. This could be corrected by tracking the number of frames an alignment path has stuck on a particular input frame, between processing each input frame.

6.9 Conclusions

In this chapter a constraint and a variety of penalties were applied to the on-line time warping algorithms introduced in the previous chapter. The motivation for this study was to ascertain the impact of these approaches and evaluate their potential to optimise each of these time warping algorithms.

Overall the Type V constraint improved the performance of different time warping algorithms more consistently than penalties. In the alignment test six out of nine algorithms improved their performance when the constraint was applied. In comparison, penalties only enhanced the performance of one of the four algorithms tested. Where the constraint did not enhance the performance of an algorithm it did not degrade performance significantly, unlike penalties which did significantly degrade the performance of some algorithms. This means that penalties should be implemented with more caution than constraints.

Table 6.8 provides an overview of which on-line time warping algorithms, and constraint or penalty configurations, could be considered for use in a real world scenario. It shows which of the algorithms and configurations tested, met or exceeded either of two key performance thresholds. Threshold U, is the quality of alignment threshold set by offline UTW in the alignment test. Threshold F, is the point at which an alignment's deviation from the alignment plotted by the DTW algorithm, is enough to be perceptible. An algorithm is deemed to have met this threshold, if the median alignment from the data-set of motions aligned using the algorithm, meets or exceeds the threshold.

Each of the three time warping methods showed different characteristics when the constraint and penalties were implemented.

Method A responded to the implementation of both constraints and penalties with a significant improvement in performance. The performance of method A is quite malleable, responding positively and predictably to both optimisation methods. Although Method A was initially the worst performing method, both constraints and penalties were each able to improve its performance enough to take it over one of

Table 6.8: Overview of which on-line time warping algorithms met key performance thresholds, both with and without the constraint or penalties applied. U means the algorithm performed better than an offline UTW time warp. F means that the majority of alignments produced by the algorithm, were so close to the that produced by offline DTW, that differences may not be perceptible. Black cells represent test that were not performed.

Constraint or Penalty	Type	Ma		Mb w10		Mb w20		Mb w40		Mb w80		Mc w10		Mc w20		Mc w40		Mc w80		
		U	F	U	F	U	F	U	F	U	F	U	F	U	F	U	F	U	F	
		None																	x	x
Constraint	V	x	x								x						x	x	x	x
Penalty	Pf0.025		x														x	x		
Penalty	Pf0.050																x	x		
Penalty	Pf0.100																x	x		
Penalty	Pn0.5		x														x	x		
Penalty	Pn1.0		x														x	x		
Penalty	Pn2.0																x	x		

the key thresholds in table 6.8.

While algorithms based on Method B did not improve as much as Method A, when constraint and penalties were applied. They did in a consistent manner, showing a small but consistent improvement when the Type V constraint was applied and no improvement when penalties were applied. Neither the constraint or penalties caused a deterioration in performance when applied to an algorithm based on Method B.

Algorithms based on Method C, responded poorly to both the implantation of either the constraint or penalties. The Type V constraint produced inconsistent results, with the performance of some algorithms improving, while the performance of others deteriorated. In addition, the implementation of penalties consistently resulted in a deterioration in performance across different window sizes. Neither approach, constraint or penalties, was able to counteract the problem Method C has, of producing alignments which incorrectly stick on an input frame. However, a useful benefit of using constraints with the Method C algorithm, is that it produced smoother alignment paths and smoother alignments as a result.

While other studies have implemented penalties in such a way as encourage or allow alignment paths to deviate way from a one to one alignment between input and

target frames. In this study penalties were implemented as an alternative form of constraint, discouraging alignments from deviating away too much from a one to one alignment.

Implementing both constraints and penalties together, within a single time warping algorithm has not been explored within this study. Combining constraints and penalties is an interesting area for further exploration, which could potentially create a more optimal time warping algorithm. Two such scenarios which could be explored are:

1. Algorithms such as *Ma*, which responded well to the implementation of both a constraint and penalties, could potentially be further optimised by implementing both.
2. Penalties implemented in such a manner as to encourage warping away from a one to one alignment, and act as a balance against a constraint in an overly constrained algorithm such as *Mc w40 + Type V*.

Chapter 7

Impact of Movement and Motion Data Characteristics on the Performance of Time Warping Algorithms

7.1 Introduction

The the quality of alignment produced by a given time warping algorithm varies significantly, not just across the different movements contained within the data set used in this thesis, but in some cases across alignments of different recordings of the same movement. While all nine online time warping algorithms tested in Chapter 5, achieved a median score in the alignment test that was lower than that achieved using offline DTW, the variation of the scores for every algorithm was higher than than of offline DTW, which had a standard deviation of (0.141), with the alignment scores of algorithms *Mc w10* and *Mb w80*, having the lowest (0.146) and highest (0.188) standard deviations respectively. Additionally motions such as ‘Stand Up Sit Floor’ and ‘Elbow To Knee 3 Reps L Start’ exhibit a high variance in the alignment test scores of time warps performed on different recordings of the same movement, with an average standard deviation of 0.201 and 0.195 respectively, across the different

time warping algorithms tested in Chapter 5.

These variations in alignment scores suggests that the on-line time warping algorithms proposed in Chapter 5 perform less consistently than the offline DTW algorithm, and are potentially more affected by subtle features in a recorded motion, such as a mistimed start or unusual pose. In addition the characteristics of the movements being performed, such as the number of joints being utilised and features contained in the motion (e.g. ballistic and cyclic movement), also have the potential to cause a time warping algorithm to perform inconsistently. This chapter explores the impact of various characteristics in the movement and data of the motions being aligned, on the performance of on-line time warping algorithms, with the aim of developing a deeper understanding of how to optimise the implementation and application of the on-line time warping algorithms presented in Chapters 5 and 6. This chapter will explore the impact of the following characteristics:

- **Under Utilised or Under Constrained Joints:** Movement and pose variations that occur in joints that are not utilised or constraint by the movement being performed.
- **Data Capture Errors:** Errors such as noise and dropped frames, that can occur in the data capture process.
- **Mistimed Starts:** Differences in the timing of the start of the input and target motions.
- **Speed Differentials:** Differences in the speed at which input and target motions are performed.

The first two sections of the chapter outline the danger of overly weighting joints which are not particularly pertinent to a motion, Section 7.2, then discusses errors that can occur in motion data, Section 7.3. Where possible the occurrence of these data errors in both the BCU and HDM05 data-sets have been measured.

Sections 7.4 and 7.5, present a more detailed evaluation of two characteristics which

particularly impacted the performance of the on-line time warping algorithms presented in this thesis, mistimed starts and speed differentials respectively.

7.1.1 Identification of Most Impactful Characteristics

Within the data-set, every unique combination of two motions appears twice, with the input and target motion swapped around in one occurrence (i.e. one sample aligns input motion A to target motion B, while another sample aligns input motion B to target motion A). The results of the tests performed in Chapter 5, were searched to find the combinations of two motions with alignment test scores that diverged the most between their two occurrences. When these high divergent motion combinations were reviewed, it was found that they often contained motions with mistimed starts or motions performed at different speeds.

7.2 Under Utilised and Under Constrained Joints

Under utilised joints refer to joints which are not directly utilised by a given motion. An example can be seen in the kick motion on the right of Figure 7.1, in which the pose of the arms and hands, which are not directly involved in the motion, vary considerably between motions A and B.

Under constrained joints, are joints which are used by a given motion but do not have to be in a particular pose to achieve it. An example can be seen in the clapping motion on the left of Figure 7.1, in which the clapping movement can be achieved with arms in a higher or lower position (i.e. spatial variation), as seen in the side views, and with arms spread apart to different extents between claps as seen in the front view of pose A (i.e amplitude variation). In this example, the under constraint leads variations between the poses of the arms and hands, despite being utilised by the motion.

The variations between the poses of unconstrained motions also result in variations in the motion curves, as shown in Figure 7.1. For example the motions curves for the shoulder joint of kick motions A and B, shown on the right, all start with a form

that is symmetrical (or opposite) to each other.

A consequence of the greater variation between the poses of unconstrained motions, is that they do not respond as well to time warping algorithms, generally resulting in less satisfactory alignments, than more constrained motions. This variation in the performance of on-line time warping algorithms, between aligning under constrained and unconstrained motions, can be seen in Table 7.1, which shows the mean similarity score attained by the three best performing algorithms, *Ma + Type V*, *Mc W40*, and *Mc w80 + Type V*, for each type of motion sourced from the HDM05 data-set. As other factors, associated with the capture of a particular data-set, can also affect the performance of time warping algorithms, this table compares only motions sourced from the single data set. These factors are explored further in the proceeding section.

Table 7.1 shows time warping algorithm performing worse when aligning under constrained motions, such as the clap and kick motions shown in Figure 7.1, compared to aligning more unconstrained motions, such as the walk and skier reps shown in Figure 7.2. The skier motion, sometimes referred to as cross-ski, is a cardio vascular exercise that mimics the movement of cross country skiing. The pose of both the upper and lower body is well defined, with the exercise swapping between, a right arm and left leg forward with left arm and right leg back pose, and a right arm and left leg back with left arm and right leg forward pose, with limited possible variation in how a subject can transition between these two poses.

Motions which only directly utilise the upper or lower body such as: kicks; punches; and throwing, tend to perform worse than motions that utilise the whole body such as: skier; elbow to knee; and jumping jack exercises. There are exceptions such as walking, although only directly utilising the legs, the leg motion is often accompanied by an arm motion to counter weight the motion of the legs, thus indirectly utilising both the upper and lower body, resulting in time warps achieving good alignments when applied to walk motions. More stylised walks, such as the sneak walk, with room for different interpretations of what the arms poses should be, result in poorer alignments.

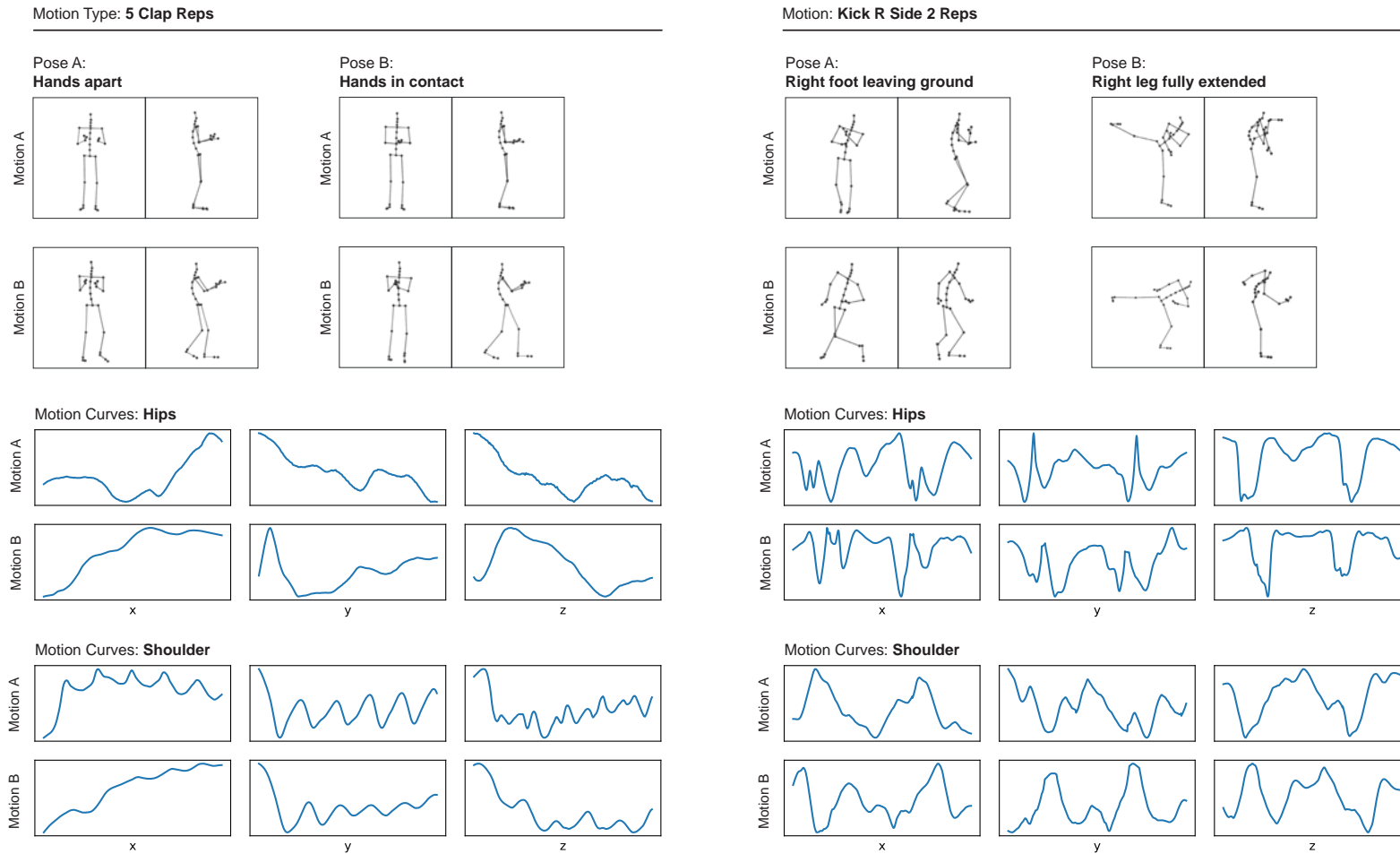


Figure 7.1: Examples of motions which are under constrained. Due to the variety of ways in which the movement in the motion can be achieved, there is more variation between the poses and motion curves of different takes (recordings) of the same motion.

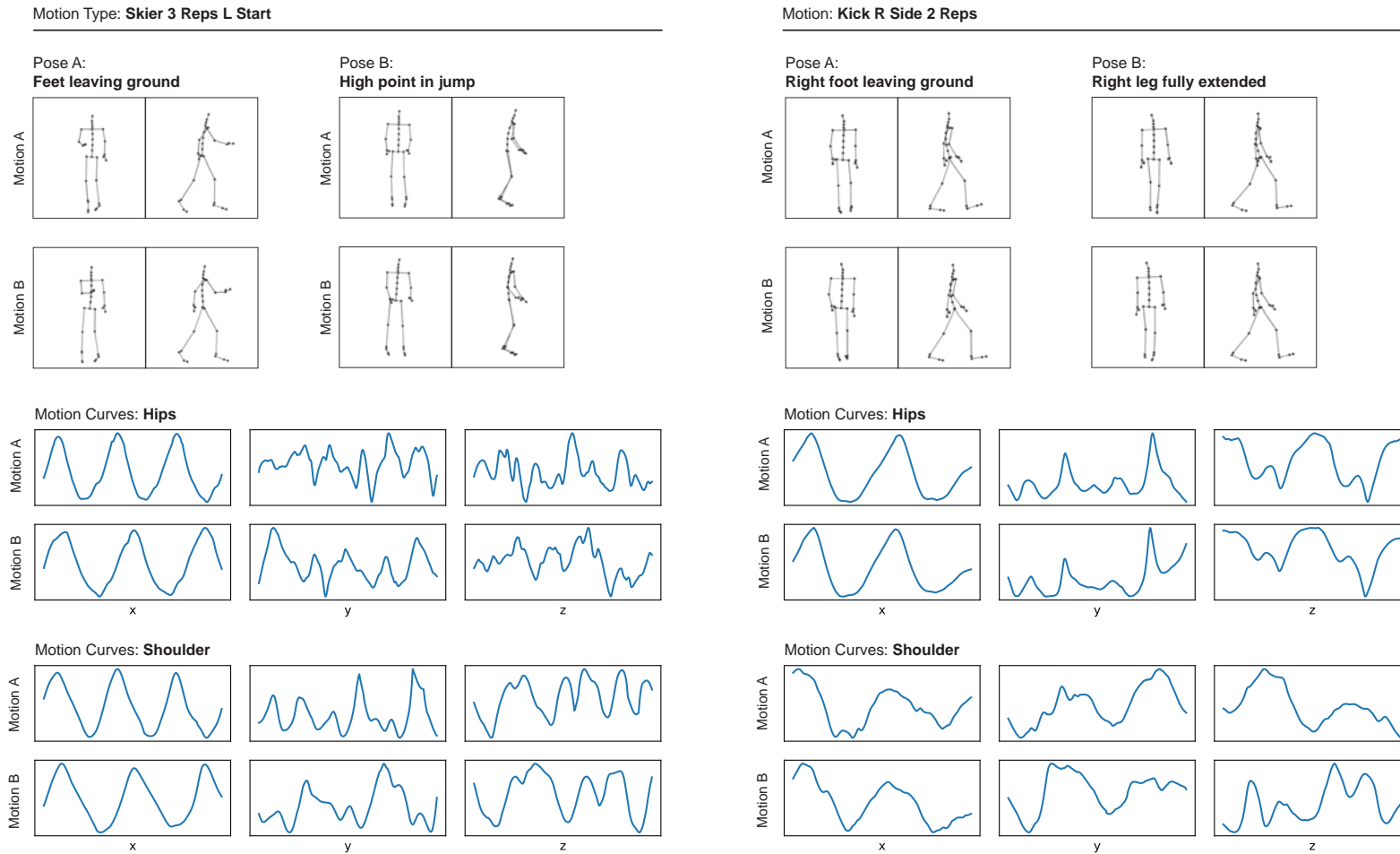


Figure 7.2: Examples of motions which are constrained. Due to the limited ways in which the movement can be achieved, there is less variation between the poses and motion curves of different takes (recordings) of the same motion.

Table 7.1: Table of the mean similarity test scores attained by different on-line time warping algorithms, for each of the different movements used from HDM05 data-set.

Motion	Types of Movement					Mean Similarity Score		
	Cyclic	Non-Cyclic	Ballistic	Lower Body	Upper Body	Ma (+Type V)	Mc w40	Mc w80 (+Type V)
Clap 5 Reps	x				x	0.030	0.044	0.040
Deposit Floor R Hand		x		x	x	0.311	0.318	0.380
Elbow To Knee 3 Reps L Elbow Start	x			x	x	0.326	0.332	0.418
Grab Middle Shelf R		x			x	0.110	0.133	0.126
3 Jumps	x		x	x	x	0.180	0.225	0.240
Hop L Leg 3 Hops	x		x	x	x	0.235	0.218	0.232
Jog Left Circle 6 Steps R Start	x			x		0.340	0.366	0.360
Jog On Place 4 Steps R Start	x			x	x	0.200	0.273	0.260
Jump Down		x	x	x	x	0.229	0.279	0.314
Jumping Jack 3 Reps	x		x	x	x	0.352	0.352	0.390
Kick R Side 2 Reps	x			x		0.128	0.222	0.185
Punch L Side 2 Reps	x				x	0.075	0.108	0.105
Rotate Arms Both Forward 3 Reps	x				x	0.264	0.246	0.257
Shuffle 4 Steps L Start	x			x		0.222	0.289	0.295
Sit DownChair		x		x	x	0.336	0.357	0.380
Sit DownFloor		x		x	x	0.317	0.352	0.342
Skier 3 Reps L Start	x			x	x	0.554	0.559	0.543
Sneak 4 Steps R Start	x			x	x	0.157	0.203	0.187
Staircase Up 3 Rstart	x			x	x	0.327	0.399	0.382
Stand Up Sit Chair		x		x	x	0.146	0.132	0.155
Stand Up Sit Floor		x		x	x	0.221	0.275	0.347
Throw Far R		x	x		x	0.159	0.241	0.268
Turn Right		x		x		0.167	0.218	0.127
Walk 4 Steps L Start	x			x		0.372	0.400	0.392
Walk Left Circle 6 Steps R Start	x			x		0.284	0.370	0.371

In general the pose of the lower body is arguably more constrained than the upper body. In most motions the lower body is supporting the weight of the upper body, and therefore constrained to movements that allow a center of gravity to be maintained.

Related to joint utilisation, are redundant axis, previously discussed in Section 4.3.4. Depending on the method used to encode joint orientations into Euler rotations, some axis of particular joints are not used. These unused axis are referred to as redundant axis and typically reflect the freedom of movement that a particular joint would have in reality. Figures 4.4 shows a knee joint with one degree of freedom and two redundant axis, while Figure 4.5 shows an elbow joint with two degrees of freedom and one redundant axis. These figures also show the impact of the redundant axis when the joint orientation is converted to other rotational parameters.

The issues occurring from under constrained or under utilised joints can be avoided, by clearly defining motions for a subject to perform, in manner which specifies the poses or choreographed movements for both the upper and lower body. For example when a subject is performing a motion while sitting down, only using hands and upper body movement, also specify a set of lower body movements. As Table 7.1 shows, motions which contain both upper and lower body movements align more accurately, therefore, the additional choreographing of lower body movements will improve performance of the time warping algorithm. In some scenarios this approach may be too limiting, preventing free expression or not allowing ideas to be explored.

A potential solution is joint weighting as discussed in Section 3.1.5. This allows joints which are more pertinent to motion to be given more weight, when calculating the cost (or difference) between poses of input and target frames. As well as the optimal weightings already discussed, joints can be automatically weighted based on their range of movement in a given motion (Patrona et al., 2018). For longer motions containing a range of movements, the weighting could be dynamic, changing with each frame or for different segments of the motion. Alternatively multiple takes of a motion could be analysed, to determine the joints exhibiting the greatest variation

between takes, allowing those joints to be given less weighting.

7.3 Data Capture Errors

No motion capture system is completely error free, and these errors will potentially impact the performance of a time warping algorithm. This section explores some of the errors that can occur in the process of capturing and encoding motion data.

7.3.1 Noise

Errors occur in any process that involves sampling and digitizing real world information, such as the pose of an actor. These errors can stem from the accuracy of the measurement instrument(s) being used or in the way the measurements processed and manipulated to produce meaningful information.

In the case of optical motion capture systems, such those used to capture motions for both the BCU and HDM05 data sets used in this study. Multiple cameras are used to capture the position of markers and a best fit location in 3D space is determined for each marker, that best fits the corresponding 2D position of the marker within each camera's view. For this process to work accurately the system needs to be correctly calibrated. Errors in the calibration process, noise from the cameras sensors and rounding errors, which occur in the process of fitting 3D points and encoding rotational data, all contribute errors in the motion data captured using the system. While errors can never be completely illuminated, the desire is reduce the level of error to a point where it has a minimal impact on the motion capture data (i.e. the signal).

An example of a motion with noise can be seen in Figure 7.3. The noise can be seen as small ripples in the motion curve.

Figure 7.4 shows the noise present in the motions sourced from the BCU and HDM05 data sets. The noise was measured by smoothing the motion curves for each joint using a Butterworth filter, configured to a cut off frequency of 8.088Hz following Equation 5.19, then calculating absolute difference between the original and smoothed

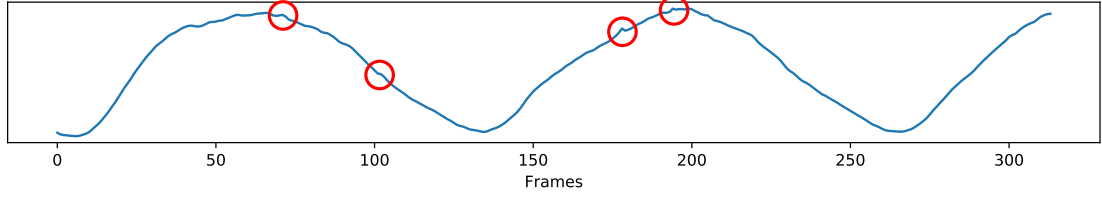


Figure 7.3: An example of noise in a motion signal. Noise artefacts in the motion signal have been highlighted in red.

motion curves using Equation 7.1. n is the noise in a given motion, where a is a motion curve used by the subset of joints used in the study in Chapter 5, where difference between the maximum and minimum value within the motion curve is greater than 0.01, and f is a frame within the motion curve. A more established approach to measuring noise is to use PSNR (Peak Signal to Noise Ratio), in which the amplitude of the noise is expressed as ratio to the peak signal value, however, issues explored later in this section, such as rolling over rotations and changes that can occur mid way through a motion curve, in the way Euler angles are used to express angles, effect the accuracy of the PSNR approach.

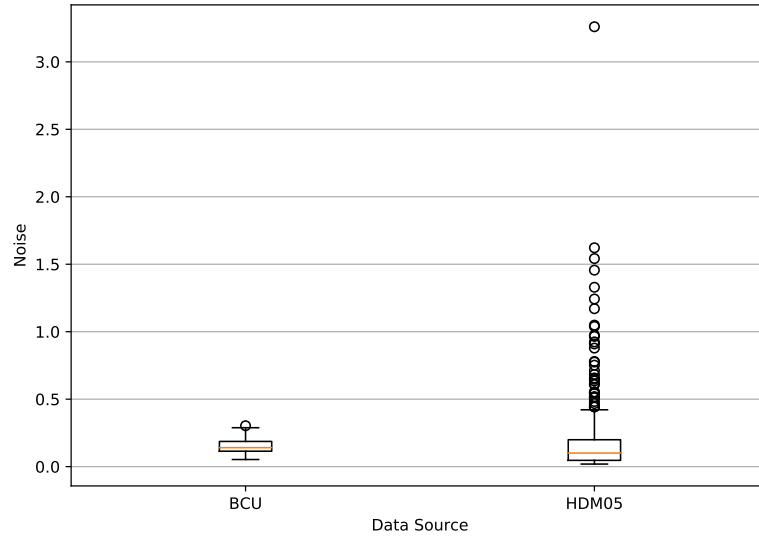


Figure 7.4: The noise present in motions sourced from the BCU and HDM05 data-sets.

$$n = \frac{\sum_{a=1}^k \frac{\sum_{f=1}^r [O_{a,f} - S_{a,f}]}{m}}{k} \quad (7.1)$$

The motions sourced from the HDM05 data-set have a lower median noise level than those sourced from the BCU data-set, however, there are number of outlier motions in the HDM05 data-set which have high levels of noise up to 30 times greater than the median value. Hence, despite having a lower median, the motions sourced from the HDM05 data-set have a higher mean noise levels than those sourced from the BCU data-set, with mean noise levels of 0.207 and 0.157 for motion sourced from the HDM05 and BCU data-sets respectively.

The mean noise level of both data-sets was low, less than a degree of rotation, so it is not anticipated that this would have had much of an impact of on the performance of the time warping algorithms.

7.3.2 Inconsistent Expression of Joint Rotations

Euler angles are an ambiguous method of expressing joint rotations, as the same orientation can be expressed in multiple different ways. Sometimes the manner in which a Euler angles are used to express rotations changes in the middle of a motion.

Changes in the way that Euler angles are expressed can be caused by joints rotating beyond 180° or -180° . Depending on how the motion is being encoded on to the joints, rotations can be allowed to continued beyond these limits, or, in the case of the top plot in Figure 7.5, the rotations instead jump between 180° and -180° , to prevent them from exceeding these limits. If required these rotations can be unrolled to remove the jumps.

Changes in the way three dimensional Euler rotations are expressed, can also occur arbitrarily as shown in bottom plot of Figure 7.5, as the same orientation can be expressed by Euler rotations in a number of different ways. As Euler rotations are realised by rotating around each axis sequentially in a given order, the axis are interdependent, meaning that a change in the way that one axis rotates will result in changes in all the other axis, in order to achieve the same orientation.

The impact of changes in the way Euler rotations are expressed, either between motions or within a motion, can be managed by using a rotational distance function

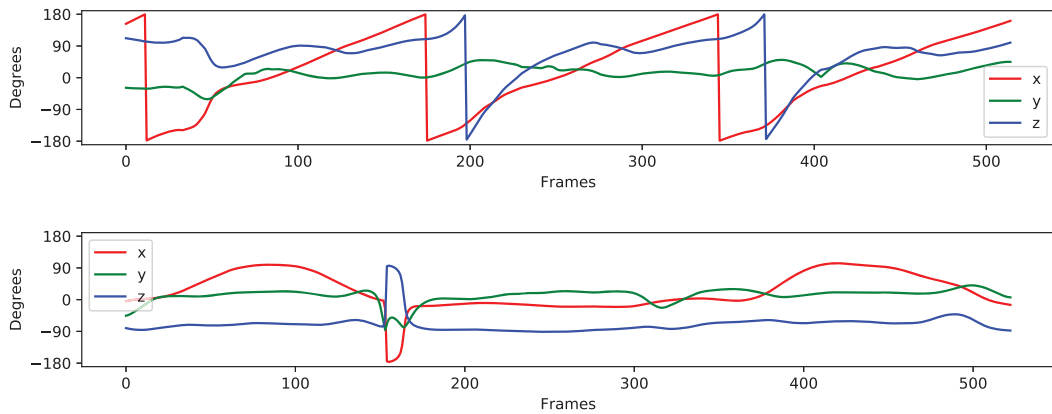


Figure 7.5: Examples of the manner in which a joints orientations are expressed using Euler rotations, changing mid motion. In the top example the x and z axis jump from 180 to -180 as arm rotates in a complete circle. In the bottom example, the way in which the rotations in each axis are used to orientate the joint, appears to change arbitrarily between frames 160 and 175.

based on quaternions rather than Euler angles.

7.3.3 Flat Spots

Flat spots, in which the same joint orientations are written to consecutive frames, can occur in motion capture data. Two examples of flat spots can be seen in Figure 7.6.

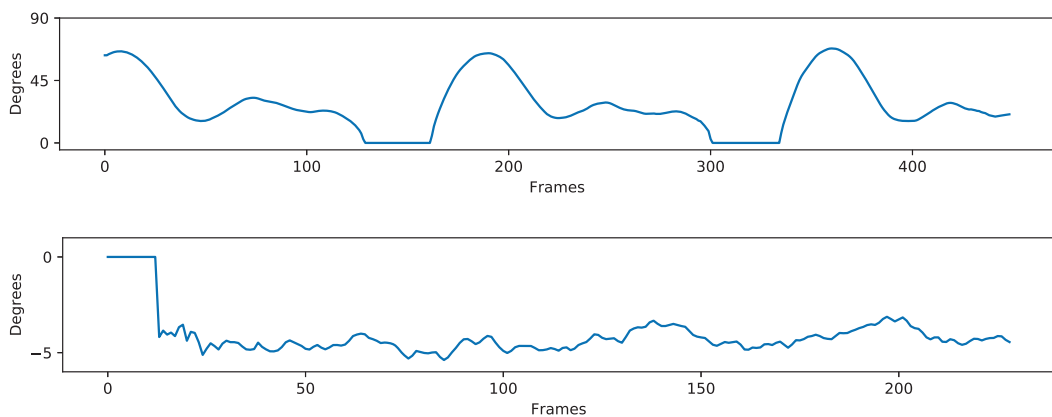


Figure 7.6: Examples of flat spots in motion data. The top example is caused by constraints stopping a knee joint from over extending, the bottom example is a motion with starts with a series dropped frames, which are substituted with zero values.

The flat spots in the top plot of Figure 7.6, are caused by a constraint applied to a

knee joint. To prevent a motion capture system from producing motions that contain implausible joint poses, the range of motion of certain joints can be constrained. A draw back with this approach is that a flat spot occurs when a joints orientation crosses over the limit imposed by the constraint. In the example at the top of Figure 7.6, the extension of a knee joint is not allowed below 0° . This an example of an overly constrained joint, as the Table A.1 shows that knee extensions down to -5.1° are possible. As the movement of knee is restricted, hitting this constraint can also cause foot skate to occur, where the foot appears to slide along the floor.

The flat spot that occurs at the start of the motion curve plotted in the bottom of 7.6, is caused by dropped frames, which occur when motion capture system fails to capture the pose of a joint on particular frame. Dropped frames can occur when there is not enough computing power to maintain the sample rate that the motion capture system has been set to, or when markers are occluded from the view of too many cameras, to be able to determine its position. When dropped frames occur in the middle of a capture, data for those frames is simply not recorded, however, if frames are dropped at the start of a capture, the default (or zeroed out) pose of joint will remain on the first few frames.

Table 7.2 shows the number of flat spots that occur across all the motions sourced from the BCU and HDM05 data-sets, along with the length of the longest flat spot. All of the 57 flat spots that occurred within the motions captured for the BCU data set, occurred on the first two frames and would have a very little impact of on the performance of a time warping algorithm. The small number of flat spots that occurred within the motions sourced from the HDM05 data-set, had longer flat spots, which would potentially impact on the performance of a time warping algorithm.

Table 7.2: The number of times flat spots occur and the length of the longest flat spot, in motions sourced from each data source.

Data Source	Total Frames Across All Motions	No. of Flat Spot Occurrences	Longest Flat Spot (frames)
BCU	59,550	57	2
HDM05	91,138	4	24

7.3.4 Comparison of BCU and HDM05 Data-sets

To identify if the on-line time warping algorithms perform differently when applied to motions sourced from the BCU or HDM05 data sets, the mean similarity test scores of a selection of comparable motions, containing similar movements, where compared in Table 7.3. Four types of motion were compared in Table 7.3 comprising of: jumps; walk; sneaky walk; and transitioning from sitting to walking. In each case the the time warping algorithms performed better when applied to the motions from BCU data set.

Table 7.3: A comparison of the mean similarity test scores attained by on-line time warping algorithms for comparable motions sourced from the BCU and HDM05 data sets.

Source Data-Set	Motion	Ma (+Type V)	Mc w40	Mc w80 (+Type V)
BCU	3 Jumps	0.339	0.443	0.437
HDM05	3 Jumps	0.180	0.225	0.240
BCU	Walk	0.674	0.634	0.525
HDM05	Walk 4 Steps L Start	0.372	0.400	0.392
BCU	Sneaky Style Walk	0.462	0.474	0.430
HDM05	Sneak 4 Steps R Star	0.157	0.203	0.187
BCU	Sit Walk	0.633	0.611	0.581
HDM05	Stand Up Sit Chair	0.146	0.132	0.155

There are two differences between the BCU and HDM05 data-sets, which could potentially explain the differences in the way the time warping algorithms perform, when applied to motions from each source:

- Unlike the BCU data-set the HDM05 data set comprised of motions performed by different actors. All the motions in the BCU data-set where performed by one actor.
- While both data-sets were recorded using Vicon motion capture systems, they are not the same systems. The BCU and HDM05 data-set was created in 2018 and 2005 respectively, on capture system that would would have been representative of the state of motion capture technology at the time. The lower mean noise amplitude and shorter flat spot of the motions sourced from the BCU data-set, are indicative of the improvements in the accuracy of motion

capture solutions between 2005 and 2018.

This section has identified several differences between the HDM05 and BCU data-sets, such as: the amount of noise in each data-set; the occurrence flat spots; and the method of capture. Accurately evaluating the individual impact of each of these differences on the performance of the time warping algorithms, is not possible from these data-sets and would require data-sets specifically designed or synthesised for this purpose.

7.4 Mistimed Starts

7.4.1 Description of Mistimed Starts

Multiple recordings captured of the same motion often start at different points in the motion, resulting in a mistiming between the starting points of the input and target motions being aligned. These mistimed starts can be caused by either the recording of a motion starting early, resulting in a delay in the motion's movement starting within the recording, or the recording starting late, resulting in start of the motion being cropped.

Mistimed starts occur when there is a delay in a performer starting a motion or in the initialising the motion capture process. This results in the start of the performers movement and the capture process not being synchronised.

An example of a mistimed start can be seen in Figure 7.7, which shows the motion curve of the X axis of the right hip and side views of poses taken from the first 150 frames of two recordings of the same **3 Jump** motion containing three jumps. The start of *Jump 01* has been cropped, removing the initiation phase of the first jump, in which the performer crouches down to prepare for a jump. This results in *Jump 01* starting approximately 100 frames 0.833 seconds ahead of the *Jump 02* motion.

This section explores the impact of mistimed starts on the performance of online time warping algorithms, analysing the impact of the start of an input motion being either: ahead of the start of a target motion, referred to as a **leading input** within

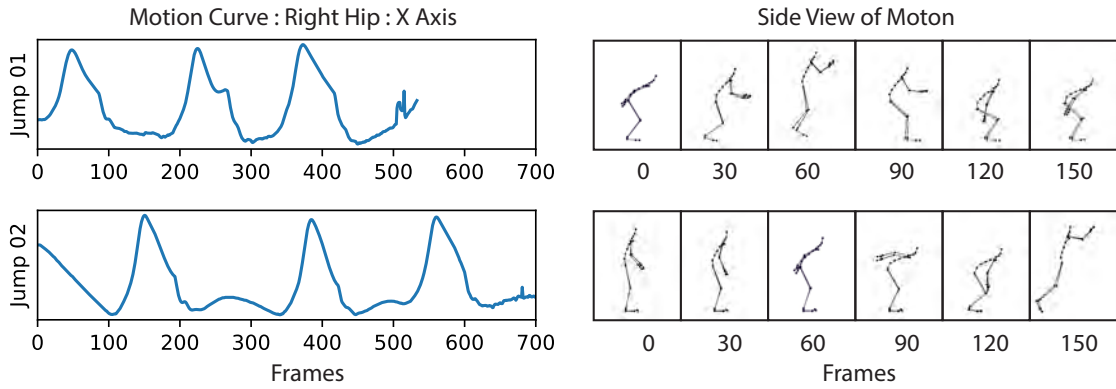


Figure 7.7: An example of mistiming between two jump motions, in which the recordings starts at different points in the motion. The recording of *Jump 01* starts the end initiation phase of the first jump, while the recording of *Jump 02* starts at the beginning of the initiation phase

this study; or behind the start of a target motion, referred to as a **trailing input** within this study.

7.4.2 Identifying and Visualising Mistimed Starts

The alignment of two mistimed motions containing a leading input, results in an alignment path in which the first frame of the input motion is mapped to multiple frames at the start of the target motion. Until the target motion catches up to the input motion, there are no better frames than the first frame of the input motion, to map to the target motion. Examples of this can be seen in Figure 7.8, which contains a number of plots of alignment paths for pairs of motions with a leading input. The alignment path produced by the offline DTW algorithm, shown in light blue, starts with a flat line, mapping the first η number frames of the target motion to the first frame of the input motion. Therefore, the gap between the motions (i.e. the number of frames by which the input motion leads), is equal to η , and a sample pair of motions with a leading input, can be identified if their DTW alignment path has an $\eta > 1$.

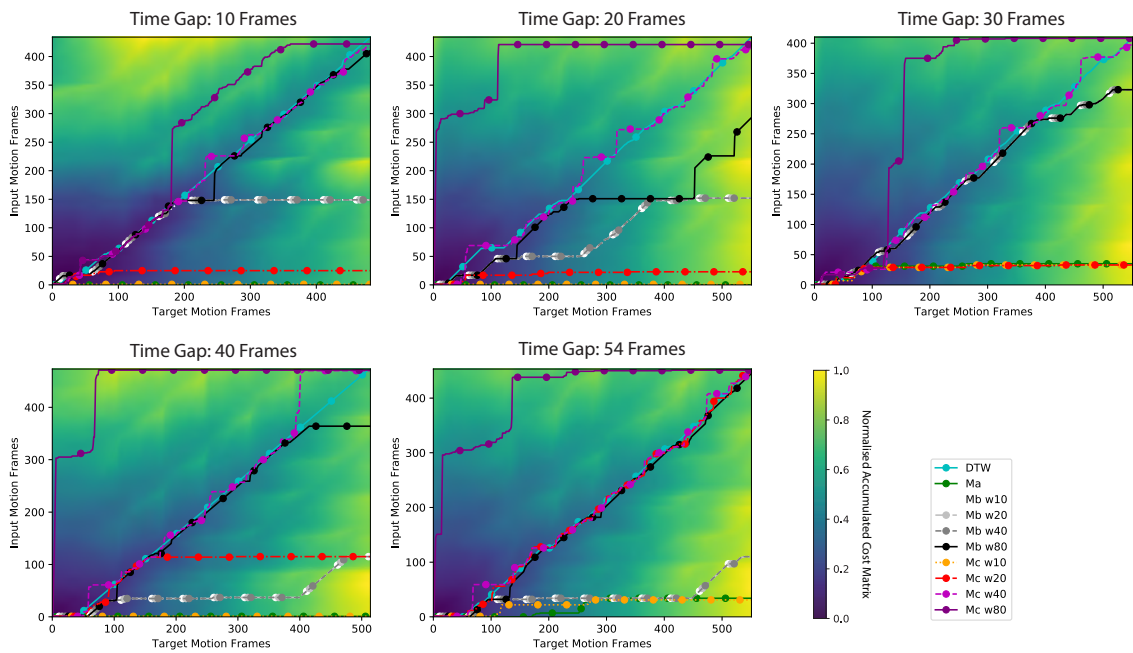
The alignment of two mistimed motions containing a trailing input, results in first η' frames of the input motion being skipped and not mapped to the target motion. As the first few frames of the input motion are outside of the movement contained in

the target motion, they are not required. Examples can be seen in Figure 7.9, which contains a number of plots of alignment paths for pairs of motions with a trailing input. The alignment path produced by the offline DTW algorithm, again shown in light blue, starts at frame η' of the input motion. Therefore, the gap between the motions (i.e. the number of frames by which the input motion trails), is equal to η' , and a sample pair of motions with a trailing input, can be identified if their DTW alignment path has an $\eta' > 1$.

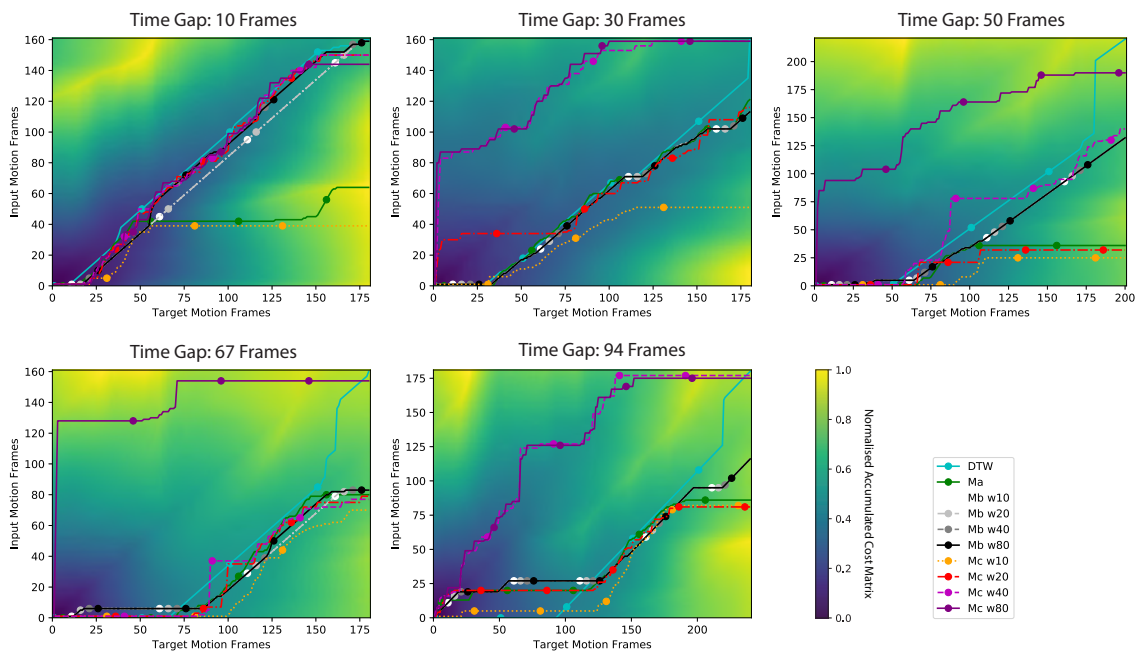
The DTW alignment paths for every sample motion pair within the data set compiled in Chapter 5, were searched to identify motions with leading or trailing inputs. Two motions were identified, **Walk Circle 6 Steps R Start** and **Right Turn**, which have samples with leading and trailing input motions with a range different size gaps of η and η' respectively. Samples containing either of the two identified motions were selected with an approximately evenly distributed gap sizes of η and η' , and plotted in Figures 7.8 and 7.9. It should be noted that each sample plotted in Figure 7.9 contains the same two recordings as the respective sample in Figure 7.8, but with recordings assigned to the input and target motions swapped around to produce an inverted version of the alignment.

Each plot in Figures 7.8 and 7.9 shows the alignment path plotted by each of the online time warping algorithms presented in Chapter 5 as well as that of the offline DTW algorithm, for a given sample pair of motions. The alignment paths have been overlaid over a heat-map of the accumulated cost matrix of the two motions within the sample.

The heat-map visualises how the accumulated cost matrix influences the alignment path determined by each algorithm. The dark blue area of the heat-map indicates a low cost area with minimal difference between the poses input and target frames, the bright yellow areas indicates areas of high cost with a substantial difference between the input and target frames. The alignment algorithms' objective is to plot a path through the dark blue, low cost, areas of the matrix, to determine the most cost effective alignment.

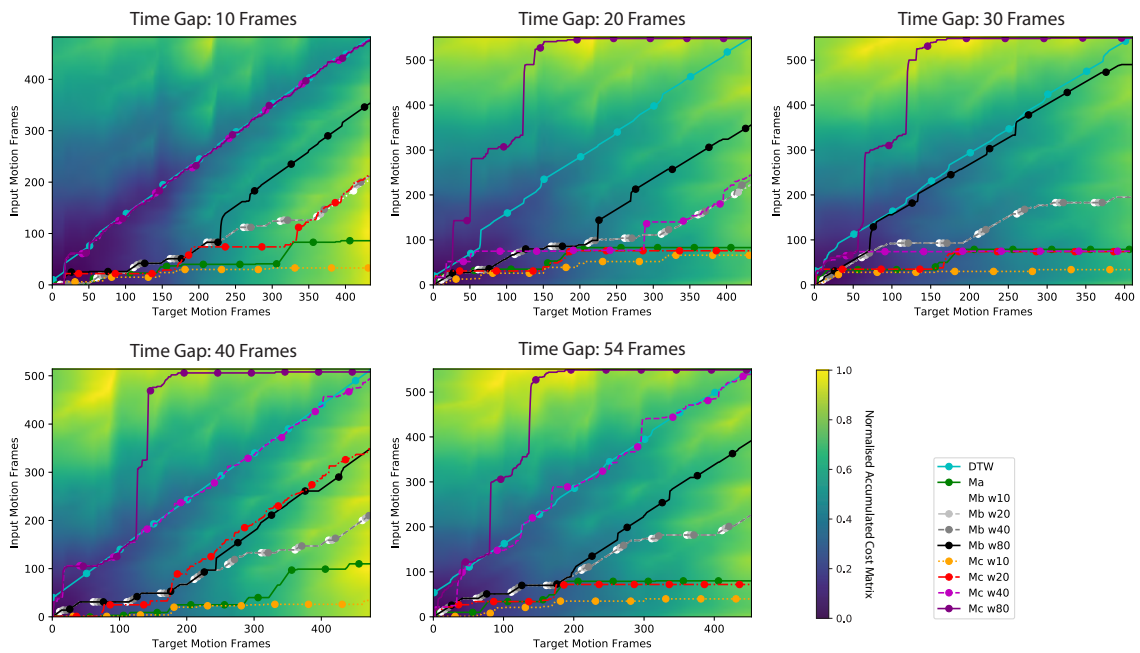


(a) Alignment paths plotted when time warping **Walk Circle 6 Steps R Start** motions.

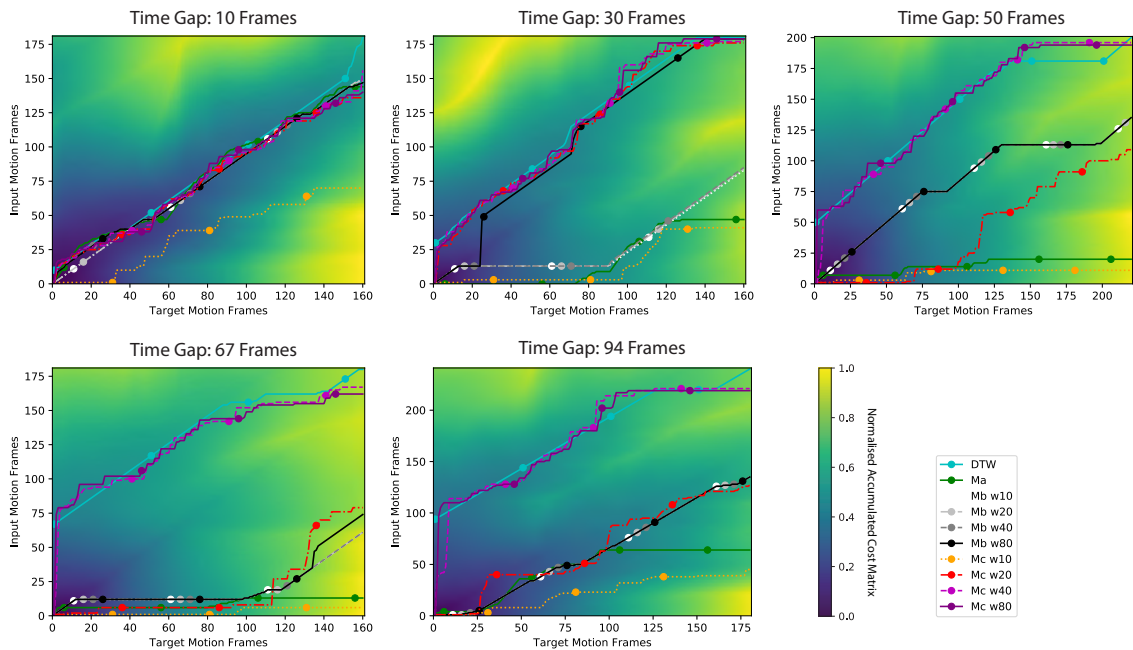


(b) Alignment paths plotted when time warping **Turn Right** motions.

Figure 7.8: Alignment paths plotted by different time warping algorithms, to align a selection of **Walk Circle 6 Steps R Start** and **Turn Right** motions. Every plot has a **leading input** which starts ahead of the target motion by a given number of frames. This should result in an alignment starting with the first frame of the input motion being mapped to multiple frames of the target motion, as the target motion catches up to the input motion. The deviation between the alignment paths produced by each online time warping algorithm and offline DTW, can be seen in Table 7.4.



(a) Alignment paths plotted when time warping **Walk Circle 6 Steps R Start** motions.



(b) Alignment paths plotted when time warping **Turn Right** motions.

Figure 7.9: Alignment paths plotted by different time warping algorithms, to align a selection of **Walk Circle 6 Steps R Start** and **Turn Right** motions. Every plot has a **trailing input** which starts behind the target motion by a given number of frames. This results in an alignment that should start with the first few frames of the input motion being skipped, and not being mapped to the target motion as they are not required. The deviation between the alignment paths produced by each online time warping algorithm and offline DTW, can be seen in Table 7.5.

7.4.3 Evaluation of Alignments

Figures 7.8 and 7.9 show that a mistiming in either direction (i.e. lead or trailing input) can have a negative impact on the performance of an on-line time warp. An online algorithm’s partial knowledge of the motions being aligned, in comparison to the offline DTW algorithm’s complete knowledge of both motions, clearly has a significant impact on the algorithm’s ability to deal with mistimed starts.

Where online algorithms fails to plot a correct alignment path, they fall into their natural bias towards incorrectly skipping or sticking on input frames during alignment, as discussed in Sections 5.4.6 and 5.4.7, with algorithms *Mc w40*, *Mc w80* tending to incorrectly skip frames and algorithms *Ma*, *Mb w10*, *Mb w20*, *Mb w40*, *Mb w80*, *Mc w10* and *Mc w20* tending to incorrectly stick on input frames during alignment.

To provide a more accurate view of the quality of the alignment paths plotted in Figures 7.8 and 7.9, the absolute average deviation in frames, between alignment paths produced by each on-line time warping algorithm and the offline DTW time warping algorithm, in each plot, are shown Tables 7.4 and 7.5. The method for calculating the average frame deviation is described in more detail in Section 5.4.4, and utilises a version of Equation 5.15, without the normalisation based on the length of the motion. The lower the deviation the better the algorithm has performed. Alignment paths with a deviation of less than 18 frames from the path produced by the offline DTW algorithm, are considered good alignments and have been highlighted in green. According to Hoyet, McDonnell and O’Sullivan (2012), timing errors of less than 150ms (18 frames) in interactions between two animated characters are visually imperceptible.

While no one online time warping algorithm successfully plotted a good alignment for every pair of motions, algorithm *Mc w40* was able to plot a good alignment for 17 of the 20 pairs of motions tested, but was only the best performing algorithm for 13 of the 20 pairs of motions. While the *Mc w40* was reasonably consistent at providing a good alignment (85% of motions aligned), it was not as consistent at providing the best alignment (65% of motion aligned).

Table 7.4: The average deviation in frames between alignments produced by various online time warping algorithms and the offline DTW algorithm when applied to **Walk Circle 6 Steps R Start** and **Turn Right** motions, with mistimed **leading inputs** of a given number for frames, using a variety of online time warping algorithms. Green and red are used to indicate if the deviation is below the threshold at which the difference between the online time warp and DTW alignments would be visually perceptible (i.e. below 18 frames). The alignment paths plotted for these time warps can be seen in Figure 7.8.

Motion	Walk Circle 6 Steps R Start					Turn Right				
	10	20	30	40	54	10	30	50	67	94
Ma	200	200	159	217	179	49	4	43	12	19
Mb w10	86	115	19	176	160	12	6	19	16	19
Mb w20	86	115	19	176	160	12	6	19	16	19
Mb w40	86	115	19	176	160	6	6	19	15	19
Mb w80	8	73	19	18	8	6	6	19	15	19
Mc w10	199	199	161	217	176	56	26	52	21	18
Mc w20	177	181	161	126	4	5	14	45	14	22
Mc w40	6	9	8	17	7	4	75	15	13	81
Mc w80	62	196	117	229	211	6	77	90	101	80

Table 7.5: The average deviation in frames between alignments produced by various online time warping algorithms and the offline DTW algorithm when applied to **Walk Circle 6 Steps R Start** and **Turn Right** motions, with mistimed **trailing inputs** of a given number of frames, using a variety of online time warping algorithms. Green and red are used to indicate if the deviation is below the threshold at which the difference between the online time warp and DTW alignments would be visually perceptible (i.e. below 18 frames). The alignment paths plotted for these time warps can be seen in Figure 7.9.

Motion	Walk Circle 6 Steps R Start					Turn Right				
	10	20	30	40	54	10	30	50	67	94
Ma	216	116	237	236	251	4	99	127	125	131
Mb w10	177	96	176	182	192	7	89	60	114	115
Mb w20	176	96	176	182	192	7	89	60	114	115
Mb w40	176	3	176	182	192	6	88	60	114	115
Mb w80	124	3	25	139	144	6	10	60	112	115
Mc w10	235	120	268	260	282	46	102	134	131	154
Mc w20	187	4	239	134	253	7	5	100	113	110
Mc w40	4	3	225	7	12	6	5	7	7	9
Mc w80	4	3	148	118	131	6	5	7	8	7

The next best performing algorithm was *Mb w80*, which plotted a good alignment for 9 of the 20 pairs of motions. This is large drop from the success rate of *Mc w40*, however, for the three motion pairs that *Mc w40* was not able to align, *Mb w80* either achieved or came close to achieving a good alignment with deviations of 6, 19 and 25, while *Mc w40* deviated by 75, 81 and 225 when aligning the same respective motions. This demonstrates that the *Mb w80* algorithm is a good complement to *Mc w40* performing well where the other algorithm is weakest. The manner in which the *Mb w80* and *Mc w40* algorithms complement each other stems from the contrasting approaches taken by Methods B and C. As described in more detail in Chapter 5, Method C achieves better alignments at a lower window size by using a less constrained approach, determining the which input frame to map to a given target frame, by selecting an input frame directly from an accumulated cost matrix, while Method B takes a more constrained approach by plotting a path through the accumulated cost matrix.

Why "Comparison to UTW" Benchmark Is Unsuitable

Within this study the performance of on-line time warping algorithms have been compared to two benchmarks. First, is the similarity score achieved by the algorithm higher than that achieved by offline UTW algorithm. Second, is the average deviation between alignment path plotted by the algorithm and that plotted the offline DTW algorithm, less than 18 frames. The second benchmark is utilised in Tables 7.4 and 7.5, as the first, UTW based benchmark, is unreliable when applied to assessing alignments between pairs of motions which start with large mistiming errors.

As the UTW algorithm aligns based on a straight diagonal alignment path from frame (0,0) to frame (m, n), when aligning an input motion of m frames to a target motion of n frames. This diagonal path becomes a less accurate benchmark to measure against, the further a mistiming error requires a correct alignment to move away from it. Using Table 7.5 as an example, when applied to motion pairs starting with a mistimed trailing input of 10 frames, the UTW benchmark identifies the same set algorithms as producing a good alignment, as the frame deviation benchmark,

for both **Walk Circle 6 Steps R Start** and **Turn Right** motions. As the the size of the mistimed trailing input increases the similarity score achieved by the UTW alignment becomes very low, sitting an extremely low benchmark that can be easily satisfied by every time warping algorithm.

The similarity score attained by an alignment using the UTW algorithm for a pair of **Walk Circle 6 Steps R Start** motions starting with a mistimed **trailing input** of 54 frames was -0.012, while a UTW aligned pair of **Right Turn** motions starting with a mistimed **trailing input** of 94 frames, scored -0.024. Both of these scores are easily beaten by the alignments produced by all of the time warping algorithms tested, despite the most of them clearly producing unsatisfactory alignments in the respective plots in Figure 7.9.

A comparison between the good alignments identified in Tables 7.4 and 7.5, using the second benchmark based on the deviation from the offline DTW alignment, and the visualisations of the same alignments in plotted Figures 7.8 and 7.9, shows that this is a reliable method of identifying good alignments.

7.4.4 Relationship Between Size of Mistiming and Window Size

The alignments plotted for the **Turn Right** motion show that time warping algorithms that tend to skip input frames incorrectly, perform better at aligning pairs of motions with a trailing input, as they require a certain number of frames to be skipped at the start of the alignment. Likewise, the same alignments also show that time warping algorithms that tend to stick on input frames incorrectly, perform better at aligning pairs of motions with a leading input, as they require the alignment to stick on the first frame of the input motion at the start of the alignment. While both trends is to be expected, it is however not consistent, both Figures 7.8 and 7.9 show the **Walk Circle 6 Steps R start** motion not following these trends as closely as the **Turn Right** motion.

The frame deviation results in Table 7.5, shows that pairs of motions which start with bigger **trailing input** mistiming errors, require time warping algorithms with larger

forecast windows to successfully align them. This trend is more clearly demonstrated within the results of **Right Turn** motion in Figure 7.9b, where the motions with smallest mistiming error of 10 frames, can be successfully aligned by all but one of the time warping algorithms, while the motions with three largest mistiming errors of 50, 67 and 94 respectively can only be solved by Method C algorithms with largest window sizes $Mc\ w40$ and $Mc\ w80$.

The trend of the performance of particular time warping algorithms dropping, as the magnitude of the mistiming at the start of the motions gets larger, is not as apparent with pairs of motions with **leading input** mistimed starts. The changes in the amount of deviation (or error) in Table 7.4, between pairs of motions with a smallest and largest mistimed starts, is less dramatic than in Table 7.5.

This divergence between motions with **leading** and **trailing** mistimed starts, in the way the performance of online time warping algorithms change as the magnitude of the mistimed start increases, is a product of the use of the windowing in the online time warping algorithms evaluated in this study.

Aligning pairs of motions that start with mistimed **leading inputs**, requires a time warping algorithm to map the first η frames of the target motion, to first frame of the input motion, effectively sticking the alignment path to a single input frame. Without any local continuity constraints applied, all time warping algorithms are capable of sticking on an single input frame and aligning any number of target frames to it. This is not the case, however, with pairs of motions that start with mistimed **trailing inputs**, as these require an alignment path that starts by skipping input frames, and the number of input frames that can be skipped between two consecutive alignments points (i.e. two consecutive target frames), is limited by the windows size of the time warping algorithm (i.e algorithm $Mc\ w10$ can only skip up to 10 input frames between alignment points). This would explain why time warping algorithms with smaller window sizes are able align pairs of motion with large **leading input** mistimed starts, but not those that start with large mistimed **trailing inputs**.

This points to a weakness in the general approach used by the on-line time warping algorithms used in this study. As they are ran once for every target frame, in order

to determine the best input frame to align each target frame to, there is no limit to how many target frames can be mapped to a single input frame, while there is a limit to how many input frames can be skipped between two consecutive target frames. This is a limitation imposed by using a forecasting window and explains the bias the online time algorithms showed towards sticking on input frames within the aggregate plots in Section 5.4.6. This bias can be managed with approaches such as, using the forecasting window in manner that potentially allows for more input frame skipping as shown in with Method C or local continuity constraints as described in Chapter 6.

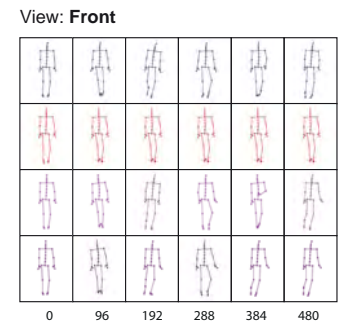
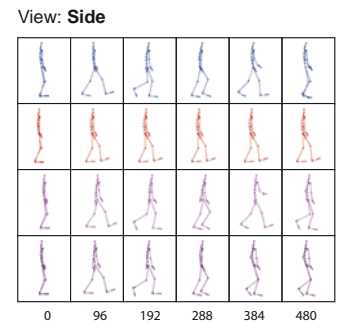
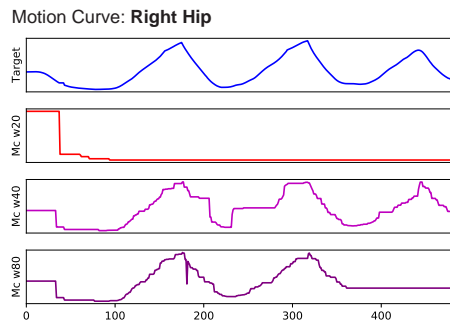
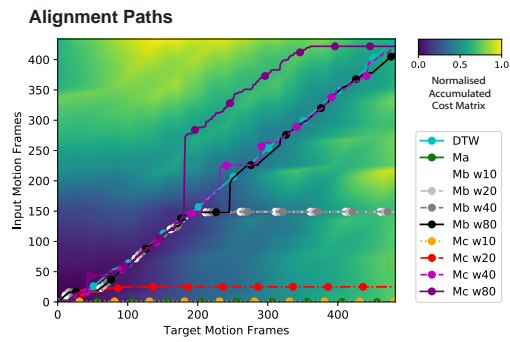
7.4.5 Impact on Motion and Alignment Quality

Figures 7.10 and 7.11 show a more detailed visualisation of the impact of mistimed starts, with **leading** and **trailing** inputs respectively, on alignment accuracy and motion quality. For each sample pair of motions, the alignment path, motion curves and character poses from selected frames have been plotted, for different time warping algorithms, in the same manner as the visualisations in Section 5.4.7. Videos of the aligned motions have been rendered and can be located and downloaded from [Randall \(2022b\)](#) using the sample number.

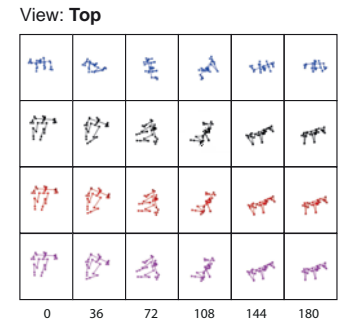
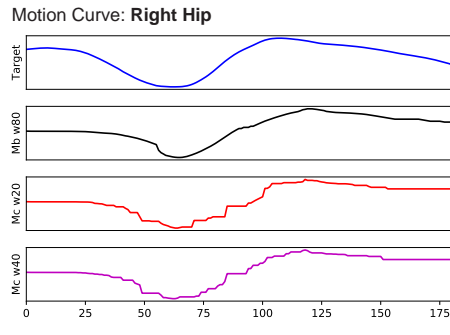
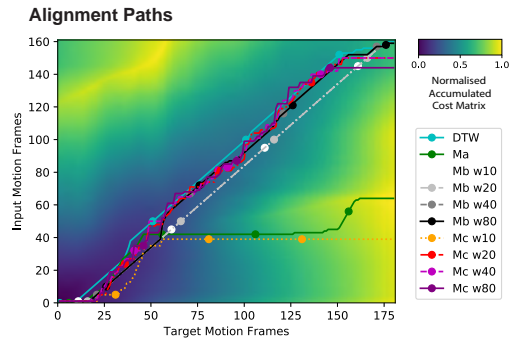
The contrasting forms of the **Turn Right** and **Walk Circle 6 Steps R Start** motions curves can be clearly seen, with the cyclic **Walk Circle 6 Steps R Start** motion containing repetitive peaks and troughs.

Poor alignments show up as flat lines in the motion curves and repeated character poses in consecutive frames. Figure 7.11 shows all the Method B algorithms along with algorithms *Ma*, *Mc w10* and *Mc w20*, struggling to align pairs of motions with a mistimed **trailing input** starts, with flat lines occurring at various points in the alignment. Sample number 2795 in Figure 7.10 and Sample number 3196 in Figure 7.11, both show mistimed starts causing algorithm *Mc w80* to skip frames incorrectly at the start of the alignment. In both cases this causes the alignment to run out of input frames early in the alignment, resulting in a flat line at the end of the motion curve, where many multiple target frames have to be aligned to the last

Motion: Walk Circle 6 Steps R Start
 Gap: 10
 Sample No: 3239



Motion: Turn Right
 Gap: 10
 Sample No: 2897



Motion: Turn Right
 Gap: 67
 Sample No: 2795

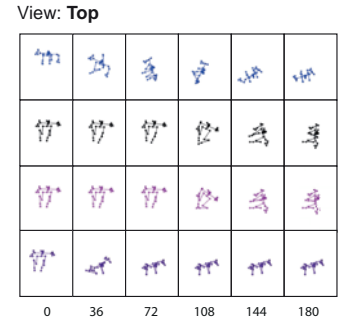
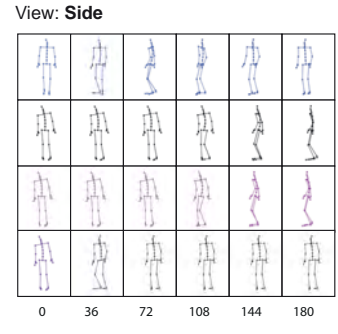
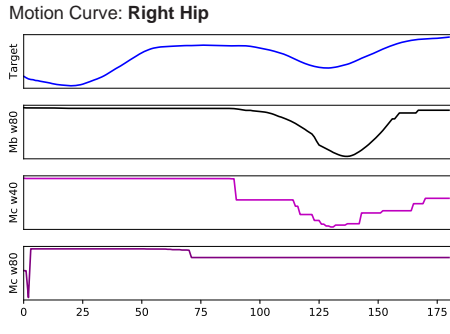
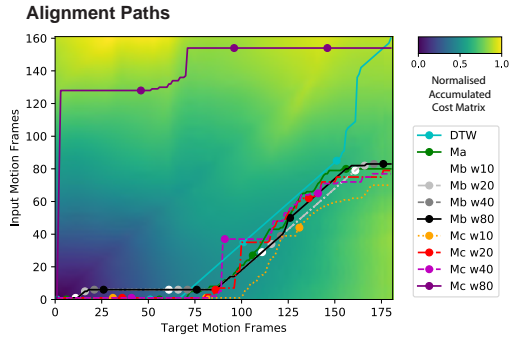


Figure 7.10: A detailed presentation of the alignments produced by online time warping algorithms, when used to align motions with a **leading input** of a given number of frames.

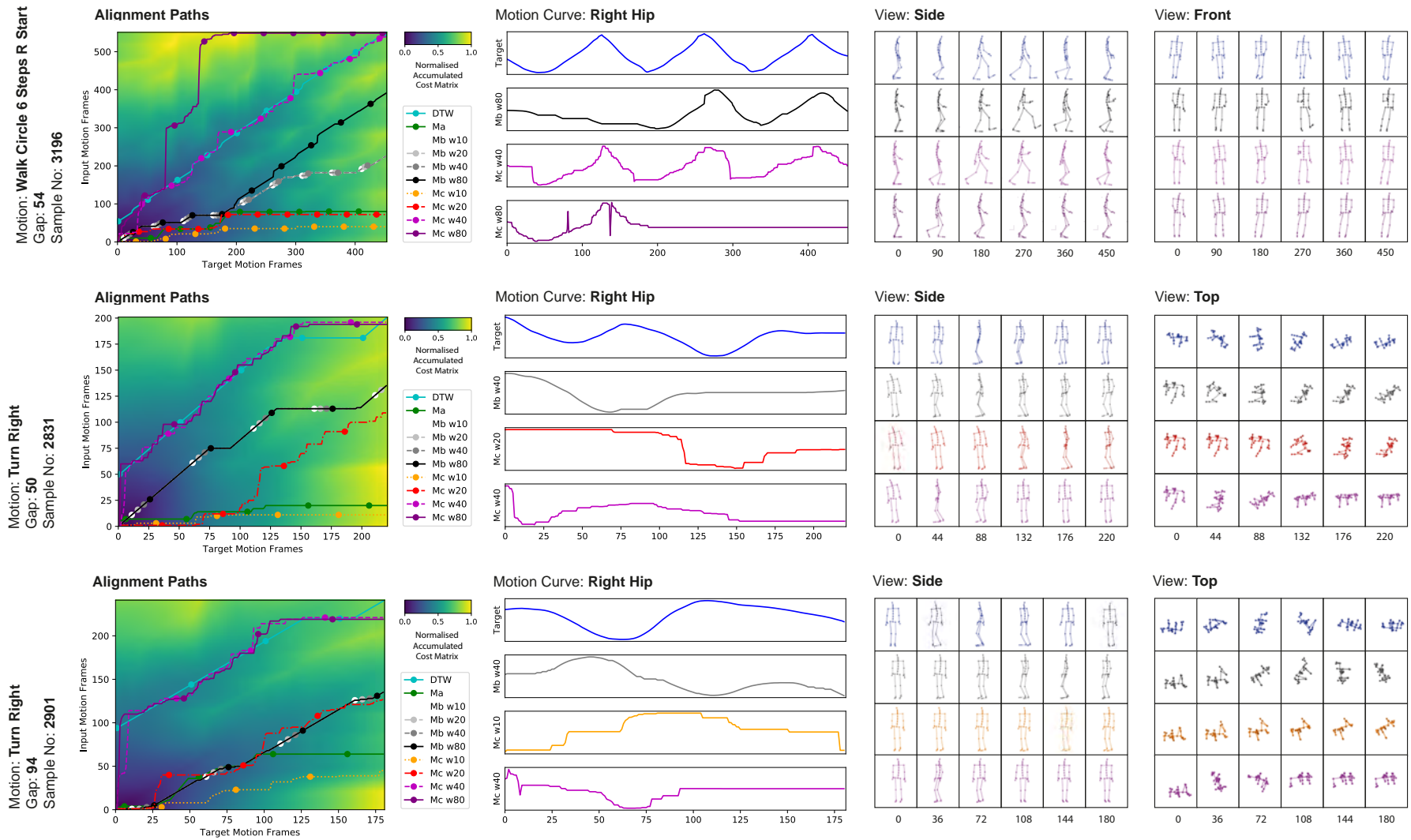


Figure 7.11: A detailed presentation of the alignments produced by online time warping algorithms, when used to align motions with a **trailing input** of a given number of frames.

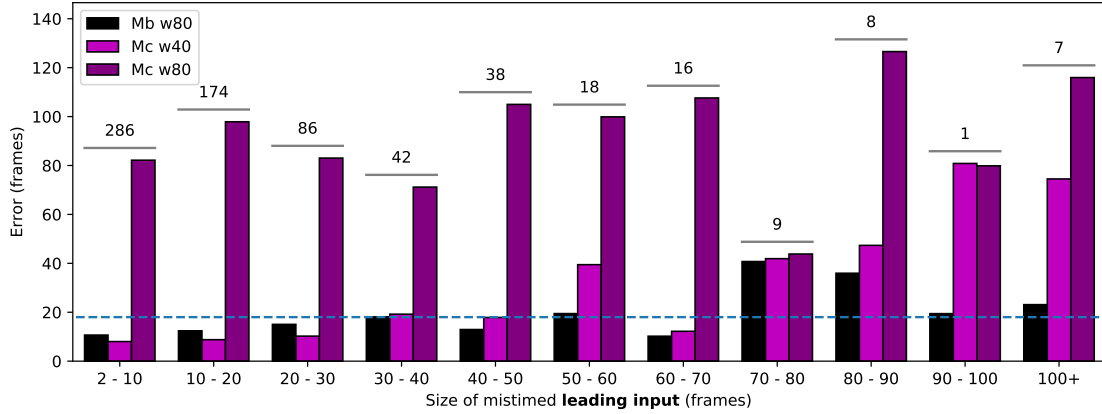
input frame.

7.4.6 Tolerance of On-line Warping Algorithms for Mistimed Starts

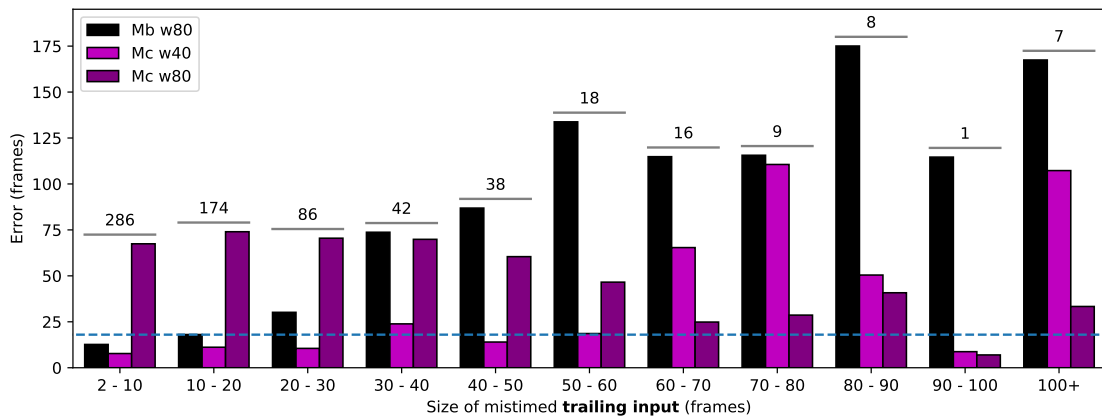
For real-world applications it is important to consider how large a mistiming gap, between the starting points of an input and target motion, can an on-line time warping algorithm tolerate and still produce useful alignments. To determine this, all the pairs of motions in the data-set established in chapter 5, were group based in the alignment plotted for them by the offline DTW time warping algorithm. The pairs were group together based on the type (**leading** or **trailing** input) and magnitude of the mistiming error (i.e. the gap, in frames, between the start of the input and target motions). For each time warping algorithm the median performing sample of motion pairs within each group was charted in Figure 7.12. In line with Section 7.4.3, the median performing sample and subsequent Error values in Figure 7.12, are based on the average deviation of the sample alignment from the DTW alignment, in frames. The median rather than the mean was chosen as this is less likely to be affected by outlier results. The number above each group indicates the number of samples within the group, the curation of the data set in Chapter 5, means that the number of samples that start with larger mistiming gaps is small.

Figures 7.12a and 7.12b show the capability of different on-line time warping algorithms to align pairs of motions with mistimed start of various sizes, with **leading inputs** and **trailing inputs** respectively. The error values can be assessed against the previously established benchmark of 18 frames, shown as a dash blue line in each graph.

There is some contrast between which online time warping algorithms cope well with different types of mistimed starts. According to Figure 7.12a, the most capable algorithm for aligning pairs of motions starting with mistimed **leading inputs** is algorithm *Ma w80*, which is able to cope timing gaps of up to 50 frames or 416ms. Meanwhile, Figure 7.12b, shows that the most capable algorithm for aligning pairs of motions starting with mistimed **trailing inputs** is algorithm *Mc w40*, which is



(a) The performance of on-line time warping algorithms when aligning pairs of motions starting with mistimed **leading inputs** of various sizes.



(b) The performance of on-line time warping algorithms when aligning pairs of motions starting with mistimed **trailing inputs** of various sizes.

Figure 7.12: Charts showing the capability of different on-line time warping algorithms at aligning pairs of motions starting with mistimed **leading inputs** and **trailing inputs** of various sizes. Each bar represents the median sample from each group, the **Error** is the average number of frames by which the sample alignment deviates from the offline DTW alignment. The dashed blue line represents the threshold at which timing errors in character interactions can be perceived (Hoyet, McDonnell and O’Sullivan, 2012). The numbers above each group of bars indicates the number of alignments the results are based on.

able to cope timing gaps of up to 30 frames or 250ms.

The on-line time warping algorithm that performs the most consistently across both types of mistimed starts is *Mc w40*, which is able to align motions with mistimed starts of either type to up to 30 frames, to below the benchmark error level. For both types of mistiming error the *Mc w40* algorithm dips in performance at 30 - 40 frames, with results below the benchmark, then improves to beat the benchmark at 40 - 50 frames. Beyond 50 frames the *Mc w40* algorithm does beat the benchmark in places but this is patchy. It should be noted that the small number of samples with larger mistiming gaps, means that data is not as reliable beyond 50 frames.

There is a difference in the performances of *Mb w80* and *Mc w80*, between aligning pairs of motions with mistimed **leading** or **trailing** inputs. While *Mb w80* is able to align motions with a **leading input** mistiming of up to 50 frames, to the benchmark standard, it is only able to align motions with a **trailing input** mistiming of up to 20 frames. As the size of the mistimed start increases, the difference in the performance of *Mb w80* between aligning motions with mistimed **leading** or **trailing** inputs increases, with a median Errors of 23 and 167 frames respectively, for mistimed starts of above 100 frames. Interestingly the performance of the *Mb w80* algorithm is not as affected by the magnitude of the **leading input** as the *Mc w40* algorithm, maintaining a more consistent performance across a mistimed starts of a range of sizes. The *Mb w80* algorithm's capability to align motions with mistimed **leading input** starts, stems from its tendency to stick on frames, as discussed in Section 7.4.4. Therefore, this capability does not extend to aligning motions with mistimed **trailing starts** which require input frames to be skipped at the start of the alignment.

The *Mc w80* algorithms tendency to skip input frames during alignment, caused it to have a contrasting performance to that of *Mb w80*. The *Mc w80* algorithm generally performs poorly, only beating the DTW alignment deviation benchmark in the "90-100 **trailing input**" group which contained only one sample. While it performed consistently poorly at aligning motions with mistimed **leading input**

starts of various magnitudes, when applied to motions with mistimed **trailing input** starts, its performance improves as the magnitude of the mistiming increases, performing better than the *Mb w80* and *Mc w40* algorithms at aligning motions which start with a mistimed **trailing input** of greater than 60 frames.

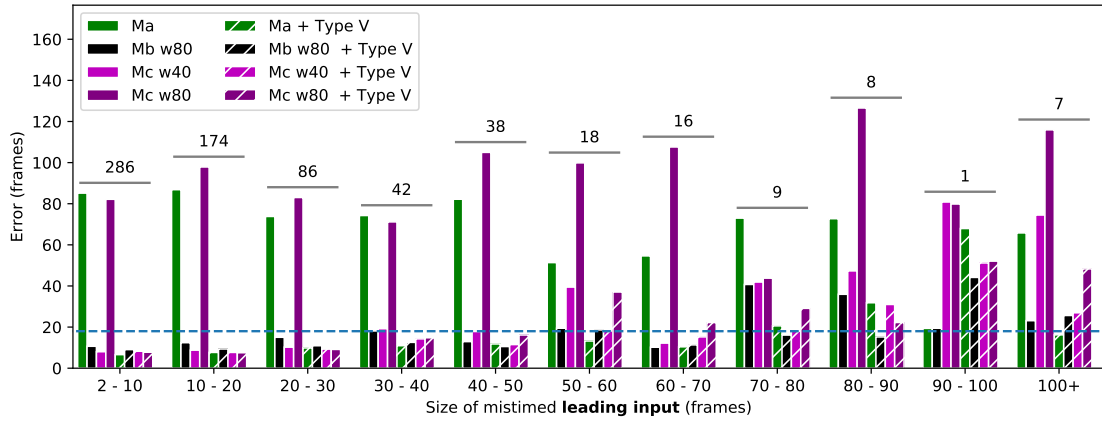
7.4.7 Impact of Constraints on Mistimed Starts

Local constraints, as presented in Chapter 6, can potentially improve the ability of the time warping algorithms to cope with mistimed starts. To explore this the performance of selected time warping algorithms, with and without the Type V local constraint implemented, were compared in Figure 7.13, when applied to both **leading input** and **trailing input** mistimed starts of varying magnitude. The time warping algorithms plotted were selected as they were either shown to be the best performing algorithms in Chapter 5 or were their performance was most improved by the use of a constraint in Chapter 6. The chart plots an Error value for each time warping algorithm at different magnitudes, which is determined in the same manner as that in Figure 7.12.

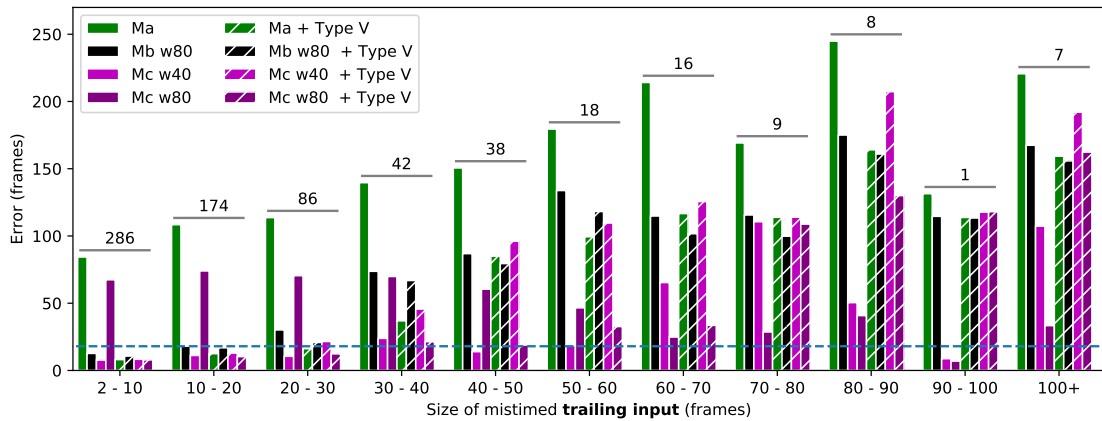
Figures 7.14 and 7.15, show a more detailed visualisation of the impact of constraints on the ability of online time warping algorithms to cope with mistimed starts. The figures show the alignment paths plotted by time warping algorithms with and without constraints implemented on the left and right respectively, for selected sample motion pairs. To help visualise the impact of the alignment on the quality of the motion, motion curves have also been incorporated.

Impact of Constraints on Aligning Leading Input Verses Trailing Input Mistimed Starts

Figure 7.13 shows that constraint had a larger positive impact on time warping algorithms, when they were applied to pairs of motions with mistimed **leading inputs** with average Error reduction of 133 frames across the selected algorithms, however, when applied to motion pairs with **trailing inputs**, a smaller average Error reduction of 16 frames was achieved by the same algorithms, with some algorithms



(a) The performance of on-line time warping algorithms, with and without local constraints applied, when aligning pairs of motions starting with mistimed **leading inputs** of various sizes.



(b) The performance of on-line time warping algorithms, with and without local constraints applied, when aligning pairs of motions starting with mistimed **trailing inputs** of various sizes.

Figure 7.13: Charts showing the capability of different on-line time warping algorithms, with and without local constraints applied, at aligning pairs of motions starting with mistimed **leading inputs** and **trailing inputs** of various sizes. Each bar represents the median sample from each group, the **Error** is the average number of frames by which the sample alignment deviates from the offline DTW alignment. Bars with a hash pattern, represent time warping algorithms with a the Type V constraint applied. The dashed blue line represents the threshold at which timing errors in character interactions can be perceived. The numbers above each group of bars indicates the number of alignments the results are based on.

performing worse. With the constraint implemented, the median Error for algorithms *Ma*, *Mb w80*, *Mc w40* and *Mc W80*, reduced by 48, 3, 14, 68, respectively when applied to pairs of motions with **leading inputs**. In comparison, when applied to pairs of motions with **trailing inputs**, the constraint only produced an average improvement in the performance of algorithms *Ma* and *Mb w80*, reducing the Error by 76 and 9 frames respectively, with the Error increasing for remaining two algorithms *Mc w40* and *Mc w80*, by 57 and 12 frames respectively.

For time warps applied to mistimed **leading inputs**, Figure 7.13a shows the constraint improving the performance of every algorithm up to a mistiming magnitude of 60 frames, with every algorithm meeting the Error level benchmark for mistimings of up to 50 frames. In comparison, when applied to pairs of motions with mistimed **trailing inputs**, Figure 7.13b shows the Type V constraint produced no improvement in the *Mc 40* algorithm at any magnitude, and all the constrained algorithms only met the benchmark when applied to a mistiming errors of up to 20 frames.

The implementation of the constraint tends to be more effective at correcting alignment paths that incorrectly skip frames than those that incorrectly stick on frames. In Figures 7.14 and 7.15, Samples 2897, 2795 and 3239 all show the incorrect skips in the alignment path plotted by algorithm *Mc w80* being fully resolved or reduced by the implementation of the constraint. In contrast Samples 3196 and 2831 in Figure 7.15, both show the frame sticking in the alignment paths plotted by algorithms *Mc w10* and *Ma* only being partially resolved, with the algorithms still only plotting the shallowest alignment path allowed by the constraint. Apart from algorithm *Ma*, in Samples 3239 and 2897 in Figure 7.14, alignment paths that incorrectly stick on input frames, are only partially resolved when a constraint is incorporated into the time warping algorithm. In case of sample 2897 the incorrect sticking within the alignment plotted by algorithm *Mc w10* is made worse when the constraint is applied to the algorithm.

The effectiveness of constraints at preventing alignment paths from incorrectly skipping input frames, in comparison to preventing them from incorrectly sticking on an input frame, is supported by the findings in Chapter 6 and the aggregate plots in

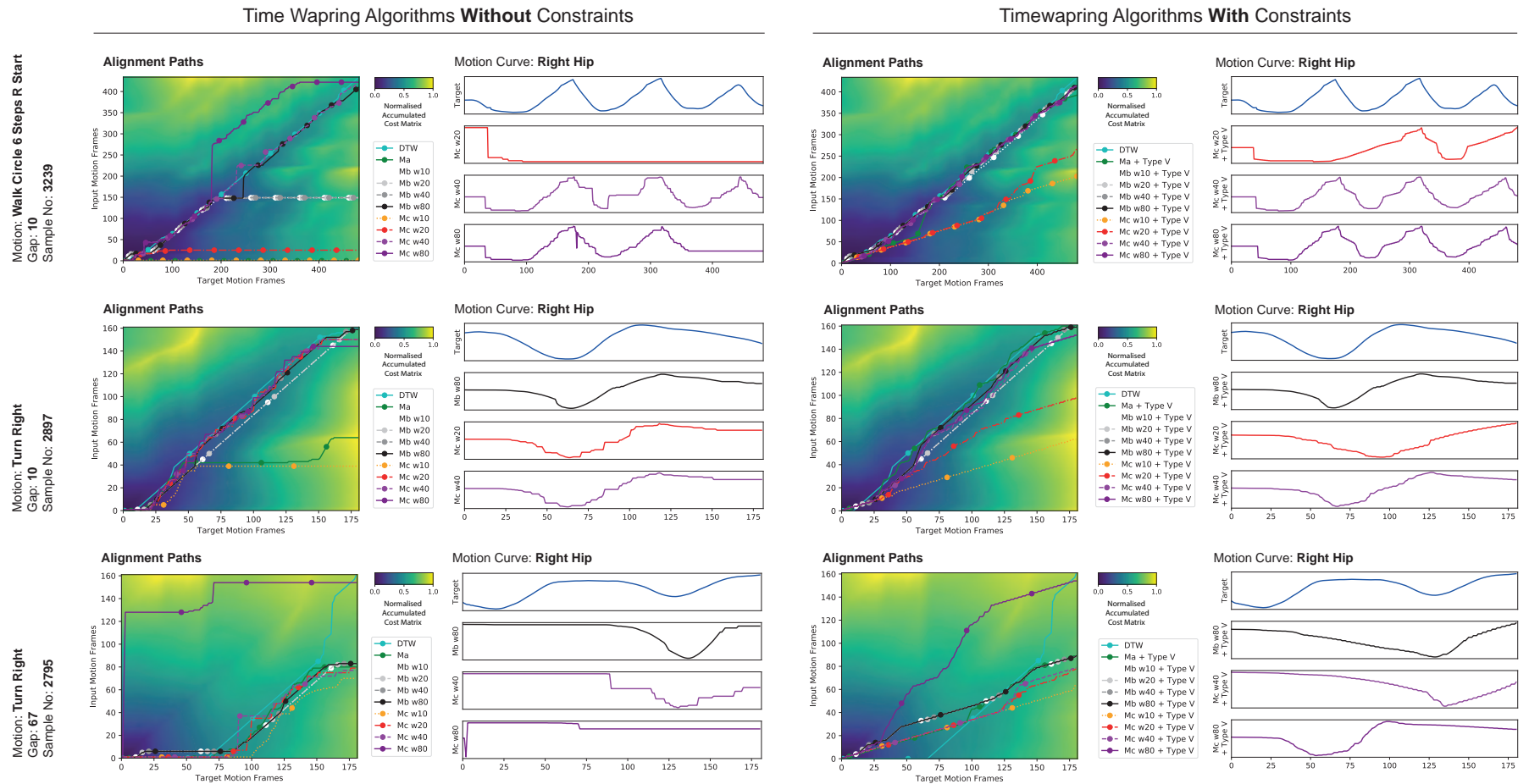
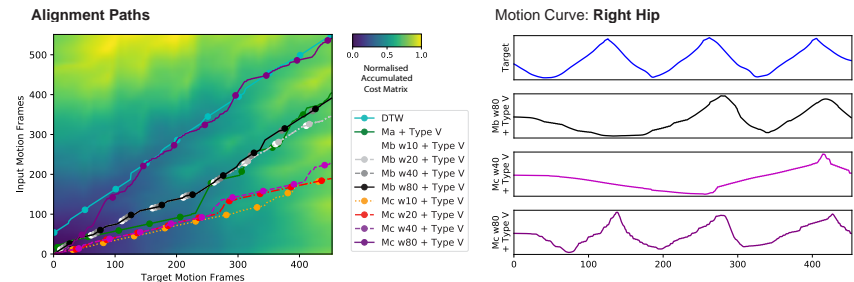
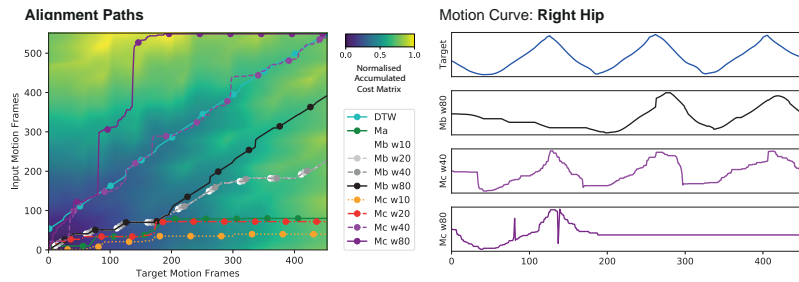


Figure 7.14: A detailed presentation of alignments produced by online time warping algorithms, with (right) and without (left) constraints implemented, when used to align motions with a **leading input** of a given number of frames.

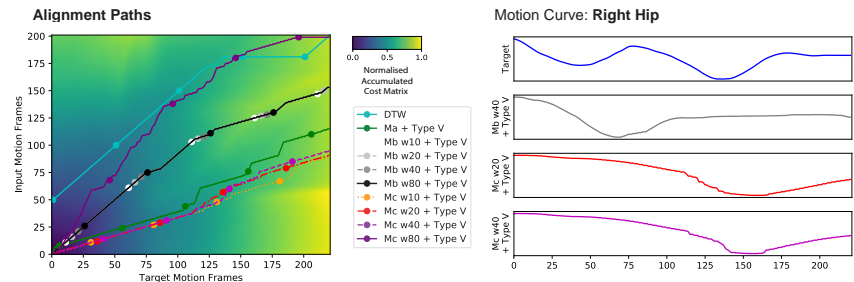
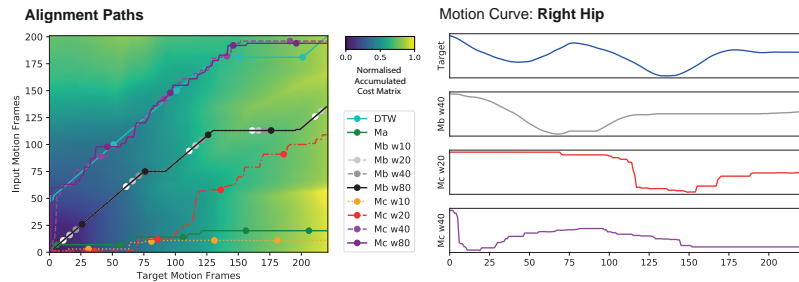
Time Warping Algorithms **Without** Constraints

Timewarping Algorithms **With** Constraints

Motion: Walk Circle 6 Steps R Start
Gap: 54
Sample No: 3196



Motion: Turn Right
Gap: 50
Sample No: 2831



Motion: Turn Right
Gap: 94
Sample No: 2901

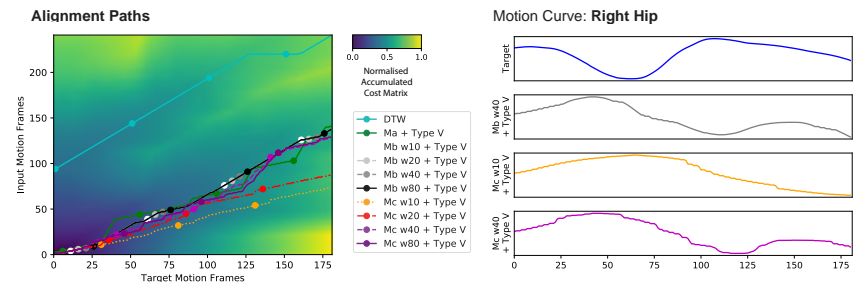
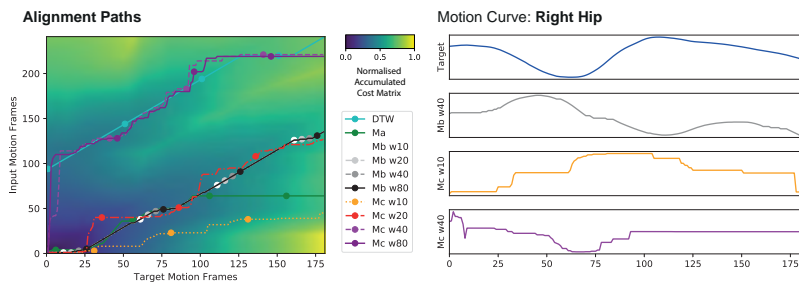


Figure 7.15: A detailed presentation of the alignments produced by online time warping algorithms, with (right) and without (left) constraints implemented, when used to align motions with a **trailing input** of a given number of frames.

Figure 6.6, comparing the alignment paths plotted by time warping algorithms with and without the implementation of constraints. It also explains why the implementation of constraints produced more improvements, in the results of time warping algorithms when applied to pairs of motions with mistimed **leading inputs** than mistimed **trailing inputs**. Mistimed **leading input** starts, require the alignment path to start by sticking on the first input frame, so improvements that reduce the tendency of an algorithm to incorrectly skip frames will benefit this use case.

Impact of Constraints on Aligning Mistimed Starts of Larger Magnitudes

When applied to pairs of motions with mistimed starts of larger magnitudes, the impact of implementing constraints is significantly different between motion pairs with **leading inputs**, Figure 7.13a, and **trailing inputs**, Figure 7.13b. For example with mistimed Errors of over 100 frames, the performance of algorithms *Ma*, *Mc w40* and *Mc w80* improved significantly with implementation of constraints, when applied to pairs of motions with **leading inputs**, with the Error for algorithms *Ma*, *Mc w40* and *Mc w80* reducing by 50, 48 and 68 frames respectively. In comparison, when applied to pairs of motions with **trailing inputs**, constraints had a limited and sometimes negative impact on the performance of the same algorithms, with the Error for algorithm *Ma* reducing by 61 frames and the Error for algorithms *Mc w40* and *Mc w80* increasing by 85 and 129 frames respectively.

Mistimed starts with a larger magnitude can be seen in Sample 2901 in Figure 7.15 and Sample 2795 in Figure 7.14. In both cases constraint is preventing the algorithm from finding the optimal alignment path, as the ideal path falls outside cells that can be reached by the constraint. While the constraint does improve the accuracy of time warping algorithms, the limit it places on the extremity of warp that can be achieved by the algorithm, can cause problems in cases such as this.

Impact of Constraints on Individual Time Warping Algorithms

Out of the four algorithms evaluated, the performance of *Ma*, was enhanced the most by the implementation of the constraint. The algorithm consistently performed better with the constraint applied, for both **leading input** and **trailing input**

mistimed starts at every magnitude. The only outlier was mistimed **leading input** starts between 90 -100 frames, however, this is not a reliable result as only one pair of motions fit into this category. The *Ma* algorithm with the Type V constraint, is able to align motions with mistimed **leading input** starts of up to 70 frames and mistimed **trailing inputs** of up to 30 frames, to the benchmark standard. The *Ma + Type V* algorithm was also the only algorithm able to align pairs motions with **leading input** mistimed starts of greater 100 frames to the benchmark standard.

Although not enhanced to the same extent, the *Mb w40* algorithm also consistently performed better at dealing with mistimed starts, with the constraint implemented. Despite the improvements in performance, the implementation of the constraint did not increase the magnitude of the mistimed starts, neither **leading input** or **trailing input**, that algorithm *Mb w40* could deal with and still meet the Error benchmark.

When implemented with the constraint, the *Mc w40* algorithm performed worse than without the constraint, at aligning pairs of motions which start with mistimed **trailing inputs** of any magnitude. In all the Samples in Figure 7.15, the constraint is stopping the *Mc w40* algorithm from skipping the required frames at the start of the alignment. In the case of Sample 3196 the alignment path never finds it way back to the optimal alignment path, resulting in two of the three walk cycles being skipped in the aligned motion. The constraint has a better impact on the performance of the *Mc w40* algorithm when aligning pairs of motions with mistimed **leading inputs**. The *Mc w40* algorithm can meet the benchmark with mistimed **leading input** starts of up to 50 frames, with the constraint implemented, compared to up to 30 frames without.

Implementing the constraint on the *Mc w40* algorithm, results in smoother alignment paths, producing smoother motions as demonstrated by the motion curves in all the samples in Figures 7.15 and 7.14. The jitter discussed in Section 5.4.7 is not eliminated by the constraint but has been reduced.

The ability of the *Mc 80* algorithm to align pairs of motions with mistimed **leading inputs** of all magnitudes is improved by the implementation of the constraint.

Without the constraint the *Mc 80* algorithm is not able to align mistimed **leading inputs** of any magnitude to the benchmark standard. With the constraint implemented the algorithm is able to align mistimed **leading inputs** of 50 frames within the benchmark Error level.

In regards to aligning pairs of motions with mistimed **trailing inputs** starts, the implementation of the constraint on the *Mc 80* algorithm produces highly contrasting results, depending on the magnitude of the mistiming. At smaller magnitudes (up to 50 frames) the constraint greatly improves the performance of the *Mc 80* algorithms, but at larger magnitudes (above 70 frames), the constraint greatly reduces the performance of the algorithm.

7.4.8 Impact of Penalties on Mistimed Starts

As well as constraints, another approach to improving the ability of time warping algorithms to deal with pairs of motions with mistimed starts is to implement penalties, as presented in Chapter 6. To explore this, the performance of the same time warping algorithms selected in Section 7.4.7 were evaluated, with and without penalties implemented, when applied to pairs of motions with **leading input** or **trailing input** mistimed starts. Charts plotting the error values for each time warping algorithm, for both types of mistimed starts at different magnitudes, can be seen in Figure 7.16, for algorithms *Ma* and *Mb w10*, and Figure 7.17, for algorithms *Mc w10* and *Mc w40*. The error value for each time warping algorithm was determined in the same manner as Figure 7.12.

The only algorithm, whose capability to align motions with mistimed starts, was improved significantly by the implementation of the penalties was the *Ma* algorithm. With penalties implemented, on average the algorithm reduced the error caused by mistimed starts with a **leading** and **trailing** input by 29 and 76 frames respectively. For comparison the implementation of penalties had no impact in the performance of the *Mb w10* algorithm, reduced the performance of the *Mc w10*, increasing the error by an average of 11 and 18 frame for **leading** and **trailing** inputs respectively, and only achieved a small improvement in the *Mc w40* algorithm for mistimed **leading**

inputs, reducing the error for by an average of 7 frames for mistimed **leading inputs** and increasing the error by an average of 49 frames for mistimed **trailing inputs**.

Without the implementation of penalties, the *Ma* algorithm is not able to align any pairs of motions with any form of mistimed start at any magnitude, as shown in Figure 7.16. With penalties implemented the algorithm is able produce alignments below the Error benchmark, with mistimed **leading inputs** of up to 70 frames and mistimed **trailing inputs** of up to 10 frames, using the normalised *Pn0.5* penalty. Interestingly, the plot at the top of Figure 7.16, shows a clearer difference between the performance of the *Pn0.5* normalised penalty and the penalties using a fixed coefficient, than test performed in Chapter 6.

More detailed visualisations of the alignment paths plotted by the *Ma* algorithm, with and without penalties applied, along with an example resulting motion curves can be seen in Figure 7.18. Examples of time warping pairs of motions with **leading inputs** can be seen on the left and **trailing inputs** on the right.

Figure 7.18 shows the implementation of penalties pulling the alignment towards plotting a linear path, where each alignment point steps forward one frame in both the input and target motions. For example Sample 3196 in Figure 7.18, shows the *Ma* algorithm without a penalty, shown in green, sticking on a number input frames when plotting an alignment. With penalties applied, the larger the penalty weighting, the closer the alignment path plotted resembles a slope with a one to one gradient.

Despite improving the ability of the *Ma* algorithm to align pairs of motions with mistimed starts, it is important to recognize that the implementation of penalties, is causing the a time warping algorithm to plot an alignment that more closely resembles a one to one gradient, not to plot an alignment that more closely resembles the optimal alignment path, as plotted by the DTW algorithm shown in blue. In many cases where the alignment path plotted by the *Ma* algorithm incorrectly sticks on input frames, such as Samples: 3196; 2831; 2901; and 2897, plotting an alignment that more closely resembles a one to one gradient has also resulted in an alignment

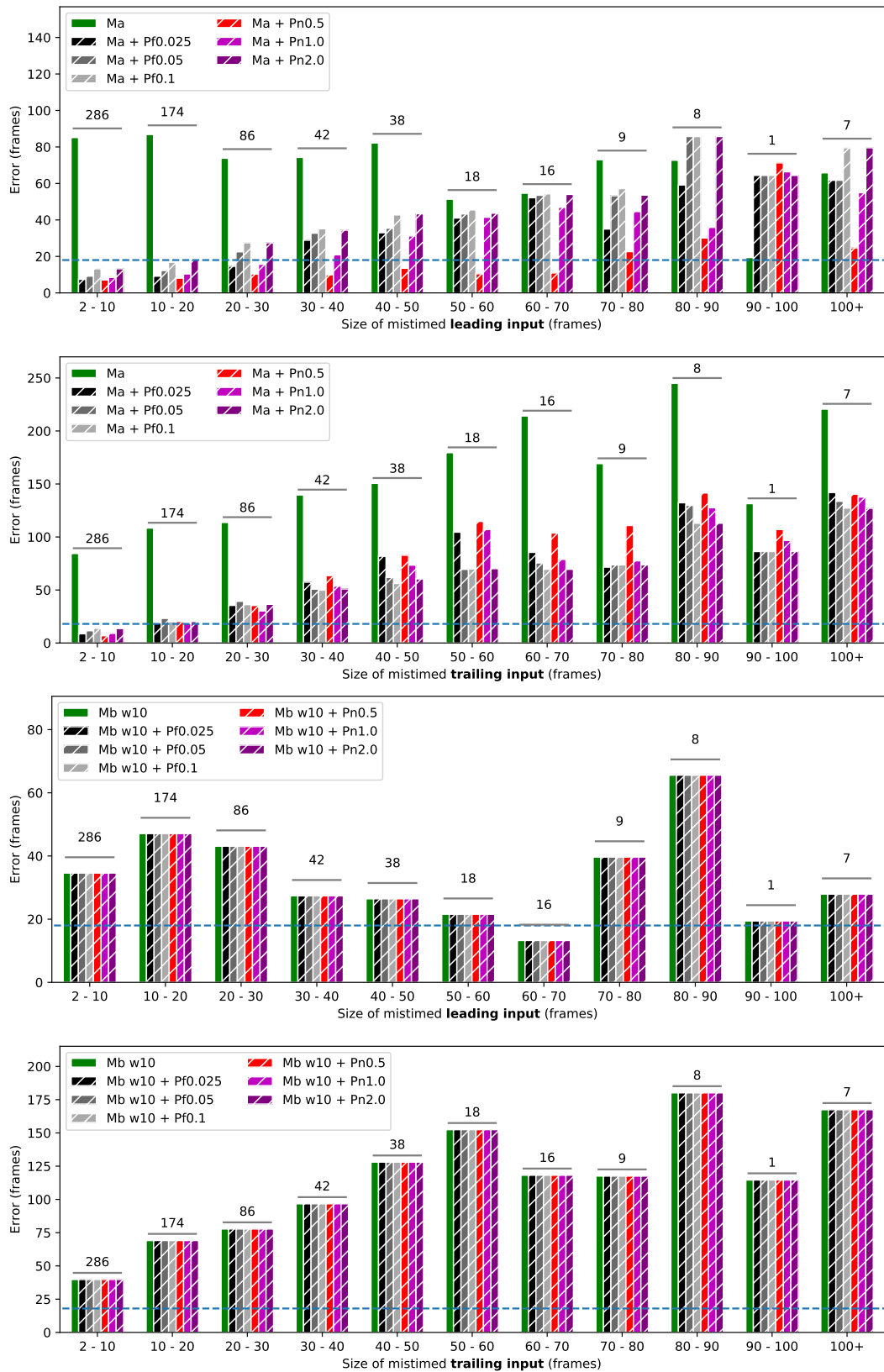


Figure 7.16: Charts showing the capability of the *Ma* and *Mb w10* time warping algorithms, with and without different penalties applied, at aligning pairs of motions starting with mistimed **leading inputs** and **trailing inputs** of various sizes, against the allowable Error or deviation from DTW, shown in as a dashed blue line.

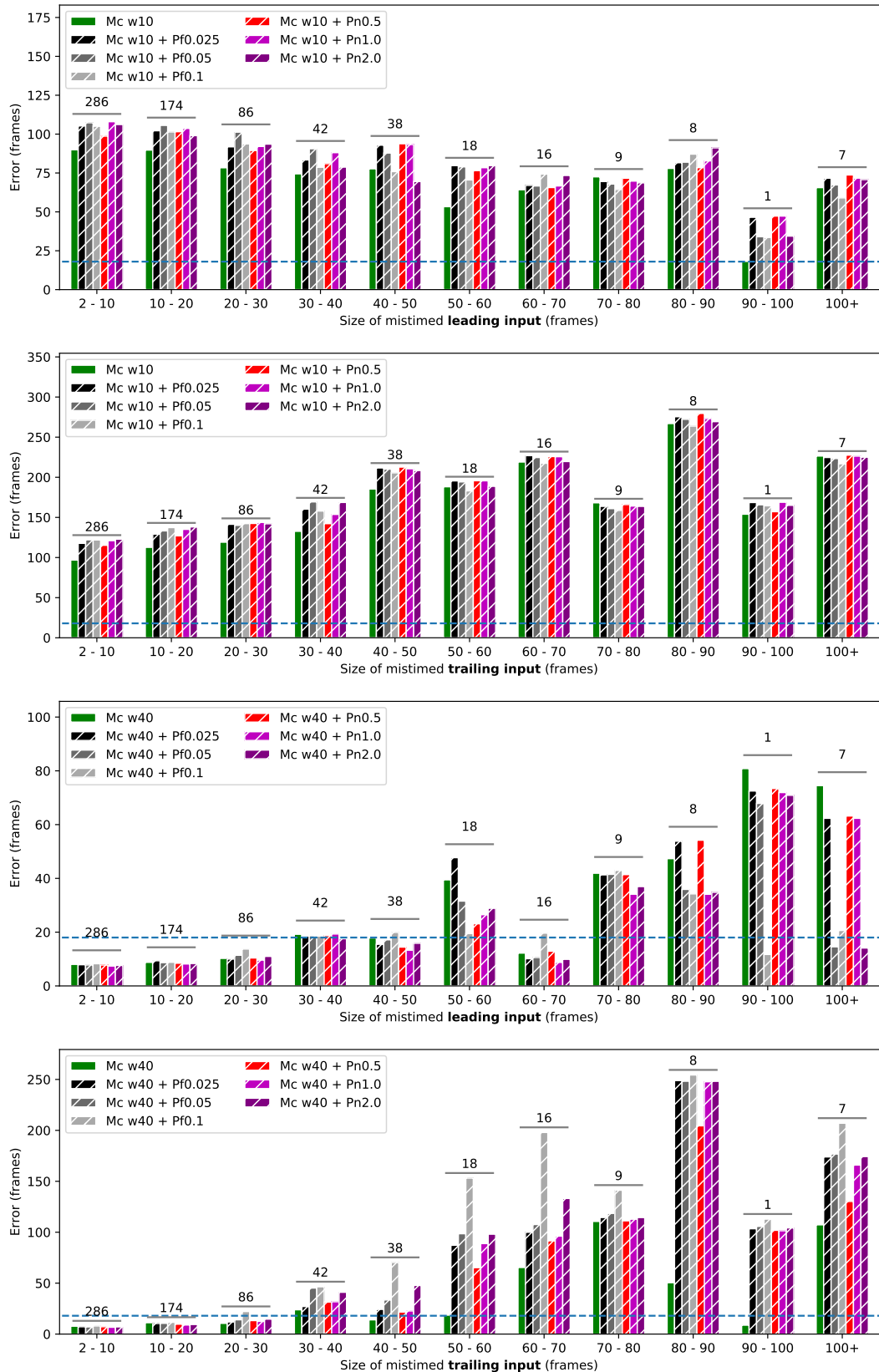
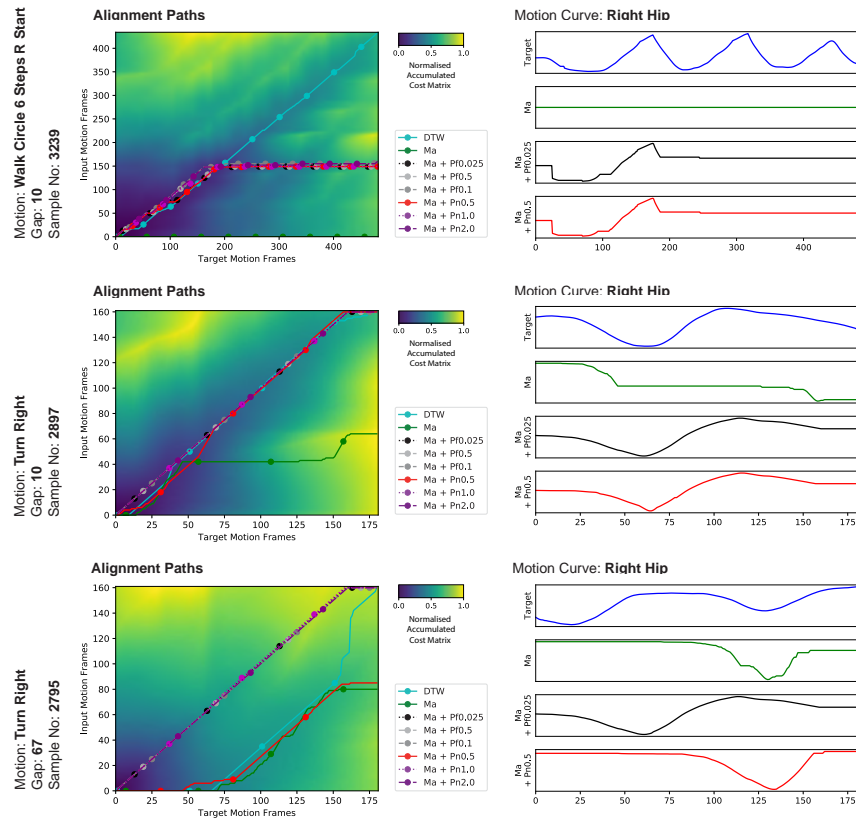


Figure 7.17: Charts showing the capability of the *Mc w10* and *Mc w40* time warping algorithms, with and without different penalties applied applied, at aligning pairs of motions starting with mistimed **leading inputs** and **trailing inputs** of various sizes, against the allowable Error or deviation from DTW, shown in as a dashed blue line.

Timewarps of Mistimed **Leading** Input Pairs of Motions



Timewarps of Mistimed **Trailing** Inputs Pairs of Motion

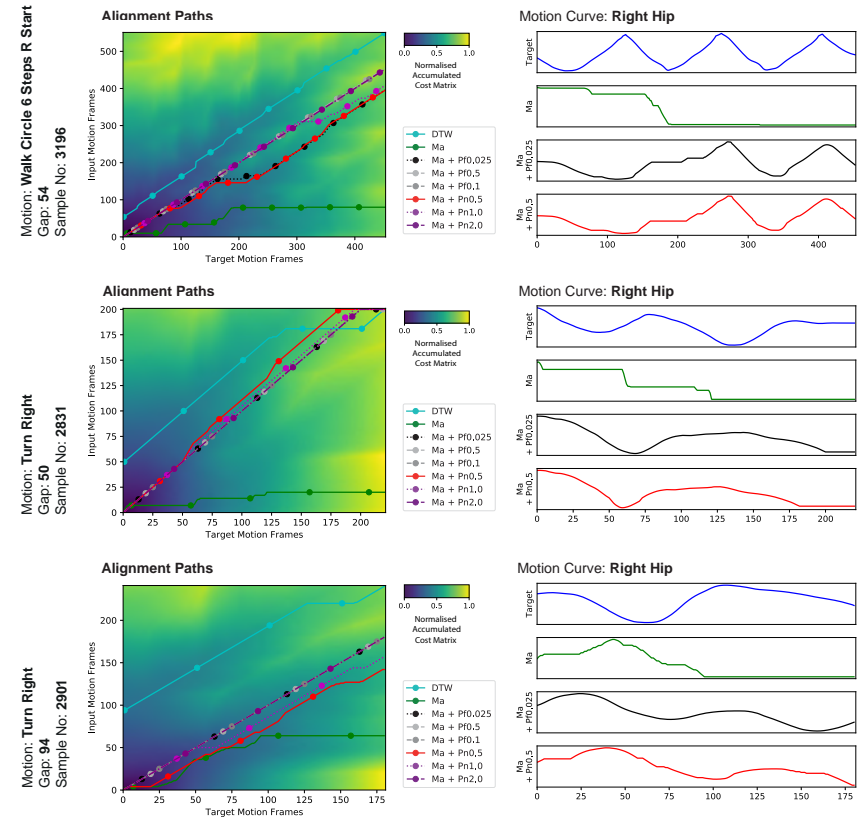


Figure 7.18: A detailed presentation of alignments produced by online time warping algorithms, with and without penalties implemented, when used to align pairs of motions starting with mistimed **leading inputs** (left) and **trailing inputs** (right) of a given number of frames.

that also more closely resembles the optimal alignment path, resulting in the implementation of a penalty, improving the alignment. In other cases where the *Ma* algorithm has already found a reasonable alignment path, such as Sample 2795, plotting an alignment the more closely resembles a one to one gradient has resulted in plotting an alignment that is further away from the optimal path. Interestingly, the algorithm with the more lightly weighted penalty that uses a normalised coefficient, $Ma + Pn0.5$, has avoided this problem.

The implementation of penalties effectively reduces the occurrence of alignments sticking on input frames, regardless whether the sticking results in a correct or incorrect alignment. A comparison of the alignment paths plotted in Figure 7.18 by the *Ma* algorithm shown in green, when aligning mistimed **leading inputs** on the left and mistimed **trailing inputs** on the right, demonstrates the algorithms greater tendency to incorrectly stick on input frames when aligning mistimed **trailing inputs** than mistimed **leading inputs**. Therefore the implementation of a penalty in the *Ma* algorithm is more likely have a positive impact with mistimed **trailing inputs**, where alignment paths sticking on input frames is more likely to result in an incorrect alignment, than with mistimed **leading inputs**, where the alignment path is required to stick on input frames at the start of the alignment.

Another effect of algorithms with larger penalties plotting alignment paths that are closer to a one to one gradient, can be seen in the results for the *Ma* algorithm when time warping mistimed **trailing inputs**, in Figure 7.16. It shows the optimal strength of the penalty changing from small, with smaller mistiming magnitudes, to large, with larger mistiming magnitudes. The *Ma* algorithm, tends to produce poor alignments when time warping pairs of motions with larger mistiming magnitudes, which are more likely to be improved by an alignment path resembling a one to one gradient.

Alignment paths that more closely resemble a one to one slope result in smoother motion curves. Examples of this can be seen Samples 2897, 2901 and 2831 in Figure 7.18, where the straight one to one sloped alignment paths plotted using a fixed penalty of 0.025 ($Ma + Pf0.025$), have produced smoother motion curves than the

less linear alignments paths plotted by the normalised penalty of 0.5 ($Ma + Pn0.5$).

7.4.9 Conclusions on Mistimed Starts

An overview of how the performance of best performing time warping algorithms, with and without penalties and constraints applied, are effected by mistimed **leading input** and **trailing input** starts, can seen in Tables 7.6 and 7.7 respectively. The tables show the median deviation of the alignment paths produced by each time warping algorithm from the path plotted by the DTW algorithm, for mistimed inputs of different magnitudes. The cells highlighted in green are deviations of less than or equal 18 frames from the alignment path plotted by the DTW algorithm and therefore satisfy the frame deviation benchmark. This visualises the magnitude of mistimed start that the each time warping algorithm is able to deal with, while still achieving an acceptable quality of temporal alignment.

The on-line warping algorithms presented in this study cope with mistimed **leading input** starts better than **trailing input** starts. This trend is most clearly demonstrated in the best performing algorithms such as: *Ma* with a Type V constraint applied, which could align **leading inputs** of up to 70 frames but **trailing inputs** of up to only 30 frames, to within the frame deviation benchmark standard; or algorithm **Mb w80** with a Type V constraint applied, which could align leading inputs of up to 50 frames but **trailing inputs** only up to 20 frames, to the benchmark. As discussed in this chapter the trend is driven by the tendency of these time warping algorithms to stick on input frames when plotting an alignment.

Summing the frame deviations for a range of magnitudes across both mistimed **trailing** and **leading** inputs allows time warping algorithms with the most consistent performance to be identified. When summing the deviations of both type of mistimed starts, for magnitudes up to 30 (250ms) and 60 (500ms) frames, the *Mc w40* algorithm is the best performing algorithm in both cases, with a total deviation of 29 and 189 frames respectively.

In real world scenarios, mistimed starts result from temporal differences between the start of the time warp and the motion being performed. These can be caused

Table 7.6: The performance of different time warping algorithms at aligning pairs of motions, starting with mistimed **leading inputs** of various sizes. Cells shown in green are where most of the alignment paths plotted, deviate less than the benchmark of 18 frames from alignment path plotted by the DTW algorithm.

Warping Method	Size of mistimed leading input in frames										
	2 -10	10 - 20	20 - 30	30 - 40	40 - 50	50 - 60	60 - 70	70 - 80	80 - 90	90 - 100	100+
Ma	85	87	74	74	82	51	55	73	73	19	66
Ma + Type V	7	8	10	11	12	13	10	21	32	68	16
Ma + Pn0.5	7	8	10	10	14	10	11	23	30	71	25
Ma + Pn2.0	13	18	27	34	43	44	54	54	86	64	80
Mb w80	11	12	15	18	13	19	10	41	36	19	23
Mb w80 + Type V	9	9	11	13	11	19	11	16	15	44	26
Mc w40	8	9	10	19	18	39	12	42	47	81	74
Mc w40 + Type V	8	8	9	14	11	18	15	18	31	51	27
Mc w40 + Pn0.5	8	8	10	19	14	23	13	41	54	73	63
Mc w40 + Pn2.0	8	8	11	18	16	29	10	37	35	71	14
Mc w80	82	98	83	71	105	100	108	44	127	80	116
Mc w80 + Type V	8	8	9	15	16	37	22	29	22	52	48

Table 7.7: The performance of different time warping algorithms at aligning pairs of motions, starting with mistimed **trailing inputs** of various sizes. Cells shown in green are where most of the alignment paths plotted, deviate less than the benchmark of 18 frames from alignment path plotted by the DTW algorithm.

Warping Method	Size of mistimed trailing input in frames										
	2 -10	10 - 20	20 - 30	30 - 40	40 - 50	50 - 60	60 - 70	70 - 80	80 - 90	90 - 100	100+
Ma	84	108	114	140	151	180	214	169	245	131	221
Ma + Type V	8	13	16	37	85	99	117	114	164	114	159
Ma + Pn0.5	7	20	35	64	83	115	104	111	142	107	140
Ma + Pn2.0	14	20	36	51	60	70	70	74	113	86	127
Mb w80	13	18	30	74	87	134	115	116	175	115	167
Mb w80 + Type V	11	17	21	67	79	118	102	100	161	113	156
Mc w40	8	11	11	24	14	19	65	111	50	9	107
Mc w40 + Type V	8	13	22	46	96	110	126	114	207	118	192
Mc w40 + Pn0.5	7	10	13	31	22	65	92	111	204	102	130
Mc w40 + Pn2.0	7	9	15	41	48	98	133	114	248	104	174
Mc w80	67	74	70	70	60	47	25	29	41	7	33
Mc w80 + Type V	8	10	12	21	19	33	34	109	130	118	162

by delays in either either the algorithm initiating the time warp, causing **leading inputs** or performer starting the movement, causing **trailing inputs**. Delays in a performer starting a movement are likely to be harder to manage and less predictable than delays in the initiating the time warping algorithm. Therefore, time warping algorithms are more likely to need to deal with mistimed **trailing inputs**, which the online time warping algorithms in this study have been shown to be weaker at dealing with.

To attain consistent results a performer needs to start their motion within 30 frames or 250ms of the time warp starting. Trained musicians have been shown to be able to start a performance with mean accuracy of 31ms (Lidar, 2016) and participants are able move their arms to follow a metronome with a mean asynchronicity of 19ms (Honisch, 2012). This indicates that with the use of timing techniques such as count downs, performers would be able to time the start of the movement to within 250ms. Alternatively the start of motions could be automatically detected using prediction algorithms based on neural networks (Carrara et al., 2019) or Support Vector Machines (SVM) classifiers (Oh et al., 2016).

7.5 Speed Differentials

No two performances of a given motion will be performed at exactly the same speed, therefore, a useful time warping algorithm should be able cope with variations in the speed at which motions are performed by a subject. This section explores how well the on-line time warping algorithms in this study are able to align input and target motions, in which the motions have been performed at different speeds.

7.5.1 Visualising the Impact of Speed Differentials on Alignment

To visualise the impact of motions performed at different speeds, on the performance of the on-line warping algorithms presented in this study. Examples of motion pairs, in which the same movement has been performed at different speeds and with minimal mistiming between the starts of two motions, were identified. The alignment

paths plotted by the time warping algorithms for each of these examples, can be seen in Figures 7.19, 7.20, 7.21 and 7.22, along the resulting motion curves. The two samples in each figure show the alignment of the same two motions in opposite directions. The sample with shortest motion set as input motion, is presented on the left, while the sample with shortest motion set as the target motion is presented on the right. For each sample the alignment paths plotted by the standard on-line time warping algorithms are shown at the top, algorithms with the Type V constrain applied are shown in the middle and the *Ma* algorithms with penalties applied are shown at the bottom.

The performance of the samples in Figures 7.19, 7.20, 7.21 and 7.22, with **faster input** motions and **faster target** motions, is summarised in Tables 7.8 and 7.9 respectively. They show the average deviation of the alignment path plotted by each time warping algorithm from the DTW alignment. Where the result is highlighted in green the alignment is within the frame deviation benchmark of 18 frames.

7.5.2 Impact of Speed Differentials on Standard Time Warping Algorithms

Tables 7.8 and 7.9 show that the on-line time warping algorithms are significantly better at aligning pairs of motions, in which the movement is performed faster in the input motion, than where the movement is performed faster in the target motion. Out of the 48 combinations of sample motion pairs with **faster inputs** and time warping algorithms, as presented in Table 7.8, 42 met frame deviation benchmark. This is in comparison to only 18 of the combinations using sample motion pairs with **faster targets**, in Table 7.9.

The difference in the performance of the on-line time warping algorithms, between aligning pairs of motions with **fast inputs** and pairs with **fast targets**, can be seen in the Figures 7.19, 7.20, 7.21 and 7.22.

The alignment paths plotted for the **fast input** motion pairs, on the left side of each figure, show better quality alignments, with many of the time warping algorithms able plot alignment paths that resembled that of the blue DTW alignment path. For

Sample No: 214, Motion: Deposit Floor R Hand
 Input Motion Length: 254 frames, Target Motion Length: 321 frames

Sample No: 215, Motion: Deposit Floor R Hand
 Input Motion Length: 321 frames, Target Motion Length: 254 frames

287

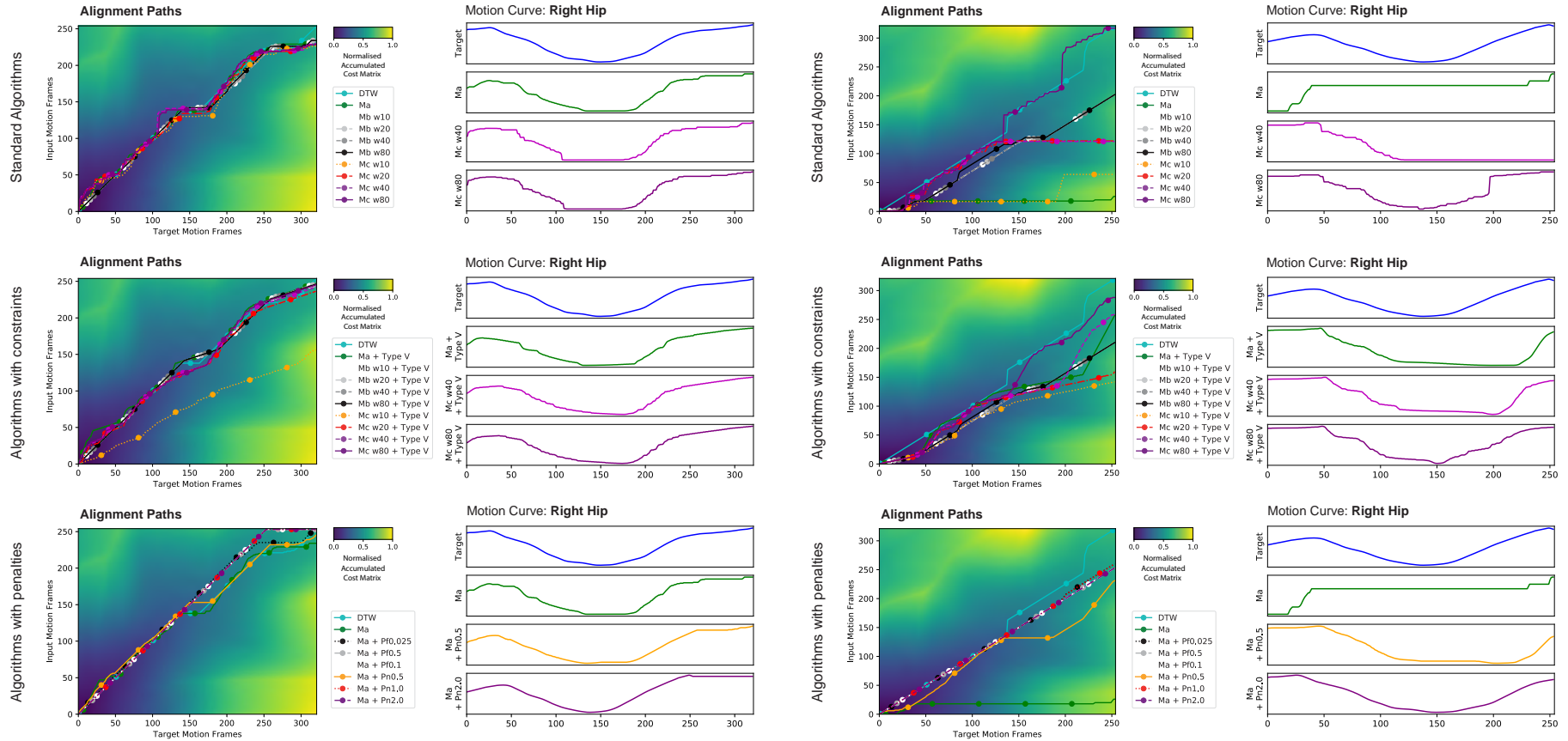
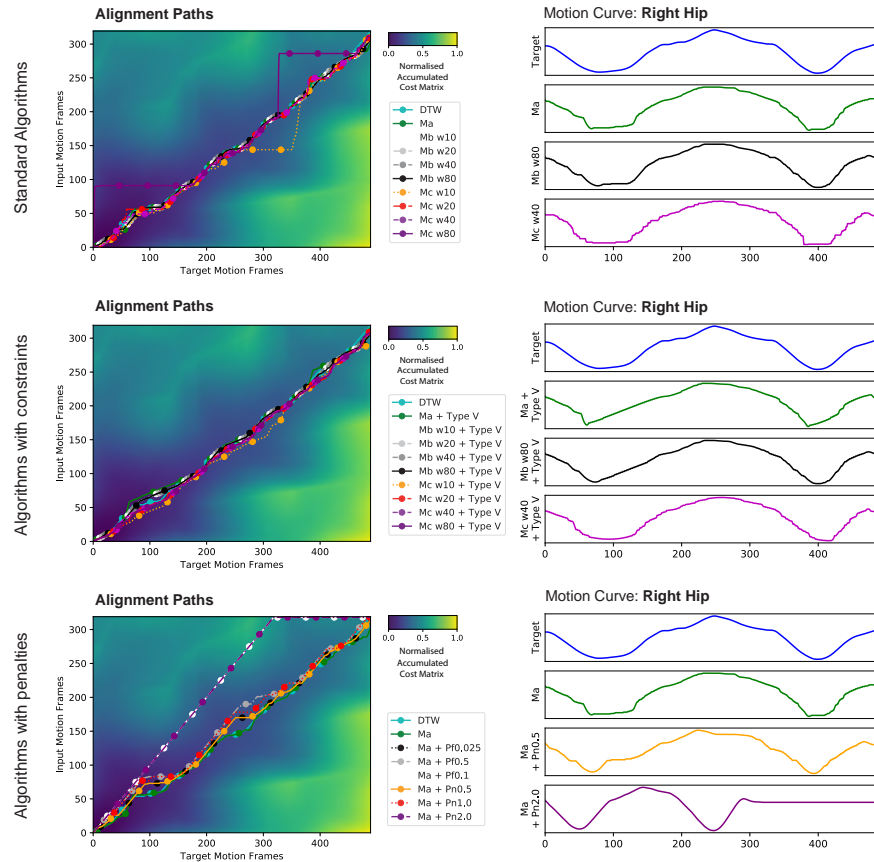


Figure 7.19: The alignment paths and motion curves, resulting from time warping two motions in which the same movements have been performed at different speeds. The same two motions have been time warped in different directions in samples 214 and 215. The input motion is performed faster than the target motion in the left hand sample and visa versa on the right.

Sample No: 334, Motion: **Elbow To Knee 3 Reps L Start**
 Input Motion Length: **319 frames**, Target Motion Length: **489 frames**



Sample No: 335, Motion: **Elbow To Knee 3 Reps L Start**
 Input Motion Length: **489 frames**, Target Motion Length: **319 frames**

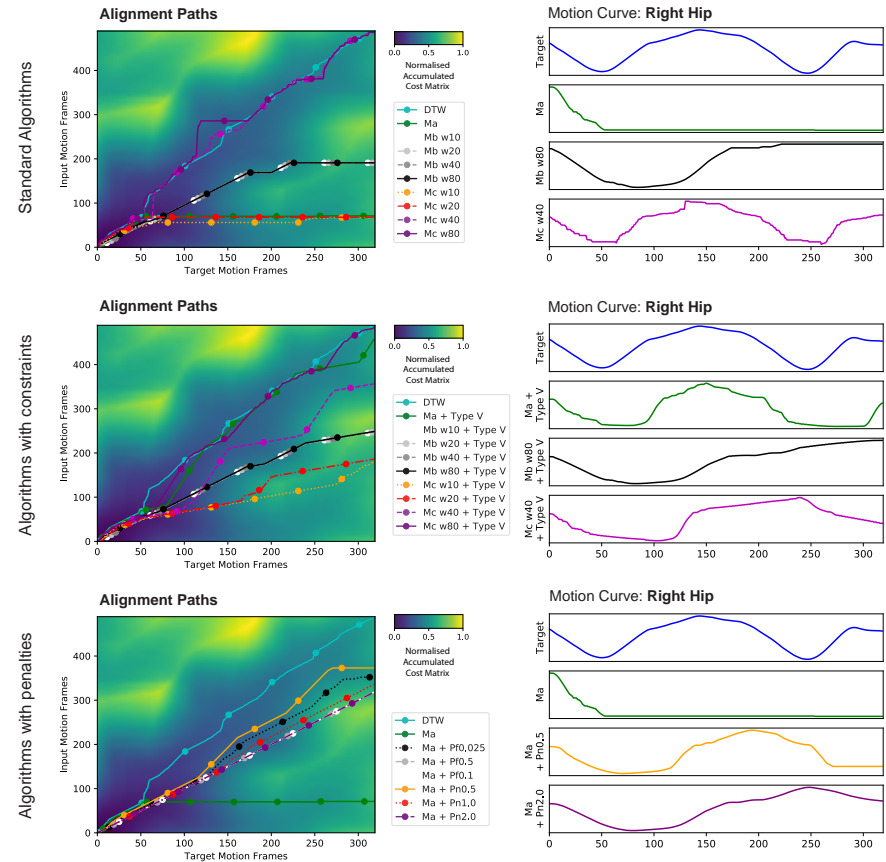
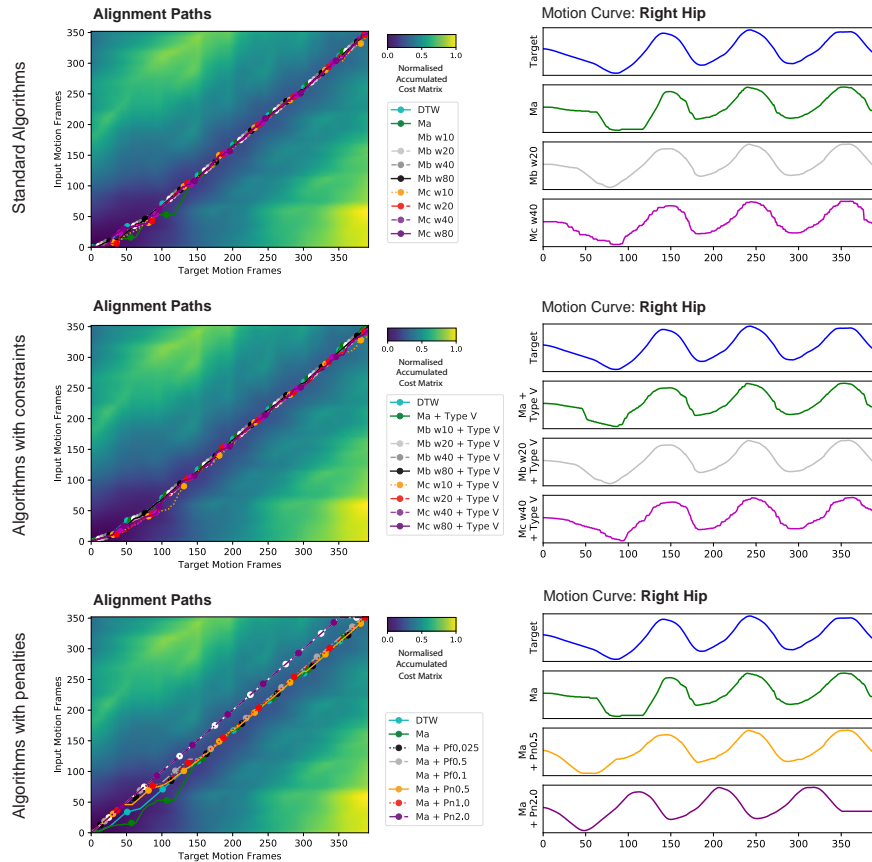


Figure 7.20: The alignment paths and motion curves, resulting from time warping two motions in which the same movements have been performed at different speeds. The same two motions have been time warped in different directions in samples 334 and 335. The input motion is performed faster than the target motion in the left hand sample and visa versa on the right.

Sample No: 842, Motion: Jog Left Circle 6 Steps R Start
 Input Motion Length: 352 frames, Target Motion Length: 392 frames



Sample No: 843, Motion: Jog Left Circle 6 Steps R Start
 Input Motion Length: 392 frames, Target Motion Length: 352 frames

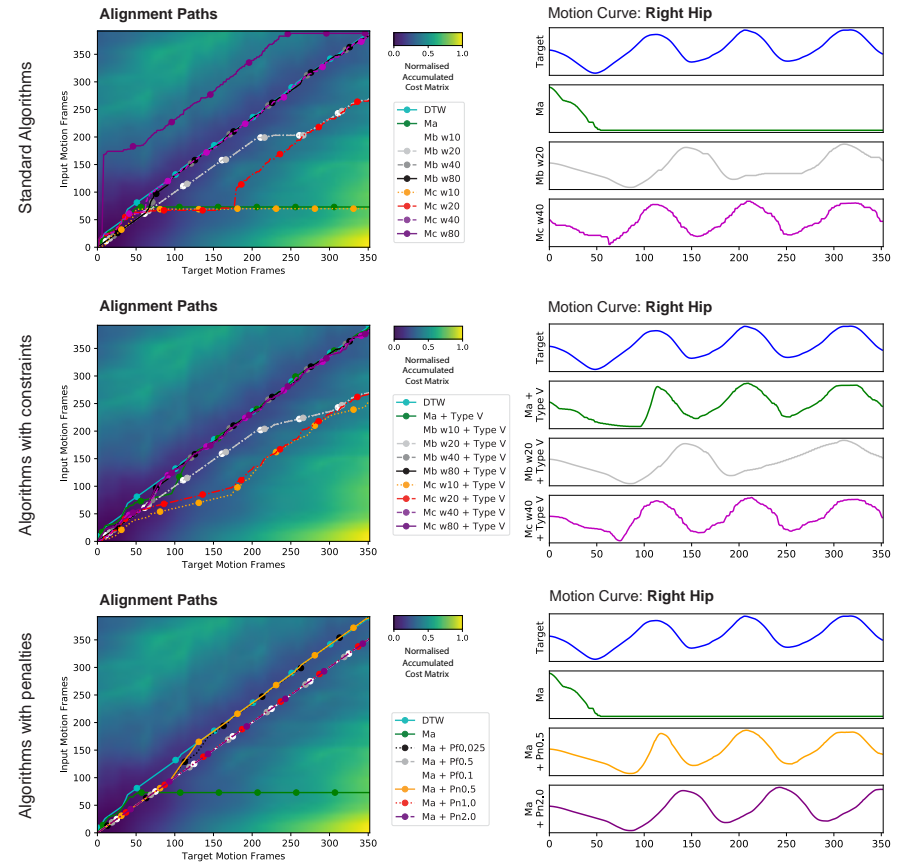


Figure 7.21: The alignment paths and motion curves, resulting from time warping two motions in which the same movements have been performed at different speeds. The same two motions have been time warped in different directions in samples 842 and 843. The input motion is performed faster than the target motion in the left hand sample and visa versa on the right.

Sample No: 2642, Motion: Stand Up Sit Chair
 Input Motion Length: 301 frames, Target Motion Length: 327 frames

Sample No: 2643, Motion: Stand Up Sit Chair
 Input Motion Length: 327 frames, Target Motion Length: 301 frames

290

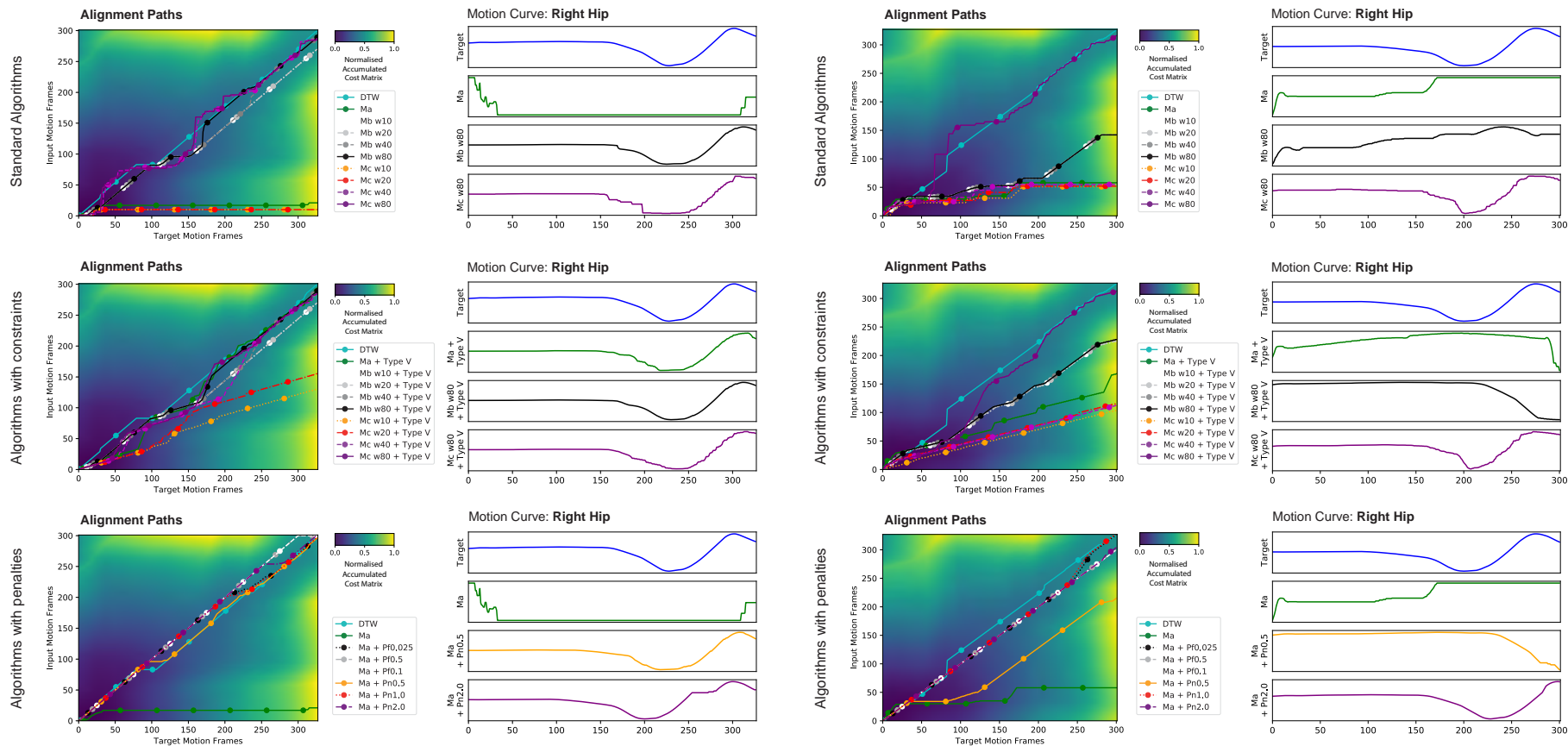


Figure 7.22: The alignment paths and motion curves, resulting from time warping two motions in which the same movements have been performed at different speeds. The same two motions have been time warped in different directions in samples 2642 and 2643. The input motion is performed faster than the target motion in the left hand sample and visa versa on the right.

Samples 214 and 842, all the standard time warping algorithms, without any penalty or constraint applied, were able to plot acceptable alignment paths. In Sample 334 the only standard time warping algorithm to fail was *Mc w80*, while *Ma*, *Mc w10* and *Mc w20* failed in Sample 2642.

The alignment paths plotted on the right for the samples with **fast target** motion pairs, have clearly resulted in poorer alignment, with many alignments paths deviating considerably from the blue DTW path, while many of the resulting motion curves show limited resemblance to the target motion. Out of the standard algorithms, *Mc w80* was the best performing algorithm for aligning **faster target** motion pairs, producing a useful alignment in 3 out of 4 examples. As seen in Section 5.4.6, many of the time warping algorithms fail by incorrectly sticking on input frames when plotting an alignment. These can be seen by the horizontal lines in the alignment paths and subsequent flat lines in the motion curves.

The tendency of the on-line time warping algorithms to stick on input frames appears to be beneficial when aligning pairs of motions with a **faster input**. The frame sticking keeps the alignment on track when the correct alignment path is shallower than a 1:1 diagonal slope. For **faster target** motion pairs, the opposite is true, with algorithms that have a tendency to skip frames, such as **Mc w80** performing better, correctly plotting alignment paths that are steeper than a 1:1 diagonal slope.

7.5.3 Impact of Constraints and Penalties on Dealing with Speed Differentials

Neither the implementation of constraints or penalties consistently improved the performance of any given time warping algorithm in all cases. In both cases the implementation of constraints and penalties had a more positive impact on the alignment of **faster target** motion pairs. Tables 7.8 and 7.9 show that constraints resulted in an improved alignment in 6 out of 16 cases with **faster inputs**, compared to 12 out of 16 for cases with **faster targets**. They also show a similar trend with penalties, which improved the alignment in 5 out of 16 cases with **faster inputs**, compared to 8 out of 16 cases with **faster targets**. This is to be expected as the

Table 7.8: The performance of different on-line time warping algorithms at aligning pairs of motions, in which the movements are performed faster in the input motion.

Sample No.	214	334	842	2642
Motion	Deposit Floor R Hand	Elbow To Knee 3 Reps L Start	Jog Left Circle 6 Steps R Start	Stand Up Sit Chair
Input Duration Frames	254	319	352	301
Target Duration Frames	321	489	392	327
Duration Difference (%)	79%	65%	90%	92%
Ma	5	2	5	129
Ma + Type V	7	4	1	12
Ma + Pn0.5	6	6	8	3
Ma + Pn2.0	13	67	33	14
Mb w80	4	2	2	10
Mb w80 + Type V	4	3	1	11
Mc w40	7	4	3	10
Mc w40 + Type V	5	5	5	22
Mc w40 + Pn0.5	6	5	3	9
Mc w40 + Pn2.0	7	7	4	108
Mc w80	6	28	3	9
Mc w80 + Type V	6	6	5	13

Table 7.9: The performance of different on-line time warping algorithms at aligning pairs of motions, in which the movements are performed faster in the target motion.

Sample No.	215	335	843	2643
Motion	Deposit Floor R Hand	Elbow To Knee 3 Reps L Start	Jog Left Circle 6 Steps R Start	Stand Up Sit Chair
Input Duration Frames	321	489	392	327
Target Duration Frames	254	319	352	301
Duration Difference (%)	79%	65%	90%	92%
Ma	127	198	141	127
Ma + Type V	37	21	6	88
Ma + Pn0.5	40	61	4	73
Ma + Pn2.0	17	103	30	19
Mb w80	49	132	6	104
Mb w80 + Type V	45	117	7	55
Mc w40	55	7	4	127
Mc w40 + Type V	36	82	8	108
Mc w40 + Pn0.5	57	7	5	133
Mc w40 + Pn2.0	56	188	6	135
Mc w80	10	11	87	12
Mc w80 + Type V	14	11	7	17

poor alignment achieved by the time warping algorithms without a constraint or penalty, allowed more opportunity for a better alignment to be achieved.

According to Tables 7.8 and 7.9, the algorithms that performed the most consistently at aligning pairs of motions, with either **faster inputs** or **faster targets**, to within a deviation (or Error) of 18 frames, were *Mc w40* (6 out of 8), *Mc w40 + Pn0.5* (6 out of 8), *Mc w80* (6 out of 8) and *Mc w80 + Type V* (8 out of 8).

In some cases, the use of a constraint or penalty when aligning a motion pair with a **faster input**, has resulted a significant drop in the quality of alignment. For example the use of the *Pn2.0* penalty with algorithm *Ma*, increased the deviation (or Error) in the alignment from 2 to 67 for Sample 334 and from 5 to 33 in the case of Sample 842.

Out of the three constraint and penalty methods shown in Table 7.8 (*Type V*, *Pn0.5*, *Pn2.0*), the *Pn0.5* penalty was the only approach that did not result in an alignment with a deviation above the allowed 18 frame benchmark. While the *Pn0.5* penalty did significantly improve the alignment of the *Ma* algorithm in Sample 2642, algorithms with a *Pn0.5* penalty applied, perform marginally worse in 5 out of 8 cases with **faster inputs**, compared to the same algorithm without a constraint or penalty applied. The sample motion pairs with a **faster input**, shown on the left of Figures 7.19, 7.20, 7.21 and 7.22, show penalties with large weightings (*Pf0.1* and *Pn2.0*), causing the *Ma* algorithm to incorrectly skip input frames during alignment.

Tables 7.8 and 7.9 show the low weighted *Pn0.5* penalty out performing the larger weighted *Pn2.0* penalty in 14 out of 16 cases.

As seen previously in Section 5.4.7, the use of constraint and penalties with algorithms based on Method C, results in smoother motion curves. With penalties producing the smoothest curves.

7.5.4 Conclusions on Speed Differentials

Overall the time warping algorithms presented in this study are better at aligning pairs of motions with a **faster input** motion, than those with a **faster target**

motion.

The *Mc w80* algorithm with a Type V constraint is the best performing time warping approach when dealing with speed differentials, especially in cases where a **faster input** or **faster target** are equally likely to occur. However, this approach is computationally expensive, therefore without further optimisation of the algorithm, the *Mc40* algorithm would have to be used to avoid dropping frames.

As discussed in Chapter 6, constraints are more effective at controlling the tendency of algorithms to incorrectly skip input frames, than the tendency to stick on input frames, during a time warp. Hence, the *Type V* constraint is particularly effective when used with the *Mc w80* algorithm.

The findings in this section suggests that the performance of the online time warping algorithms presented in this study, could potentially be optimised by re-sampling the input motion to increase its speed, using UTW. As only a limited number of samples have been reviewed in this section, this would need further exploration.

While this section has presented some useful findings, they are based on a small selection of samples, with further research required to confirm if these are more generalizable. Such a study could also consider the impact of multi-modal speed differentials with a mixture of **faster input** and **faster target** segments within the same motion.

When designing such a study, a key point to consider from the study presented in [Randall, Williams and Athwal \(2017\)](#), is that a study using simulated speed differentials must not use pairs of the same motion recording, with one motion warped to change its duration. The performance of the time warping algorithms presented in this study, would be artificially high if applied to aligning warped versions of the same recording, as all the joints in motion would agree on the same alignment path. The variations in the timing and poses of individual joints that naturally occur between separate recordings of the same motion, often cause joints to contradict each other regarding which alignment path is optimal. A better approach to simulating speed differentials would be to use DTW to warp motion *A* to the

same duration as motion B , then use then use UTW to change duration of one of the motions as desired.

Table 7.10: Tables showing the performance of different online time warping algorithms, when aligning pairs of motions with either an input or target motion that has a shorter duration. The tables are split into algorithms with no penalty or constraint (top), with constraints (middle) and with penalties (bottom).

Algorithm	Ma	Mb w10	Mb w20	Mb w40	Mb w80	Mc w10	Mc w20	Mc w40	Mc w80
Constraint/Penalty	None	None	None	None	None	None	None	None	None
Shorter Input	0.109	0.180	0.185	0.206	0.238	0.091	0.208	0.275	0.164
Shorter Targets	0.083	0.078	0.085	0.121	0.175	0.064	0.184	0.267	0.207

Algorithm	Ma	Mb w10	Mb w20	Mb w40	Mb w80	Mc w10	Mc w20	Mc w40	Mc w80
Constraint/Penalty	Type V	Type V	Type V	Type V	Type V	Type V	Type V	Type V	Type V
Shorter Inputs	0.274	0.233	0.238	0.255	0.266	0.097	0.199	0.267	0.280
Shorter Targets	0.200	0.085	0.094	0.129	0.170	0.033	0.106	0.207	0.267

Algorithm	Ma	Ma	Ma	Ma	Ma	Ma	Ma
Constraint/Penalty	None	Pf0.025	Pf0.05	Pf0.1	Pn0.5	Pn1.0	Pn2.0
Shorter Inputs	0.109	0.229	0.169	0.113	0.254	0.198	0.109
Shorter Targets	0.083	0.158	0.118	0.102	0.178	0.138	0.104

The characteristic of on-line time warping algorithms presented in this study, are better able align pairs of motions with **faster inputs**, aligns well with the characteristic that they are better able to align pairs of motions with a **leading inputs**, established in Section 7.4. The tendency for these on-line time warping algorithms to stick on input frames during an alignment, better supports time warping these types of motion pairs.

Both **faster inputs** and **leading inputs** will have motion pairs in which the input motion is shorter in duration. Table 7.10 provides a more general picture of how the time warping algorithms perform when aligning, motion pairs with a shorter input motion (likely caused by a **faster input** or **leading input**), compared to motion pairs with shorter target motion (likely caused by a **faster target** or **trailing input**). Any of the 3248 sample motion pairs in the study’s data-set with a difference of greater than 18 frames, between the duration’s of input and target motions, were split into two bins, depending on whether the input or target motion was shorter. This process resulted in 1258 samples in each bin. Standard on-line time warping algorithms, with no penalty or constraint applied, are shown at the top, algorithms

with constraints are shown in the middle and algorithms with penalties are shown at the bottom.

Table 7.10 shows almost every time warping algorithm performing better, when aligning pairs of motions with a shorter duration input motion. The only exception is the *Mc w80* algorithm without constraints or penalties applied, due to being the only approach with tendency to skip input frames. Applying a *Type V* constraint to the *Mc w80* algorithm, increases its performance at aligning pairs of motions with shorter target motions, resulting in an algorithm that performs consistently well across both types of motion pairs. Another algorithm that performs well across both types of motion pairs is *Mc w40*. The consistent performance of the *Mc w40* algorithm combined with smaller amount of computation required, when compared to the *Mc w80 + Type V* algorithm, reinforced why this is best choice of algorithm for on-line time warping scenarios which require forward plotting of the alignment path.

7.6 Summary

This chapter explored the impact variety of different movement and motion data characteristics on the performance of the on-line time warping algorithms presented within this thesis. Suggestions were made as to how these characteristics could be controlled and particularly impactful characteristics were studied in some detail.

Key considerations and suggestions arising from this chapter, regarding the implementation of on-line time warping algorithms, are as follows:

- Rather than use a generic set of joint weights, joints should be weighted according the joints being utilised by a specific motion. Pre-recorded motions can be analysed to establish an optimal set of weights, which are specific to a motion.
- Data errors can occur during motion capture, for example if a system drops frames during recording, overly constrains the range of movement of a given joint or captures noisy data. There is need to evaluate the occurrence of these

characteristics in any motion capture system that is planned to be used as a source of motion data for on-line time warping.

- There is potential for the on-line warping algorithms to perform sub optimally when the input and target motions are performed by different subjects, in comparison to when they are performed by the same subject.
- The on-line time warping algorithms presented in this thesis cope with mistimed leading input starts better than trailing input starts. Across both types of errors the *Mc w40* algorithm was able to cope most consistently.
- To attain consistent results a performer needs to start their motion within 30 frames or 250ms of the time warp starting. A solution for automatically detecting the start of the motion could be implemented to potentially illuminate this issue.
- The on-line time warping algorithms presented in this thesis are better at aligning pairs of motions with a faster input motion, than those with a faster target motion. Although the *Mc w80 +Type V* algorithm was shown to be the best at dealing with speed differentials, it is suggested that the *Mc w40* should still be used to keep computational requirements within a feasible threshold.
- As the on-line warping algorithms performed better at aligning pairs of motions with a faster input motion, their performance could potentially be optimised by increasing the speed of the pre-recorded input motion before time warping.

As shown in the suggestions above, the studies in this chapter have identified several approaches to potentially further optimise the performance of the on-line time warping algorithms presented in this thesis. This opens up opportunities to implement and evaluate these approaches in further work.

Chapter 8

Conclusions and Recommendations

The primary aim of this work was to develop and evaluate an approach to on-line temporal alignment of human motion sequences, using a forward plotting approach to avoid the lack of continuity and non-monotonic alignment associated with existing on-line time warping algorithms. This was achieved by first establishing an approach to measuring the accuracy of the alignment of two motions. Existing approaches to evaluating similarity of two motions were reviewed in Chapter 3, then this knowledge was used to propose and evaluate an optimal approach to measuring alignment in Chapter 4. Findings showed that the a metric based on correlation is more optimal for measuring alignment, as it was better at discriminating between temporally aligned and non-aligned recordings of the same movement.

A number of novel on-line time warping algorithms were then developed and implemented in Chapter 5, which plot alignment paths in a forward direction rather than backwards. The algorithms were based on established techniques use by existing time warping algorithms such as DTW. The performance of the algorithms was evaluated using a number of tests including the metric established in Chapter 4. Findings showed that if configured correctly, a forward plotting time warping algorithm could time warp a motion with sufficient accuracy to be potentially useful in real-world applications.

Following this, in Chapter 6, the forward plotting online time warping algorithms

proposed in this research were implemented with two optimisation techniques, penalties and constraints, and the impact of each technique evaluated. It was found that constraints raised the performance of most forward plotting time warping algorithms. Most importantly the implementation of constraints increased the number of time warping algorithms which work sufficiently well enough to be potentially used in real world applications, providing a choice forward plotting time warping algorithms to choose from.

Finally there was an exploration of the impact of different characteristics within the motion data, on the performance of the on-line time warping algorithms in Chapter 7. The findings showed that not all characteristics adversely effect the performance of the time warping algorithms, with the performance of some algorithms being enhance by particular characteristics or differences between the input and target motions being aligned.

This chapter discusses the recommendations arising from this work and potential applications that this work could be utilised in. This chapter then considers some of the limitations and potential challenges to implementing online forward plotting time warping algorithms in real-world applications, before disussing future work.

8.1 Recommendations

8.1.1 Measuring Similarity and Alignment

Different metrics should be used to measure the similarity and alignment of two motions. Chapter 4 showed that no single metric was optimal for measuring both the alignment and similarity of two motions. Metrics based on correlation were shown to be the optimal choice when measuring the alignment of two motions, while metrics based on rotational or positional distance are the optimal choice when measuring similarity.

This shows that the similarity and alignment of two motions should not be considered the same thing, additionally when choosing a similarity metric the use case should be considered, in terms of what motion characteristic are a priority to be matched

for a given application. For example when classifying a motion the overall pose of a motion will be the priority, therefore a distance base metric using joint rotations is the best approach. When trying to find a motion that best fits interactions within an environment, the position or reach of the end effectors is the priority, therefore a distance based metric based on joint position is the best approach. When trying to blend to motions together, transfer stylistic elements between motions or evaluate the performance of a time warping algorithm, identifying motions that are well aligned is the priority, therefore a similarity metric based on alignment is the best choice.

The optimal approach to measuring alignment is to use a correlation based metric, utilising Kendall Tau correlation on joint rotations parameterised as displacement vectors. Parameterizing joints using displacement vectors confirms the approach suggested by [Etemad and Arya \(2015\)](#), however, the findings in this thesis suggest that Kendall Tau is a better choice than PCC, which has previously been used. Using a correlation method based on rank correlation, results in a similarity metric that is not affected by logarithmic differences between rotation parameters. Kendall Tau’s use of concordant and discordant value pairs, results in a similarity metric which is sensitive to small differences between motions. Both these attributes are useful when measuring the alignment of two motions, however, will cause the metric to be unreliable when applied to dissimilar motions. This should only be considered an optimal metric to use when evaluating the ability of a time warping algorithm to accurately align similar motions.

8.1.2 On-line Time Warping of Motions

The on-line forward plotting time warping methods, to be considered for use in real-world applications are *Ma + Type V*, *Mc w40* and *Mc w80 + Type V*. All of these algorithms meet both the UTW alignment, and the deviation from DTW alignment path benchmarks. They should therefore be considered to provide an accurate enough alignment for real-world applications. While the *Mc w40* algorithm is the best compromise between computational performance and alignment performance, it is useful to have alternative algorithms that can also

satisfy both benchmarks, especially when they complement each other in terms of their ability to align motions with different characteristics. For example the *Ma + Type V* and *Mc w40* algorithms perform better at aligning a faster input motion to slower target motion, while the *Mc w80 + Type V* algorithm performs better at aligning slower inputs motions to faster target motions.

Without the use of a local constraint, the optimal size of forecast window to use with algorithms based on Method C, is approximately a third of the duration of the main cyclic movement within a given motion. The performance of Method C was found to vary significantly, depending on the size of forecast window it was implemented with. Without the implementation of a local constraint, the performance of Method C dropped significantly when implemented with too large window size, unlike Method B. When implementing Method C without a constraint, consideration should be given to the frequency of the most prominent cyclic movement in a motion and the rate at which the motion is being sampled, to determine the optimal window size to use. The most prevalent movement cycle is the walk cycle, which, at a sample rate of 120Hz is typically 133 frames in duration, approximately three times the duration of the best performing window size for Method C.

Assuming a sample rate 120Hz, to ensure that no frames are dropped during alignment, due to processor overload, window sizes over 40 should not be used. The processing requirements of both Methods B and C grow quadratically with the size of forecast window used. Window sizes of 80 were shown to take approximately 17,000 μ s to process, which is larger than the 8,333 μ s duration between each frame of motion captured at 120Hz sample rate. This means that while the *Mc w80 + Type V* algorithm performed best in some tests, it may not be feasible to use in many scenarios. Motions could be captured at a lower sample rate, giving more processing time between each sample, but using a large forecast window with a low sample rate would conflict with the recommendation above.

When implementing an online forward plotting time warping algorithm, consideration should be given to implementing it with local constraints.

The performance of most of the online time warping algorithms proposed in this study, improved with the implementation of constraints. Any further evolution or adaptation of the time warping algorithms proposed in this study should also be tested with local constraints implemented. When optimising a forward plotting time warping algorithm, the implementation of local constraints should be prioritised over the implementation of penalties.

8.2 Review of Aims and Objectives

The overall aim of this work was to develop and evaluate an approach to automatic on-line temporal alignment of human motion sequences. This was achieved through the following objectives:

1. **Review existing approaches to measuring motion alignment and trends within on-line time warping research.** A state of the art review of similarity metrics and time warping algorithms was presented in Chapter 3. Research gaps were identified, regarding approaches to measuring the alignment of motions, the use of correlation in motion similarity metrics, and the use of forward plotting in on-line time warping algorithms to produce monotonically consistent results, when continuously aligning a prerecording motion to a stream of motion data.
2. **Design a robust approach to evaluating and comparing the performance of different similarity metrics for measuring motion alignment.** This was achieved in the methodology presented Chapter 4. Four data-sets of motions were created, then a MAP test and an adapted version of the Mann-Whitney U test, were used to evaluate how well each similarity metric could distinguish between pairs of motions from the Aligned and Non-Aligned data-sets and between pairs of motions from the Similar and Dissimilar data-sets.
3. **Explore and implement appropriate approaches to measuring the**

similarity and alignment of two motions, then determine which approach is optimal for measuring alignment. This was achieved in study presented in Chapter 4. A variety of similarity metrics were implemented, including established metrics based on distance and some novel metrics based on correlation. There was a detailed exploration of the impact of different correlation methods and approaches to parameterise angles, on the performance of correlation base similarity metrics.

4. **Develop and implement solutions for on-line time warping of human motions, evaluating the accuracy and quality of alignments produced by them.** This was achieved in Chapter 5, in which a number of novel approaches to online time warping are presented and evaluated. Two useful performance benchmarks were established, which could be used as threshold to determine if the alignments produced by a time warping algorithm were accurate enough for use in visualisation applications.
5. **Assess the impact of additional optimisation techniques on performance of on-line time warping solutions.** This was achieved in Chapter 6, in which the on-line time warping algorithms presented in Chapter 5, were optimised using penalties and constraints. The performance of the optimised algorithms was compared to the non-optimised algorithms allowing the effectiveness of each optimisation technique to be evaluated.
6. **Assess the impact of characteristics in human motion and motion data, on the performance of on-line time warping solutions, making recommendations on how these characteristics could be managed.** This was achieved in Chapter 7 in which a number of key characteristics associated with human motion and the capture of motion data were identified. Pairs of motions exhibiting these characteristics, were identified within the data-set and used to evaluate their impact on the performance of the time warping solutions proposed in Chapters 5 and 6. Where appropriate, recommendations have been made regarding how to control or mitigate against

characteristics which have a negative impact on the performance of time warping solutions, while identifying characteristics which could potentially have a positive impact on the performance of these solutions.

8.3 Potential Applications of This Work

The online forward plotting time warping algorithms presented and evaluated within this thesis, are able to align a known prerecorded motion, to a partially known motion as it is being captured, without the continuity issues associated with some existing on-line time warping methods. This allows a prerecorded motion, that matches the movements being performed by a live actor, performer or user, to be aligned in real-time, provided their motion is being captured.

Importantly, aligning a matching prerecorded motion to a users motion in real-time, gives a continuous, rather than discreet, understanding of the users temporal position within a movement and the speed at which they are performing it. This information can be used by a computer to provide feedback to the user within motion or movement training applications such dance training or learning medical procedures.

This continuous temporal information could also be used to allow a computer to more accurately co-ordinate an opposing motion, continually updating it as a user performs a given movement. By recording an opposing motion at the same time as the prerecorded input motion, the alignment resulting from the on-line time warping of the prerecorded input motion, can be used to control the opposing motion, by applying the same alignment. The best approach would be to record the opposing motion at the same time as recording the motion being aligned, using two performers.

The time warping algorithms proposed in this thesis, combined with the approach discussed in this section, have a variety of potential applications as outlined in Section 1.1. These included: coordinating the performance of a virtual character when visualising visual effects during film production or during a live stage performance; providing real-time feedback within motion training applications; or allowing cobots

to interact more responsively and intuitively when working with humans.

8.4 Implementation Challenges

While the time warping algorithms proposed and evaluated in this work have been shown to be effective, there is still a number of challenges to overcome to implement them within a complete solution. The main challenges are outlined below and should be considered challenges not barriers, as significant progress has already been made within existing research, to solve these challenges.

- A method of accurately recognising when a motion starts is required, to automatically initiate the time warping process and minimise mistimed starts.
- The time warping algorithms presented within this study need to be modified to cope with frame overloading, which can occur if a time warping algorithm does not process a frame before the next frame is captured.
- The algorithms presented within this study only temporally align motions, there is still a need to spatially align an input motion to support interactions between a virtual character and a live actor. This is, however, a well studied problem ([Ho, Komura and Tai, 2010](#); [Kim et al., 2016](#)).
- When applied to live performance or film production scenarios, there is a need to unobtrusively capture the motion of the actor, using either a markerless motion capture system, which may not coexist well with production lighting, or hidden inertial sensors. The captured motion would also need to be retargeted in real-time, to a joint system that matches that of the prerecorded input motion or visa versa.

8.5 Limitations and Future Work

8.5.1 Joint Weightings

All of the similarity metrics and time warping algorithms used within these studies, where implemented using a fixed subset of equally weighted joints, considered most pertinent to general human motion according to [Lee et al. \(2002\)](#) and [Wang and Bodenheimer \(2003\)](#). While the choice of joints was considered and informed, joints were not individually weighted and there was no analysis of joint movement within motions to determine the optimal weighting to apply to each joint for a given motion. The impact of individually weighted joints and automatically determined joint weightings on the time warping algorithms proposed in this study could be explored further.

8.5.2 Impact of Motion Characteristics

While the impact of motion characteristics has been explored and analysed within Chapter 7, the data-set used in this thesis did not allow the individual impact of each characteristic to accurately evaluated. A more conclusive study needs to be performed using data-sets in which these characteristics are properly isolated. This study would also allow a detailed and accurate exploration of characteristics which improved the performance of the time warping algorithms proposed in this study, with a view to synthesising these characteristics within the prerecorded input motion to enhance their performance.

The data-set used in Chapters 5, 6, 7, was compiled as a set of natural motions to evaluate time warping algorithms, rather than exhibit specific characteristics. To accurately study the impact of particular movement and data characteristics on time warping algorithms, a data-set containing individual isolated and measured characteristics is required. The creation of such as data-set may involve synthesising certain characteristics. As discovered in the naive study presented in [Randall, Williams and Athwal \(2017\)](#), aligning motion A to the same motion with a characteristic and warp applied, A' , will not accurately assess how a time warping algorithm will perform

when aligning two different motions. The following are two approaches which could be taken to avoid this issue:

- Simulating more natural temporal and spatial deviations between motion A and A' , which are independently synthesised for each joint. Deviations that occur between similar motions could be analysed to inform this approach.
- A given characteristic could be removed or matched between two different motions A and B , then synthesised on one of the motions.

Such a data-set would support more detailed studies of time warping and motion manipulation techniques.

8.5.3 Simulating Time Warps

Applying time warping algorithms to simulated frames of single motion curve, would allow a more detailed exploration of their behaviour. Different magnitudes of temporal and spatial deviation, could be simulated in controlled cyclic signals. Simulating the alignment of frames at different phases of the motion cycle, rather than trying to align the entire motion signal, would show how a time warping algorithm performs across different phases of motion. This would provide an insight into how an algorithm deals with peaks and troughs in a motion curve, where the directional change may cause time warping algorithms based on forecasting to perform sub optimally. This study may identify ways to further optimise the time warping algorithms presented in this work.

8.5.4 Perception Testing

While the time warping algorithms presented in this thesis have been evaluated against thresholds established through perception tests ([Hoyet, McDonnell and O'Sullivan, 2012](#)). Further perception tests could be used expand upon this work. For example perception tests are needed to evaluate how sensitive viewers are to motions that are out of alignment and determine the threshold of temporal deviation that can be perceived by the average viewer. Such a study would more fully

evaluate the validity of the UTW and frame deviation benchmarks proposed and utilised in this work. Such a study would need to consider how sensitive viewers are to misalignments in different scenarios involving different types of motion and interactions at various distances and orientations in relation to the camera. The aligned motions produced within this work could potentially be used as an initial data-set for such a study.

Perception tests could be used to evaluate if the jitter within the alignments plotted by time warping Method C are perceptible or not.

8.5.5 Similarity Metrics

Future work could explore other factors that inform the choice of similarity metric such as computing efficiency and sensitivity to errors in motion capture data. For applications working in real-time or with large data-sets, a trade-off of using a slightly less accurate similarity metric for a faster search algorithm maybe more appropriate. Additionally, the motion capture process is susceptible to errors such as noise or gaps in the data, while these errors can be cleaned up, it is an expensive process, so it is often desirable or necessary to work with large data-sets of uncleaned motion data. It is therefore important to understand how different similarity metrics are effected by the data errors described in Section 7.3.

8.5.6 COVID-19

Personal circumstances during the period of the COVID-19 pandemic impacted the progress of this work during this period. The scope of the study was modified to ensure its completion within the registration period. This did not effect the contributions and findings within this work.

8.6 Impact

The optimal similarity metrics and benchmarks established within this thesis, form a robust method of evaluating the performance time warping algorithms and in

particular on-line time warping algorithms, which can be utilised by other researchers working in this field.

The novel forward plotting approach to on-line time warping, opens up a range of potential applications which require a continuous temporal alignment to be maintained, between a set of prerecorded data and a stream of incoming data.

Bibliography

- Ageberg, E., Bennell, K.L., Hunt, M.A., Simic, M., Roos, E.M. and Creaby, M.W., 2010. Validity and inter-rater reliability of medio-lateral knee motion observed during a single-limb mini squat. *BMC musculoskeletal disorders*, 11(1), p.265.
- Akila, A. and Chandra, E., 2013. Slope finder—a distance measure for dtw based isolated word speech recognition. *International Journal of Engineering and Computer Science*, 2(12), pp.3411–3417.
- Al-Asqhar, R.A., Komura, T. and Choi, M.G., 2013. Relationship descriptors for interactive motion adaptation. *Proceedings of the 12th acm siggraph/eurographics symposium on computer animation*. pp.45–53.
- Al-Naymat, G., Chawla, S. and Taheri, J., 2012. Sparsedtw: A novel approach to speed up dynamic time warping. *arXiv preprint arXiv:1201.2969*.
- Alemi, O., 2019. *Pymo*. Available from: <https://omid.al/projects/pymo/>.
- Ali, A., Samara, W., Alhaddad, D., Ware, A. and Saraereh, O.A., 2022. Human activity and motion pattern recognition within indoor environment using convolutional neural networks clustering and naive bayes classification algorithms. *Sensors*, 22(3), p.1016.
- Anguera, X. and Ferrarons, M., 2013. Memory efficient subsequence dtw for query-by-example spoken term detection. *2013 ieee international conference on multimedia and expo (icme)*. IEEE, pp.1–6.
- Aouaidjia, K., Sheng, B., Li, P., Kim, J. and Feng, D.D., 2019. Efficient body

- motion quantification and similarity evaluation using 3-d joints skeleton coordinates. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51(5), pp.2774–2788.
- Arici, T., Celebi, S., Aydin, A.S. and Temiz, T.T., 2014. Robust gesture recognition using feature pre-processing and weighted dynamic time warping. *Multimedia Tools and Applications*, 72(3), pp.3045–3062.
- Arikan, O., 2006. Compression of motion capture databases. *Acm siggraph 2006 papers*, pp.890–897.
- Arikan, O. and Forsyth, D.A., 2002. Interactive motion generation from examples. *ACM Transactions on Graphics (TOG)*, 21(3), pp.483–490.
- Aristidou, A., Stavrakis, E., Charalambous, P., Chrysanthou, Y. and Himona, S.L., 2015. Folk dance evaluation using laban movement analysis. *Journal on Computing and Cultural Heritage (JOCCH)*, 8(4), pp.1–19.
- Ateş, G., Stølen, M.F. and Kyrkjebø, E., 2022. Force and gesture-based motion control of human-robot cooperative lifting using imus. *2022 17th acm/ieee international conference on human-robot interaction (hri)*. IEEE, pp.688–692.
- Autodesk, 2020. *Fbx python sdk*. Available from: <https://www.autodesk.com/developer-network/platform-technologies/fbx-sdk-2020-0>.
- Autodesk, 2021. *Motionbuilder 2022*. Available from: <https://www.autodesk.com/products/motionbuilder/>.
- Bankó, Z. and Abonyi, J., 2012. Correlation based dynamic time warping of multivariate time series. *Expert Systems with Applications*, 39(17), pp.12814–12823.
- Bao, H. and Yao, X., 2020. Human motion data retrieval based on staged dynamic time deformation optimization algorithm. *Complexity*, 2020, pp.1–11.
- Bin, W., Weibin, L. and Weiwei, X., 2020. Motion capture data segmentation using riemannian manifold learning. *Computer Animation and Virtual Worlds*, 31(1), p.e1885.

- Bindiganavale, R. and Badler, N.I., 1998. Motion abstraction and mapping with spatial constraints. *Modelling and motion capture techniques for virtual environments: International workshop, captech'98 geneva, switzerland, november 26–27, 1998 proceedings*. Springer, pp.70–82.
- Boone, D.C. and Azen, S.P., 1979. Normal range of motion of joints in male subjects. *JBJS*, 61(5), pp.756–759.
- Boulic, R., Le Callennec, B., Herren, M. and Bay, H., 2003. *Motion editing with prioritized constraints*.
- Brankovic, M., Buchin, K., Klaren, K., Nusser, A., Popov, A. and Wong, S., 2020. (k, l)-medians clustering of trajectories using continuous dynamic time warping. *Proceedings of the 28th international conference on advances in geographic information systems*. pp.99–110.
- Bruderlin, A. and Williams, L., 1995. Motion signal processing. *Proceedings of the 22nd annual conference on computer graphics and interactive techniques*. pp.97–104.
- Cao, Z., Simon, T., Wei, S.E. and Sheikh, Y., 2017. Realtime multi-person 2d pose estimation using part affinity fields. *Proceedings of the ieee conference on computer vision and pattern recognition*. pp.7291–7299.
- Carrara, F., Elias, P., Sedmidubsky, J. and Zezula, P., 2019. Lstm-based real-time action detection and prediction in human motion streams. *Multimedia Tools and Applications*, 78, pp.27309–27331.
- Carvalho, S., 2009. *Data-driven constraint-based motion editing*. Ph.D. thesis. Swiss Federal Institute of Technology Lausanne. Available from: <http://doi.org/10.5075/epfl-thesis-4558>.
- Celebi, S., Aydin, A.S., Temiz, T.T. and Arici, T., 2013. Gesture recognition using skeleton data with weighted dynamic time warping. *Visapp (1)*. pp.620–625.
- Chan, J.C., Leung, H., Tang, J.K. and Komura, T., 2010. A virtual reality dance

- training system using motion capture technology. *IEEE transactions on learning technologies*, 4(2), pp.187–195.
- Chantaprasert, B., Chunchuen, P. and Wangsiripitak, S., 2019. Comparison of gesture in thai boxing framework using angular dynamic time warping. *2019 16th international conference on electrical engineering/electronics, computer, telecommunications and information technology (ecti-con)*. pp.601–604. Available from: <http://doi.org/10.1109/ECTI-CON47248.2019.8955434>.
- Choi, H., Cho, H. and Kim, T., 2015. Dynamically weighted dtw for dynamic full-body gesture recognition. *Proc. of the int. conf. on intelligent systems and image processing*. pp.126–129.
- Choi, K.J. and Ko, H.S., 1999. On-line motion retargetting. *Proceedings. seventh pacific conference on computer graphics and applications (cat. no.pr00293)*. pp.32–42. Available from: <http://doi.org/10.1109/PCCGA.1999.803346>.
- Choi, K.J., PArk, S.H. and Ko, H.S., 1999. Processing motion capture data to achieve positional accuracy. *Graphical Models and Image Processing*, 61(5), pp.260–273.
- Christensen, H.I., Jebara, T. and Pentland, A., 1999. Action reaction learning: Automatic visual analysis and synthesis of interactive behaviour. *Computer vision systems: First international conference, icvs'99 las palmas, gran canaria, spain, january 13–15, 1999 proceedings 1*. Springer, pp.273–292.
- Cifuentes, J., Pham, M.T., Moreau, R., Prieto, F. and Boulanger, P., 2017. Surgical gesture classification using dynamic time warping and affine velocity. *2017 39th annual international conference of the ieee engineering in medicine and biology society (embc)*. IEEE, pp.2275–2278.
- Ciklacandir, S., Ozkan, S. and Isler, Y., 2022. A comparison of the performances of video-based and imu sensor-based motion capture systems on joint angles. *2022 innovations in intelligent systems and applications conference (asyu)*. IEEE, pp.1–5.

- Claude, G., Gouranton, V., Berthelot, R.B. and Arnaldi, B., 2014. Short paper: #seven, a sensor effector based scenarios model for driving collaborative virtual environment. *Icat-egve, international conference on artificial reality and telexistence, eurographics symposium on virtual environments*. pp.1–4.
- CMU Graphics Lab, 2001. CMU Graphics Lab Motion Capture Database. Available from: <http://mocap.cs.cmu.edu/>.
- Cohen, M.F., 1992. Interactive spacetime control for animation. *Proceedings of the 19th annual conference on computer graphics and interactive techniques*. pp.293–302.
- Dixon, S., 2005. Live tracking of musical performances using on-line time warping. *Proceedings of the 8th international conference on digital audio effects*. Citeseer, vol. 92, p.97.
- Dixon, S. and Widmer, G., 2005. Match: A music alignment tool chest. *Ismir*. pp.492–497.
- Dunn, F. and Parberry, I., 2011. *3d math primer for graphics and game development*. CRC Press.
- Ekimov, A. and Sabatier, J.M., 2011. Rhythm analysis of orthogonal signals from human walking. *The Journal of the Acoustical Society of America*, 129(3), pp.1306–1314.
- Emanuel, A.W. and Widjaja, A., 2018. Real-time 3-d motion gesture recognition using kinect2 as basis for traditional dance scripting. *2018 international conference on advanced computer science and information systems (icacsis)*. IEEE, pp.379–383.
- Etemad, S.A. and Arya, A., 2014a. Classification and translation of style and affect in human motion using rbf neural networks. *Neurocomputing*, 129, pp.585–595.
- Etemad, S.A. and Arya, A., 2014b. Extracting movement, posture, and temporal style features from human motion. *Biologically Inspired Cognitive Architectures*, 7, pp.15–25.

- Etemad, S.A. and Arya, A., 2015. Correlation-optimized time warping for motion. *The Visual Computer*, 31(12), pp.1569–1586.
- Fazlali, H., Sadeghi, H., Sadeghi, S., Ojaghi, M. and Allard, P., 2020. Comparison of four methods for determining the cut-off frequency of accelerometer signals in able-bodied individuals and acl ruptured subjects. *Gait & Posture*, 80, pp.217–222.
- Folgado, D., Barandas, M., Matias, R., Martins, R., Carvalho, M. and Gamboa, H., 2018. Time alignment measurement for time series. *Pattern Recognition*, 81, pp.268–279.
- Freire, S., Santos, G., Armondes, A., Meneses, E.A. and Wanderley, M.M., 2020. Evaluation of inertial sensor data by a comparison with optical motion capture data of guitar strumming gestures. *Sensors*, 20(19), p.5722.
- Fu, A.W.C., Keogh, E., Lau, L.Y.H., Ratanamahatana, C.A. and Wong, R.C.W., 2008. Scaling and time warping in time series querying. *The VLDB Journal*, 17, pp.899–921.
- Geng, W. and Yu, G., 2003. Reuse of motion capture data in animation: A review. *International conference on computational science and its applications*. Springer, pp.620–629.
- Gleicher, M., 1997. Motion editing with spacetime constraints. *Proceedings of the 1997 symposium on interactive 3d graphics*. pp.139–ff.
- Grassia, F.S., 1998. Practical parameterization of rotations using the exponential map. *Journal of graphics tools*, 3(3), pp.29–48.
- Grest, D., Woetzel, J. and Koch, R., 2005. Nonlinear body pose estimation from depth images. *Joint pattern recognition symposium*. Springer, pp.285–292.
- Guerra-Filho, G. and Bhatia, H., 2011. A comparison and evaluation of motion indexing techniques. *International conference on motion in games*. Springer, pp.436–447.

- Hachaj, T. and Ogiela, M.R., 2019. Head motion classification for single-accelerometer virtual reality hardware. *2019 5th international conference on frontiers of signal processing (icfsp)*. IEEE, pp.45–49.
- Han, F., Reily, B., Hoff, W. and Zhang, H., 2017. Space-time representation of people based on 3d skeletal data: A review. *Computer Vision and Image Understanding*, 158, pp.85–105.
- Harada, T., Taoka, S., Mori, T. and Sato, T., 2004. Quantitative evaluation method for pose and motion similarity based on human perception. *4th ieee/ras international conference on humanoid robots, 2004*. IEEE, vol. 1, pp.494–512.
- Haslwanter, T., 2016. An introduction to statistics with python. *With applications in the life sciences. Switzerland: Springer International Publishing*.
- Hegarini, E., Mutiara, A.B., Suhendra, A., Iqbal, M. and Wardijono, B.A., 2016. Similarity analysis of motion based on motion capture technology. *2016 international conference on informatics and computing (icic)*. IEEE, pp.389–393.
- Heloir, A., Courty, N., Gibet, S. and Multon, F., 2006. Temporal alignment of communicative gesture sequences. *Computer animation and virtual worlds*, 17(3-4), pp.347–357.
- Herrington, L., Myer, G.D. and Munro, A., 2013. Intra and inter-tester reliability of the tuck jump assessment. *Physical Therapy in Sport*, 14(3), pp.152–155.
- Ho, E.S., Komura, T. and Tai, C.L., 2010. Spatial relationship preserving character motion adaptation. *Acm siggraph 2010 papers*, pp.1–8.
- Holden, D., Saito, J. and Komura, T., 2016. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (TOG)*, 35(4), pp.1–11.
- Honisch, J.J., 2012. *Dance ensemble synchronisation: movement timing between two or more people*. Ph.D. thesis. University of Birmingham.

- Hoyet, L., McDonnell, R. and O’Sullivan, C., 2012. Push it real: Perceiving causality in virtual interactions. *ACM Transactions on Graphics (TOG)*, 31(4), pp.1–9.
- Hsu, E., Pulli, K. and Popović, J., 2005. Style translation for human motion. *Acm siggraph 2005 papers*, pp.1082–1089.
- Hsu, E., Silva, M. da and Popovic, J., 2007. Guided time warping for motion editing.
- Hsu, Y.L., Chu, C.L., Tsai, Y.J. and Wang, J.S., 2014. An inertial pen with dynamic time warping recognizer for handwriting and gesture recognition. *IEEE Sensors Journal*, 15(1), pp.154–163.
- Hu, Y., Jin, W. and Ni, F., 2012. An efficient wireless sensor network for real-time multiuser motion capture system. *2012 ieee 14th international conference on communication technology*. IEEE, pp.155–160.
- Hülsmann, F., Kopp, S., Richter, A. and Botsch, M., 2017. Accurate online alignment of human motor performances. *Proceedings of the tenth international conference on motion in games*. pp.1–6.
- Hussain, S.M.A. and Rashid, A.H.u., 2012. User independent hand gesture recognition by accelerated dtw. *2012 international conference on informatics, electronics & vision (iciev)*. IEEE, pp.1033–1037.
- Hwang, J., Suh, I.H. and Kwon, T., 2014. Editing and synthesizing two-character motions using a coupled inverted pendulum model. *Computer graphics forum*. Wiley Online Library, vol. 33, pp.21–30.
- Itakura, F., 1975. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on acoustics, speech, and signal processing*, 23(1), pp.67–72.
- Janaki, V.S., Babu, S. and Sreekanth, S., 2013. Real time recognition of 3d gestures in mobile devices. *2013 ieee recent advances in intelligent computational systems (raics)*. IEEE, pp.149–152.
- Jang, M., Kim, D., Kim, Y. and Kim, J., 2017. Automated dance motion evaluation

- using dynamic time warping and laban movement analysis. *2017 ieee international conference on consumer electronics (icce)*. IEEE, pp.141–142.
- Jeong, Y.S., Jeong, M.K. and Omitaomu, O.A., 2011. Weighted dynamic time warping for time series classification. *Pattern recognition*, 44(9), pp.2231–2240.
- Johnson, M.P., 2003. *Exploiting quaternions to support expressive interactive character motion*. Ph.D. thesis. Massachusetts Institute of Technology.
- Kadner, N., 2019. *The virtual production field guide*. Epic Games.
- Kalekar, P.S. et al., 2004. Time series forecasting using holt-winters exponential smoothing. *Kanwal Rekhi school of information Technology*, 4329008(13), pp.1–13.
- Keogh, E.J. and Pazzani, M.J., 2001. Derivative dynamic time warping. *Proceedings of the 2001 siam international conference on data mining*. SIAM, pp.1–11.
- Kim, D., Jang, M., Yoon, Y. and Kim, J., 2015. Classification of dance motions with depth cameras using subsequence dynamic time warping. *2015 8th international conference on signal processing, image processing and pattern recognition (sip)*. IEEE, pp.5–8.
- Kim, J.W., Fouad, H., Sibert, J.L. and Hahn, J.K., 2009. Perceptually motivated automatic dance motion generation for music. *Computer Animation and Virtual Worlds*, 20(2-3), pp.375–384.
- Kim, Y., Park, H., Bang, S. and Lee, S.H., 2016. Retargeting human-object interaction to virtual avatars. *IEEE Transactions on Visualization and Computer Graphics*, 22(11), pp.2405–2412. Available from: <http://doi.org/10.1109/TVCG.2016.2593780>.
- Kovar, L. and Gleicher, M., 2003. Flexible automatic motion blending with registration curves. *Proceedings of the 2003 acm siggraph/eurographics symposium on computer animation*. Eurographics Association, pp.214–224.

- Kovar, L., Gleicher, M. and Pighin, F., 2002. Motion graphs. *Proceedings of the 29th annual conference on computer graphics and interactive techniques*. pp.473–482.
- Kratz, S. and Rohs, M., 2011. Protractor3d: a closed-form solution to rotation-invariant 3d gestures. *Proceedings of the 16th international conference on intelligent user interfaces*. pp.371–374.
- Kratz, S., Rohs, M. and Essl, G., 2013. Combining acceleration and gyroscope data for motion gesture recognition using classifiers with dimensionality constraints. *Proceedings of the 2013 international conference on intelligent user interfaces*. pp.173–178.
- Krüger, B., Tautges, J., Weber, A. and Zinke, A., 2010. Fast local and global similarity searches in large motion capture databases. *Symposium on computer animation*. Citeseer, pp.1–10.
- Kruk, E. Van der and Reijne, M.M., 2018. Accuracy of human motion capture systems for sport applications; state-of-the-art review. *European journal of sport science*, 18(6), pp.806–819.
- Krzyszowski, T., Switonski, A., Kwolek, B., Josinski, H. and Wojciechowski, K., 2014. Dtw-based gait recognition from recovered 3-d joint angles and inter-ankle distance. *Computer vision and graphics: International conference, iccvg 2014, warsaw, poland, september 15-17, 2014. proceedings*. Springer, pp.356–363.
- Lee, J., Chai, J., Reitsma, P.S., Hodgins, J.K. and Pollard, N.S., 2002. Interactive control of avatars animated with human motion data. *Proceedings of the 29th annual conference on computer graphics and interactive techniques*. pp.491–500.
- Lee, J. and Shin, S.Y., 1999. A hierarchical approach to interactive motion editing for human-like figures. *Proceedings of the 26th annual conference on computer graphics and interactive techniques*. pp.39–48.
- Lee, Y.C. and Lee, J., 2016. Kinematic constraint method for human gesture recognition based on weighted dynamic time warping. *2016 6th international conference on it convergence and security (icitcs)*. IEEE, pp.1–5.

- Li, C. and Prabhakaran, B., 2005. A similarity measure for motion stream segmentation and recognition. *Proceedings of the 6th international workshop on multimedia data mining: mining integrated media and complex data*. pp.89–94.
- Li, S., Liang, J., Wu, B., Chen, L., Hu, Z. and Su, J., 2009. Dynamic motion editing by combining an extension of the prioritized inverse kinematics with active dynamic control. *2009 international conference on mechatronics and automation*. IEEE, pp.4067–4074.
- Li, Y., Feng, X., Xu, Y., Dong, X., Xu, Z., Huang, J. and Lu, L., 2019. A dynamic hand gesture recognition model based on the improved dynamic time warping algorithm. *2019 25th international conference on automation and computing (icac)*. IEEE, pp.1–6.
- Li, Z., Xie, X., Zhou, X., Guo, J. and Bie, R., 2015. A generic framework for human motion recognition based on smartphones. *2015 international conference on identification, information, and knowledge in the internet of things (iiki)*. IEEE, pp.299–302.
- Lidar, E., 2016. *Timing in symphony orchestras*. Master’s thesis. Chalmers University of Technology.
- Lim, I.S. and Thalmann, D., 2001. Key-posture extraction out of human motion data. *2001 conference proceedings of the 23rd annual international conference of the IEEE engineering in medicine and biology society*. IEEE, vol. 2, pp.1167–1169.
- Longo, U.G., De Salvatore, S., Carnevale, A., Tecce, S.M., Bandini, B., Lalli, A., Schena, E. and Denaro, V., 2022. Optical motion capture systems for 3d kinematic analysis in patients with shoulder disorders. *International Journal of Environmental Research and Public Health*, 19(19), p.12033.
- Loper, M., Mahmood, N., Romero, J., Pons-Moll, G. and Black, M.J., 2015. Smpl: A skinned multi-person linear model. *ACM Trans. Graph.*, 34(6). Available from: <http://doi.org/10.1145/2816795.2818013>.

- Macrae, R. and Dixon, S., 2010. Accurate real-time windowed time warping. *Ismir*. pp.423–428.
- Markley, F.L., Cheng, Y., Crassidis, J.L. and Oshman, Y., 2007. Averaging quaternions. *Journal of Guidance, Control, and Dynamics*, 30(4), pp.1193–1197.
- Menolotto, M., Komaris, D.S., Tedesco, S., O’Flynn, B. and Walsh, M., 2020. Motion capture technology in industrial applications: A systematic review. *Sensors*, 20(19), p.5687.
- Michaelis, J.E., Siebert-Evenstone, A., Shaffer, D.W. and Mutlu, B., 2020. Collaborative or simply uncaged? understanding human-cobot interactions in automation. *Proceedings of the 2020 chi conference on human factors in computing systems*. pp.1–12.
- Michoud, B., Guillou, E., Briceno, H. and Bouakaz, S., 2007. Real-time marker-free motion capture from multiple cameras. *2007 ieee 11th international conference on computer vision*. pp.1–7. Available from: <http://doi.org/10.1109/ICCV.2007.4408991>.
- Min, J.K., Choe, B. and Cho, S.B., 2010. A selective template matching algorithm for short and intuitive gesture ui of accelerometer-builtin mobile phones. *2010 second world congress on nature and biologically inspired computing (nabic)*. IEEE, pp.660–665.
- Moonen, M. and De Moor, B., 1995. *Svd and signal processing, iii: Algorithms, architectures and applications*. Elsevier Science.
- Müller, M. and Röder, T., 2008. *Human motion: Understanding, modelling, capture and animation*, Springer, chap. A Relational Approach to Content-based Analyses of Motion Capture Data.
- Müller, M., Röder, T. and Clausen, M., 2005. Efficient content-based retrieval of motion capture data. *Acm siggraph 2005 papers*, pp.677–685.
- Müller, M., Röder, T., Clausen, M., Eberhardt, B., Krüger, B. and Weber, A., 2007. Mocap database hdm05. *Institut für Informatik II, Universität Bonn*, 2(7).

- Muscariello, A., Gravier, G. and Bimbot, F., 2009. Variability tolerant audio motif discovery. *International conference on multimedia modeling*. Springer, pp.275–286.
- Myers, C., Rabiner, L. and Rosenberg, A., 1980. Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(6), pp.623–635.
- Neff, M. and Fiume, E., 2003. Aesthetic edits for character animation. *Proceedings of the 2003 acm siggraph/eurographics symposium on computer animation*. pp.239–244.
- Neff, M. and Kim, Y., 2009. Interactive editing of motion style using drives and correlations. *Proceedings of the 2009 acm siggraph/eurographics symposium on computer animation*. pp.103–112.
- Norkin, C.C. and White, D.J., 2016. *Measurement of joint motion: a guide to goniometry*. 5th ed. FA Davis.
- Oh, J., Lee, Y., Kim, Y., Jin, T., Lee, S. and Lee, S.H., 2016. Hand contact between remote users through virtual avatars. *Proceedings of the 29th international conference on computer animation and social agents*. pp.97–100.
- Oregi, I., Pérez, A., Ser, J.D. and Lozano, J.A., 2017. On-line dynamic time warping for streaming time series. *Joint european conference on machine learning and knowledge discovery in databases*. Springer, pp.591–605.
- Osawa, R., Ishikawa, T. and Watanabe, H., 2020. Pitching motion matching based on pose similarity using dynamic time warping. *2020 IEEE 9th Global Conference on Consumer Electronics (GCCE)*. IEEE, pp.1–5.
- Oz, C. and Leu, M.C., 2011. American sign language word recognition with a sensory glove using artificial neural networks. *Engineering Applications of Artificial Intelligence*, 24(7), pp.1204–1213.
- Paluszewski, M. and Karplus, K., 2009. Model quality assessment using distance constraints from alignments. *Proteins: Structure, Function, and Bioinformatics*, 75(3), pp.540–549.

- Parent, R., 2012. *Computer animation: algorithms and techniques*. Newnes.
- Parent, R., Ebert, D.S., Gould, D., Gross, M., Kazmier, C., Lumsden, C.J., Keiser, R., Menache, A., Müller, M., Musgrave, F.K. et al., 2009. *Computer animation complete: all-in-one: learn motion capture, characteristic, point-based, and maya winning techniques*. Morgan Kaufmann.
- Patlolla, C., Mahotra, S. and Kehtarnavaz, N., 2012. Real-time hand-pair gesture recognition using a stereo webcam. *2012 ieee international conference on emerging signal processing applications*. IEEE, pp.135–138.
- Patrona, F., Chatzitofis, A., Zarpalas, D. and Daras, P., 2018. Motion analysis: Action detection, recognition and evaluation based on motion capture data. *Pattern Recognition*, 76, pp.612–622.
- Petitjean, F., Ketterlin, A. and Gançarski, P., 2011. A global averaging method for dynamic time warping, with applications to clustering. *Pattern recognition*, 44(3), pp.678–693.
- Pfister, A., West, A.M., Bronner, S. and Noah, J.A., 2014. Comparative abilities of microsoft kinect and vicon 3d motion capture for gait analysis. *Journal of medical engineering & technology*, 38(5), pp.274–280.
- Plaete, J., Bradley, D., Warner, P. and Zwartouw, A., 2022. Abba voyage: High volume facial likeness and performance pipeline. *Acm siggraph 2022 talks*, pp.1–2.
- Pražák, M., McDonnell, R. and O’Sullivan, C., 2010. Perceptual evaluation of human animation timewarping. *Acm siggraph asia 2010 sketches*, pp.1–2.
- Rabiner, L. and Juang, B.H., 1993. *Fundamentals of speech recognition*. Prentice-Hall, Inc.
- Radke, R.J., 2013. *Computer vision for visual effects*. Cambridge University Press.
- Randall, M., 2022a. Motion capture dataset of single person actions. Available from: https://github.com/matRandall/Mocap_SinglePersonActions/.

- Randall, M., 2022b. Online time warped motions. Available from: <https://github.com/matRandall/exampleOnlineTimewarpedMotions/>.
- Randall, M., Harvey, C. and Williams, I., 2023a. Correlation as a measure for alignment and similarity of human motions. *Computer Animation and Virtual Worlds*, n/a(n/a), p.e2157. <https://onlinelibrary.wiley.com/doi/pdf/10.1002/cav.2157>, Available from: <http://doi.org/https://doi.org/10.1002/cav.2157>.
- Randall, M., Harvey, C. and Williams, I., 2023b. Online alignment of human motion using forward plotting-dynamic time warping. *Computer Animation and Virtual Worlds*, p.e2166. <https://onlinelibrary.wiley.com/doi/pdf/10.1002/cav.2166>, Available from: <http://doi.org/https://doi.org/10.1002/cav.2166>.
- Randall, M., Williams, I. and Athwal, C., 2017. [poster] a predictive approach to on-line time warping of motion. *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*. IEEE, pp.149–154.
- Reitsma, P.S. and O’Sullivan, C., 2009. Effect of scenario on perceptual sensitivity to errors in animation. *ACM Transactions on Applied Perception (TAP)*, 6(3), pp.1–16.
- Reitsma, P.S. and Pollard, N.S., 2003. Perceptual metrics for character animation: sensitivity to errors in ballistic motion. *Acm siggraph 2003 papers*, pp.537–542.
- Röder, T., 2006. *Similarity, retrieval, and classification of motion capture data*. Ph.D. thesis. University of Bonn.
- Rose, C., Cohen, M.F. and Bodenheimer, B., 1998. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5), pp.32–40.
- Sakoe, H. and Chiba, S., 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1), pp.43–49.
- Salvador, S. and Chan, P., 2004. Fastdtw: Toward accurate dynamic time warping

- in linear time and space. *Kdd workshop on mining temporal and sequential data*. Citeseer.
- Saund, C., Matuszak, H., Weinstein, A. and Marsella, S., 2022. Motion and meaning: Data-driven analyses of the relationship between gesture and communicative semantics. *Proceedings of the 10th international conference on human-agent interaction*. pp.227–235.
- Schatz, C. and Derry, G., 2013. Demonstration of the SIMULCAM invented for filming James Cameron’s Avatar. https://www.youtube.com/watch?v=lyHa_OyJB1w.
- Seabold, S. and Perktold, J., 2010. statsmodels: Econometric and statistical modeling with python. *9th python in science conference*.
- Senin, P., 2008. Dynamic time warping algorithm review. *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, 855(1-23), p.40.
- Seward, A.E. and Bodenheimer, B., 2005. Using nonlinear dimensionality reduction in 3d figure animation. *Proceedings of the 43rd annual southeast regional conference-volume 2*. pp.388–392.
- Shimada, M. and Uehara, K., 2000. Discovery of correlation from multi-stream of human motion. *International conference on discovery science*. Springer, pp.290–294.
- Shin, H.J., Lee, J., Shin, S.Y. and Gleicher, M., 2001. Computer puppetry: An importance-based approach. *ACM Transactions on Graphics (TOG)*, 20(2), pp.67–94.
- Shin, J., Hasan, M.A.M. and Maniruzzaman, M., 2022. Hand gesture authentication using optimal feature selection and dynamic time warping based k-nearest neighbor. *The 2022 5th international conference on electronics, communications and control engineering*. pp.22–26.

- Singer, B., 2013. Jack the giant slayer fx. <https://www.youtube.com/watch?v=evT00UF3IUU>.
- Smith, H.J., Cao, C., Neff, M. and Wang, Y., 2019. Efficient neural networks for real-time motion style transfer. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 2(2), pp.1–17.
- Spielmann, S., Schuster, A., Götz, K. and Helzle, V., 2016. VPET: a toolset for collaborative virtual filmmaking. *Siggraph asia 2016 technical briefs*, pp.1–4.
- Srivastava, R. and Sinha, P., 2015. Hand movements and gestures characterization using quaternion dynamic time warping technique. *IEEE Sensors Journal*, 16(5), pp.1333–1341.
- Stakem, F. and AlRegib, G., 2009. An adaptive approach to exponential smoothing for cve state prediction. *Proceedings of the 2nd international conference on immersive telecommunications*. pp.1–6.
- Stiegler, C., 2021. *The 360° gaze: Immersions in media, society, and culture*. MIT Press.
- Strassberger, C. and Sikkema, R., 2018. Democratising mocap: real-time full-performance motion capture with an iphone x, xsens, and maya. *Acm siggraph 2018 real-time live!*, pp.1–1.
- Sun, W., Li, F.M., Steeper, B., Xu, S., Tian, F. and Zhang, C., 2021. Teethtap: Recognizing discrete teeth gestures using motion and acoustic sensing on an earpiece. *26th international conference on intelligent user interfaces*. pp.161–169.
- Świtoński, A., Josiński, H., Jędrasiak, K., Polański, A. and Wojciechowski, K., 2010. Classification of poses and movement phases. *Computer vision and graphics*.
- Tan, W., Wu, C., Zhao, S. and Li, J., 2010. Dynamic hand gesture recognition using motion trajectories and key frames. *2010 2nd international conference on advanced computer control*. IEEE, vol. 3, pp.163–167.
- Tao, T., Zhan, X., Chen, Z. and Panne, M. van de, 2022. Style-erd: Responsive and

- coherent online motion style transfer. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp.6593–6603.
- Taranta II, E.M., Samiei, A., Maghoubi, M., Khaloo, P., Pittman, C.R. and LaViola Jr, J.J., 2017. Jackknife: A reliable recognizer with few samples and many modalities. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. pp.5850–5861.
- Thomas, J., Hall, J.B., Bliss, R. and Guess, T.M., 2022. Comparison of azure kinect and optical retroreflective motion capture for kinematic and spatiotemporal evaluation of the sit-to-stand test. *Gait & Posture*, 94, pp.153–159.
- Tian, Y., Meng, X., Tao, D., Liu, D. and Feng, C., 2015. Upper limb motion tracking with the integration of imu and kinect. *Neurocomputing*, 159, pp.207–218.
- Tormene, P., Giorgino, T., Quaglini, S. and Stefanelli, M., 2009. Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation. *Artificial intelligence in medicine*, 45(1), pp.11–34.
- Tsai, Y.Y., Lin, W.C., Cheng, K.B., Lee, J. and Lee, T.Y., 2009. Real-time physics-based 3d biped character animation using an inverted pendulum model. *IEEE transactions on visualization and computer graphics*, 16(2), pp.325–337.
- Unuma, M., Anjyo, K. and Takeuchi, R., 1995. Fourier principles for emotion-based human figure animation. *Proceedings of the 22nd annual conference on computer graphics and interactive techniques*. pp.91–96.
- Valcik, J., Sedmidubsky, J. and Zezula, P., 2016. Assessing similarity models for human-motion retrieval applications. *Computer Animation and Virtual Worlds*, 27(5), pp.484–500.
- Vander Linden, D.W., Carlson, S.J. and Hubbard, R.L., 1992. Reproducibility and accuracy of angle measurements obtained under static conditions with the motion analysis™ video system. *Physical therapy*, 72(4), pp.300–305.
- Vantigodi, S. and Babu, R.V., 2013. Real-time human action recognition from

- motion capture data. *2013 fourth national conference on computer vision, pattern recognition, image processing and graphics (ncvprimg)*. IEEE, pp.1–4.
- Vicon, 2017. *Shogun*. Available from: <https://www.vicon.com/software/shogun/>.
- Vicon, 2022. *Getting started with vicon shogun*. Oxford, UK: Vicon.
- Vicovaro, M., Hoyet, L., Burigana, L. and O’sullivan, C., 2014. Perceptual evaluation of motion editing for realistic throwing animations. *ACM Transactions on Applied Perception (TAP)*, 11(2), pp.1–23.
- Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P. and SciPy 1.0 Contributors, 2020. SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17(1.2.1), pp.261–272. Available from: <http://doi.org/10.1038/s41592-019-0686-2>.
- Wahyuni, S., Intan, I., Ahyuna, A., Suryani, S. et al., 2021. Motion recognition system with dynamic time warping method using kinect camera sensor. *2021 3rd international conference on cybernetics and intelligent system (icoris)*. IEEE, pp.1–4.
- Walugembe, H., Phillips, C., Requena-Carrion, J. and Timotijevic, T., 2020. Comparing dynamic hand rehabilitation gestures in leap motion using multi-dimensional dynamic time warping. *IEEE Sensors Journal*, 21(6), pp.8002–8010.
- Wang, J. and Bodenheimer, B., 2003. An evaluation of a cost metric for selecting transitions between motion segments. *Proceedings of the 2003 acm siggraph/eurographics symposium on computer animation*. Eurographics Association, pp.232–238.

- Whatman, C., Hume, P. and Hing, W., 2013. The reliability and validity of physiotherapist visual rating of dynamic pelvis and knee alignment in young athletes. *Physical Therapy in Sport*, 14(3), pp.168–174.
- Wilson, A.D. and Bobick, A.F., 1998. Recognition and interpretation of parametric gesture. *Sixth international conference on computer vision (ieee cat. no. 98ch36271)*. IEEE, pp.329–336.
- Witkin, A. and Kass, M., 1988. Spacetime constraints. *ACM Siggraph Computer Graphics*, 22(4), pp.159–168.
- Witkin, A. and Popovic, Z., 1995. Motion warping. *Proceedings of the 22nd annual conference on computer graphics and interactive techniques*. pp.105–108.
- Xia, S., Wang, C., Chai, J. and Hodgins, J., 2015. Realtime style transfer for unlabeled heterogeneous human motion. *ACM Transactions on Graphics (TOG)*, 34(4), pp.1–10.
- Xiao, Q. and Chu, C., 2017. Human motion retrieval based on deep learning and dynamic time warping. *2017 2nd international conference on robotics and automation engineering (icrae)*. IEEE, pp.426–430.
- Xiao, Q. and Liu, S., 2015. Motion retrieval based on dynamic bayesian network and canonical time warping. *2015 7th international conference on intelligent human-machine systems and cybernetics*. vol. 2, pp.182–185. Available from: <http://doi.org/10.1109/IHMSC.2015.73>.
- Xiao, Z., Zhang, J.J. and Bell, S., 2004. Control of motion in character animation. *Proceedings. eighth international conference on information visualisation, 2004. iv 2004*. IEEE, pp.841–848.
- Yabe, T. and Tanaka, K., 1999. Similarity retrieval of human motion as multi-stream time series data. *Proceedings 1999 international symposium on database applications in non-traditional environments (dante'99)(cat. no. pr00496)*. IEEE, pp.279–286.
- Yan, S., Xiong, Y. and Lin, D., 2018. Spatial temporal graph convolutional networks

- for skeleton-based action recognition. *Thirty-second aai conference on artificial intelligence*.
- Yang, H. and Guan, T., 2005. Motion similarity analysis and evaluation of motion capture data.
- Yang, W.T., Luo, Z., Chen, I.M. and Yeo, S.H., 2010a. A method for comparing human postures from motion capture data. *Romansy 18 robot design, dynamics and control: Proceedings of the eighteenth cism-iftomm symposium*. Springer, pp.441–448.
- Yang, Y., Leung, H., Yue, L. and Deng, L., 2010b. Evaluating human motion complexity based on un-correlation and non-smoothness. *Pacific-rim conference on multimedia*. Springer, pp.538–548.
- Yates, D., 2011. *Harry Potter the complete 8 film collection (blu-ray edition)*. Warner Bros.
- Ye, Y. and Liu, C.K., 2010. Optimal feedback control for character animation using an abstract model. *Acm siggraph 2010 papers*, pp.1–9.
- Yu, B., Gabriel, D., Noble, L. and An, K.N., 1999. Estimate of the optimum cutoff frequency for the butterworth low-pass digital filter. *Journal of Applied Biomechanics*, 15(3), pp.318–329.
- Yun, K., Honorio, J., Chattopadhyay, D., Berg, T.L. and Samaras, D., 2012. Two-person interaction detection using body-pose features and multiple instance learning. *2012 ieee computer society conference on computer vision and pattern recognition workshops*. IEEE, pp.28–35.
- Yurtman, A. and Barshan, B., 2014. Detection and evaluation of physical therapy exercises from wearable motion sensor signals by dynamic time warping. *2014 22nd signal processing and communications applications conference (siu)*. IEEE, pp.1491–1494.
- Zalkow, F., Weiß, C., Prätzlich, T., Arifi-Müller, V. and Müller, M., 2017. A multi-version approach for transferring measure annotations between music recordings.

Audio engineering society conference: 2017 aes international conference on semantic audio. Audio Engineering Society.

Zar, J.H., 2014. Spearman rank correlation: overview. *Wiley StatsRef: Statistics Reference Online.*

Zelenskaya, M. and Harvey, L., 2019. Virtual avatars as a tool for audience engagement. *The 17th international conference on virtual-reality continuum and its applications in industry.* pp.1–2.

Zhang, T., Yue, D., Gu, Y., Wang, Y. and Yu, G., 2009. Adaptive correlation analysis in stream time series with sliding windows. *Computers & Mathematics with Applications*, 57(6), pp.937–948.

Zhao, J., Ju, R., Xie, X. and Ye, Y., 2020. Multivariate time series similarity measure based on weighted dynamic time warping. *2020 2nd international conference on image processing and machine vision.* pp.173–179.

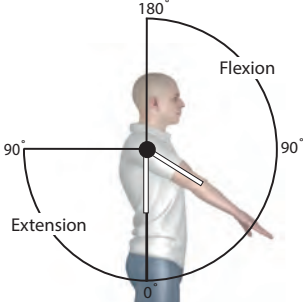
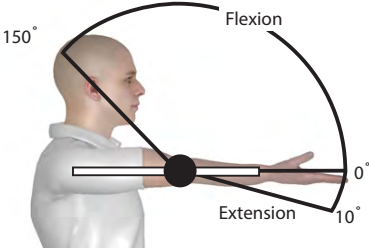
Zhou, F. and Torre, F., 2009. Canonical time warping for alignment of human behavior. *Advances in neural information processing systems.* pp.2286–2294.

Zhu, D., Wang, Z. and Xia, S., 2006. Motion editing with the state feedback dynamic model. *Computer graphics international conference.* Springer, pp.348–359.

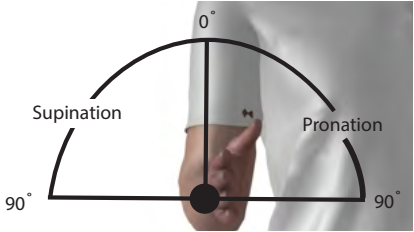
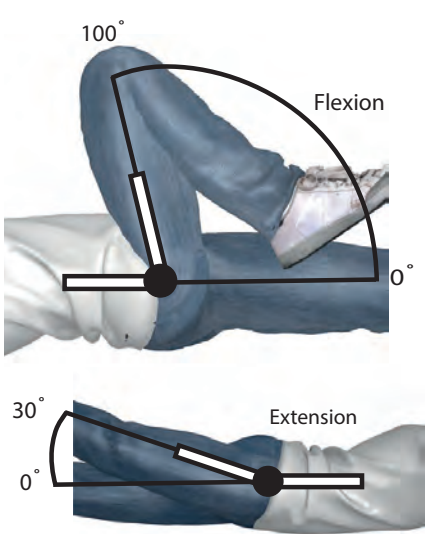
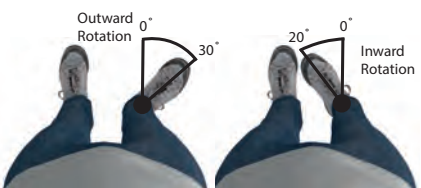
Appendix A

Joint Range of Motions

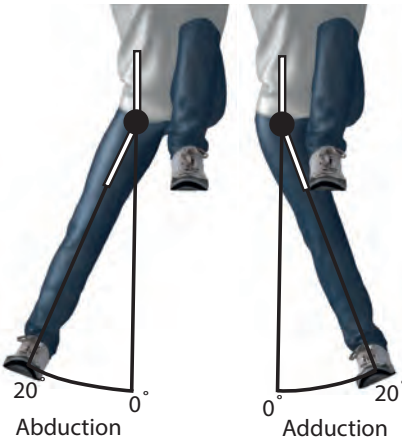
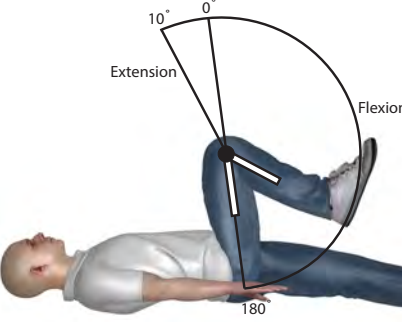
Table A.1: A specification for acceptable ROMs for key joints. The mean, μ , and standard deviation, σ , for each ROM are obtained from [Boone and Azen \(1979\)](#) and used to determine the maximum acceptable ROM for the respective joint axis using equation 5.8.

Shoulder Flexion and Extension		
	Flexion	Extension
	Sampled ROMs: $\mu = 140.7^\circ, \sigma = 4.9$ Max ROM: 164.71° Left Shoulder Constraint: $Y^{min} = -164.71^\circ$ Right Shoulder Constraint: $Y^{max} = 164.71^\circ$	Sampled ROMs: $\mu = 43.7^\circ, \sigma = 5.8$ Max ROM: 77.34° Left Shoulder Constraint: $Y^{max} = 77.34^\circ$ Right Shoulder Constraint: $Y^{min} = -77.34^\circ$
Elbow Flexion and Extension		
	Flexion	Extension
	Sampled ROMs: $\mu = 140.5^\circ, \sigma = 4.9$ Max ROM: 164.51° Left Elbow Constraint: $X^{max} = 164.51^\circ$ Right Elbow Constraint: $X^{max} = 164.51^\circ$	Sampled ROMs: $\mu = 0.3^\circ, \sigma = 2.7$ Max ROM: 7.59° Left Elbow Constraint: $X^{min} = -7.59^\circ$ Right Elbow Constraint: $X^{min} = -7.59^\circ$

Cont'd on following page

Elbow Pronation and Supination		
	Pronation	Supination
	Sampled ROMs: $\mu = 75.0^\circ, \sigma = 5.3$ Max ROM: 103.9° Left Elbow Constraint: $Y^{max} = 103.9^\circ$ Right Elbow Constraint: $Y^{min} = -103.9^\circ$	Sampled ROMs: $\mu = 81.1^\circ, \sigma = 4.0$ Max ROM: 97.1° Left Elbow Constraint: $Y^{min} = -97.1^\circ$ Right Elbow Constraint: $Y^{max} = 97.1^\circ$
Hip Flexion and Extension		
	Flexion	Extension
	Sampled ROMs: $\mu = 121.3^\circ, \sigma = 6.4$ Max ROM: 121.3° Left Hip Constraint: $X^{min} = 121.3^\circ$ Right Hip Constraint: $X^{min} = 121.3^\circ$	Sampled ROMs: $\mu = 12.1^\circ, \sigma = 5.4$ Max ROM: 41.26° Left Hip Constraint: $X^{max} = 41.26^\circ$ Right Hip Constraint: $X^{max} = 41.26^\circ$
Hip Outward and Inward Rotation		
	Outward Rotation	Inward Rotation
	Sampled ROMs: $\mu = 44.2^\circ, \sigma = 4.8$ Max ROM: 67.24° Left Hip Constraint: $Y^{min} = -67.24^\circ$ Right Hip Constraint: $Y^{max} = 67.24^\circ$	Sampled ROMs: $\mu = 44.4^\circ, \sigma = 4.3$ Max ROM: 62.89° Left Hip Constraint: $Y^{max} = 62.89^\circ$ Right Hip Constraint: $Y^{min} = -62.89^\circ$

Cont'd on following page

Hip Abduction and Adduction		
 <p>20° 0° Abduction</p> <p>0° 20° Adduction</p>	<p>Abduction</p> <p>Sampled ROMs: $\mu = 40.5^\circ, \sigma = 6.0$</p> <p>Max ROM: 76.5°</p> <p>Left Hip Constraint: $Z^{max} = 76.5^\circ$</p> <p>Right Hip Constraint: $Z^{min} = -76.5^\circ$</p>	<p>Adduction</p> <p>Sampled ROMs: $\mu = 25.6^\circ, \sigma = 3.6$</p> <p>Max ROM: 38.56°</p> <p>Left Hip Constraint: $Z^{min} = -38.56^\circ$</p> <p>Right Hip Constraint: $Z^{max} = 38.56^\circ$</p>
Knee Flexion and Extension		
 <p>10° 0° 180°</p> <p>Extension</p> <p>Flexion</p>	<p>Flexion</p> <p>Sampled ROMs: $\mu = 141.2^\circ, \sigma = 5.3$</p> <p>Max ROM: 169.29°</p> <p>Left Knee Constraint: $X^{max} = 169.29^\circ$</p> <p>Right Knee Constraint: $X^{max} = 169.29^\circ$</p>	<p>Extension</p> <p>Sampled ROMs: $\mu = 1.1^\circ, \sigma = 2.0$</p> <p>Max ROM: 5.1°</p> <p>Left Knee Constraint: $X^{min} = -5.1^\circ$</p> <p>Right Knee Constraint: $X^{min} = -5.1^\circ$</p>