Automatic Motion Capture Blending with Registration Curves

Masters Thesis

Shane Aherne

MSc Computer Animation

N.C.C.A Bournemouth University

2005

Abstract

This thesis investigates the application of registration curves to solve the issue of automatic motion capture blending. The results are compared and contrasted to linearly blending techniques to evalute their benefits. Current flaws are highlighted while future avenues for improvement are proposed.

Contents

1. Introduction

The creation of automatic and realistic motion blending techniques is an area of active research. Currently much of the blending within the industry is done manually, which can be a very time intensive and costly practice. The value of automatic motion capture blending means the time required to carry out a similar task can be substantially minimized.

The technique of automatic blending however, has not been without its difficulties , amongst which are technique sensitivity and unreliability . Complications arise with a greater dissimilarity of input motions, leading to a higher level of unpredictability in the resulting blend.

Additionally, the human eye is quite sensitized to flawed motion blends, due to its constant exposure to human motion.

The purpose of this research project is to analyze the complexities of motion capture blending and to integrate solutions which have been proposed in recent years.

2. Previous Work in this Area

Motion Signal Processing (Bruderlin, Williams 1995) introduces a multi resolution motion filtering method, with multitarget motion interpolation using dynamic time warping, wave shaping and motion displacement mapping. It allows existing motions to be modified and combined interactively, at a higher level of abstraction than conventional system support.

Online Locomotion Generation (Park *et al.* 2002) based on Motion Blending presents an incremental scheme for time warping to align various clips of different speed. A novel scheme for joint blending is also introduced. The algorithm is based on scattered data interpolation and generates on the fly locomotion.

Motion Graphs (Kovar *et al.* 2002 b) present a method of creating realistic, controllable motion. Using an initial library of original motion capture data, automatically generated motions are created. These are combined to direct an avatar along arbitrary paths. Several of the motion blending techniques utilized in (Kovar , Gleicher, 2003) are implemented in this paper.

'Verbs and Adverbs: Multidimensional Motion Interpolation' (Rose *et al.* 1998) introduces a technique for interpolating between motion capture segments. These motions are calculated through expressions or control behaviors (e.g. turning or going uphill). Such motions are referred to as "verbs" and the parameters which control them are "adverbs". A "verb graph" is formed by combining a number of verbs, allowing an animated skeleton to exhibit a substantial array of behaviours. Inverse Kinematics are implemented in order to avoid the issue of "foot sliding" and "foot hopping".

3. Approach Adopted and Motivation

The approach adopted after lengthy research was to implement registration curves (Kovar , Gleicher, 2003). A registration curve is an automatically constructed data structure that encapsulates relationships involving the timing, local co-ordinate frame and constraint states of a number of different input motions.

All this is created with raw motion parameters (root position and each frames joint angles), without requiring IK or physical information.

Aspects of registration curves have previously been implemented (combined with motion graphs) for the crowd simulation scenes in the movie '*Troy*' (Peterson, 2004), with software developed by the Research and Development team at the *Moving Picture Company* in London.

After personal correspondence with Michael Gleicher, one of the papers authors, it was confirmed that registration curves were being used successfully in industry and provided a good platform for further research and development.

The decision was taken to allow a full user interface to enable certain parameters to be adjusted by the user. Also, full visual representation of the algorithms would be accessible to the user.

This provides the user with more information on which parameter changes can be based. For example, the slope limit of the Dynamic Time Warp path may need to be adjusted depending on the desired result, or the type of motions inputted.

The application itself is programmed in C++, and utilizes OpenGL. This serves to make it easier to create cross platform versions of the software in the future.

Finally, it was decided to import and export files in the FBX file format. Alias FBX is a platform-independent 3D authoring and interchange format that provides access to 3D content from most 3D vendors and platforms, including applications such as Maya and XSI. The code therefore is based around the FBX SDK, freely available for both Windows and Linux platforms from its vendor Alias.

Other advantages to utilizing FBX, include the fact it is not restricted to holding just motion capture information (for example BVH holds just information relating to motion capture data).As a result, other features could be exported from the application, such as IK reach constraints and motion markers, all aiding in the final cleanup of the resulting blended motion.

4. Registration Curves Application Solution

4.1 Alignment Curves

4.1.1 Overview

Coordinate Frame Alignment aims to provide a solution to discontinuities with standard orientation averaging schemes like spherical linear interpolations. With such interpolations, linear blending can fail even when the two sequences are actually quite similar. For example, with two walking motions approaching each other from different angles, the blended root collapses, and sudden flipping of the character occurs when angles change by more than 180 degrees.

The solution provided observes that local motions remain constant despite rotations around the vertical y axis and translations along the floor plane. The local joint rotations of a skeleton will remain the same no matter where the root has been aligned to. Therefore, two motions can be translated to one abstract rig, which in turn can be rigidly translated along the floor plane. Different frames can have different blending weights to determine where the blended root should be positioned.

Once the abstract frames position has been determined through a rigid 2D transformation, a Dynamic Time Warp grid is constructed.

4.1.2 **Related Papers**

In (Park et al. 2002) at an equal generic time, the corresponding postures of example motions is blended rather than applying rigid 2D transformations.

Not only the joint angles are blended, but also the root position and orientation of the postures. This resulting posture is then adopted to a given trajectory.

4.1.3 Algorithms

To implement automatic alignment , three steps are undertaken. The first step, is extracting a series of frames from both segments of motion capture. Within this application seventy frames are extracted.

Following this, each frame is converted into a point cloud by attaching markers to every joint of the skeleton. Two large point clouds are now created.

Finally, the optimal sum of squared distances between corresponding points is computed, after a rigid transformation of the second point cloud.

The resulting formula is:

$$\mathbf{D}(\mathbf{F}_1, \mathbf{F}_2) = \min_{\theta, x_0, z_0} \sum_{i=1}^{n} w_i \|\mathbf{p}_i - \mathbf{T}_{\theta, x_0, z_0} \mathbf{p}'_i\|^2$$

where pi and pi' are respectively the ith point of the first and second point cloud. T0, x0 z0 is a rotation by Θ around the y axis, with the translation (Xo, Zo)

Weighting may be used to preferentially weight joints differently. The implementation used in this application used equal weighting for every joint.

The formula used to calculate the rotation of the second point cloud group is

$$\theta = \tan^{-1}\left(\frac{\sum_i w_i(x_i z_i' - x_i' z_i) - (\overline{x}\overline{z'} - \overline{x'}\overline{z})}{\sum_i w_i(x_i x_i' + z_i z_i') - (\overline{x}\overline{x'} + \overline{z}\overline{z'})}\right)$$

The same formula is also used successfully in (Kovar *et al.* 2002 b)

Once the rotation around the y axis is determined a translation along the floor plane (in X and Y) must be enforced. This is achieved by calculating a translation vector using the following:

$$x_0 = \overline{x} - \overline{x_i'}\cos\theta - \overline{z_i'}\sin\theta$$
$$z_0 = \overline{z_i} + \overline{x_i'}\sin\theta - \overline{z_i'}\cos\theta$$

where :

$$\overline{\alpha} = \sum_{i=1}^{n} w_i \alpha_i$$

These formulas have been applied successfully in the application, allowing an optimal distance algorithm to be computed and applied to the Dynamic Time Warp grid outlined in the next section.

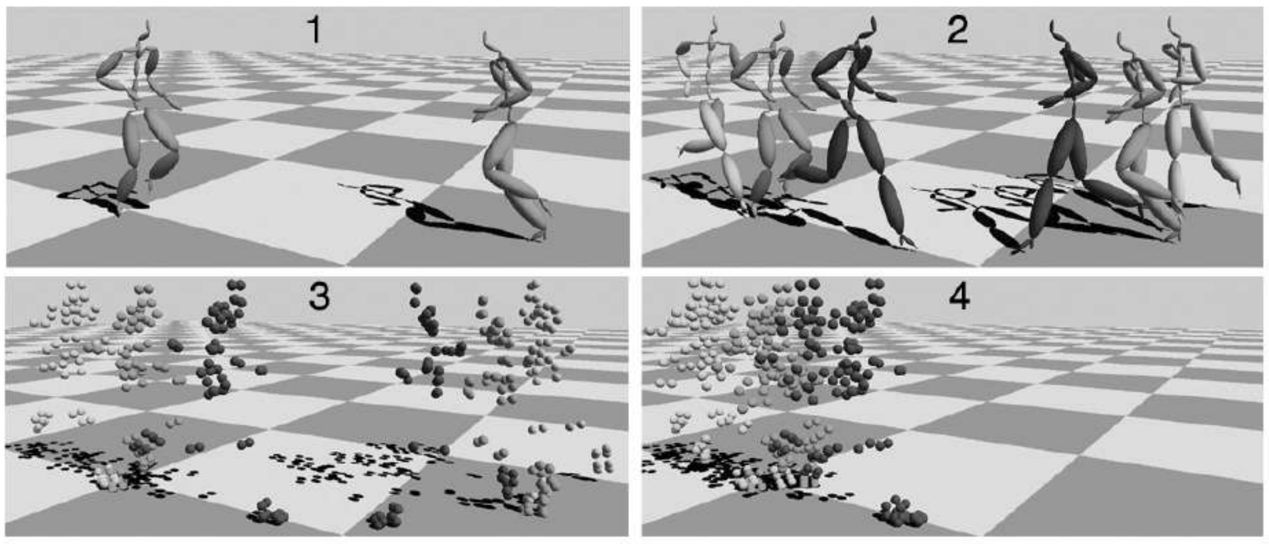


Figure 1:

The stages of coordinate alignment:

1 – The original input motions.

2 – Extracting a series of frames.

3 – Creating point clouds.

4 – Applying the alignment algorithms

Different formulas are however applied to the exporting section of the application. The above algorithm applies to point data, whereas the exporting feature of this application (which is built around the FBX SDK) is based on building rigs from rotational data.

To solve this issue, an amended approach was taken for the exporting of the blended sequence. The approach adopted was taken from (Park *et al.* 2002) , whereby all the joint angles, root positions and orientations are blended to maintain the rigs posture. The rig root is then offset to reflect the motion path, using the angular and translational information already computed (using the formulas outlined in this section), to give a visual representation of alignment and to compute the distance matrix.

4.2 Dynamic Time Warping

4.2.1 Overview

Dynamic Time Warping is traditionally associated with speech recognition. The idea states that utterances of the same words or sentences may vary in time. The solution applied warps the timing of the two inputted sequences, by “stretching” and “shrinking” them. The optimal time warp path is that which gives the minimal dissimilarity between the two patterns.

Dynamic Time Warping works on a distance matrix. Where each cell of the matrix represents the local distance similarity between each section or frame of the two inputs. In order to maintain a usable path, several restrictions are applied to how the path can behave.

Endpoint constraints ensure that the path finishes on the same final frame of each motion. Slope limits ensure a smoothness to the warped path. monotonicity conditions mean a path must constantly go forward in time. Continuity means a path cannot skip over gridpoints to reach another goal.

4.2.2 Related Papers

Text-dependent speaker recognition methods have relied heavily on template matching techniques such as Dynamic Time Warping. The input utterance is represented by a sequence of feature vectors. The timing of the input utterance and each reference template are aligned using the Dynamic Time Warping algorithm. The degree of similarity between them, which is the sum of distances from the beginning of the utterance is calculated.

More recently Hidden Markov Models (HMM) have been introduced as an extension to the DTW-based method. HMM efficiently models statistical variation in spectral features, leading to significantly better recognition system (Gregersen *et al.* 2000)

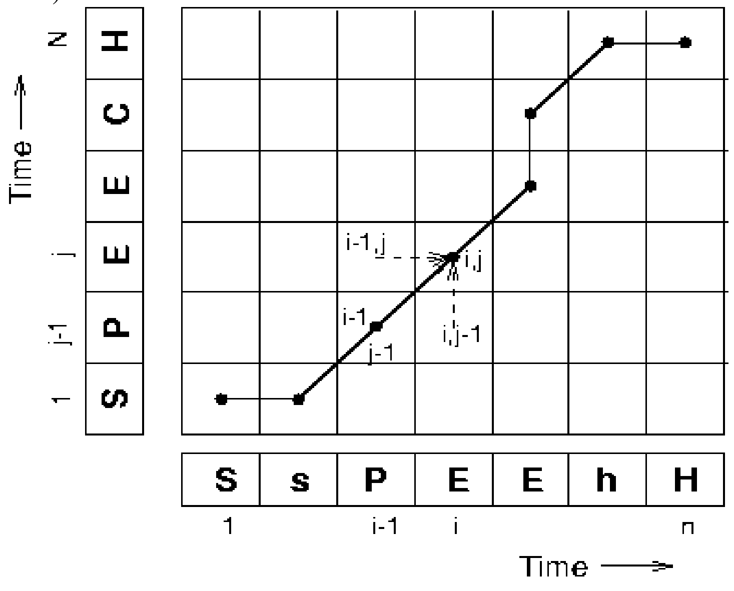


Figure 2:

Illustration of a time alignment path between a template pattern

` ` SPEECH' ánd a noisy input ` ` SsPEEhH

In (Sederberg, Greenwood , 1992) a method for determining where to insert vertices so that the shape transformation is achieved with the least cost is introduced. This technique is based on creating a distance matrix grid and creating a path through it. Several algorithms were presented to achieve such a path with the minimum implementation cost.

This approach was adopted in (Bruderlin, Williams 1995) as it provides a globally optimal solution, by visiting every node in the graph once. Within this implementation, more restrictive rules were enforced to maintain a smoother interpolation method.

In (Kovar *et al.* 2002 b) , to avoid complications and inconsistencies with computing differences between vector norms, the dynamic time warp approach from (Bruderlin, Williams 1995) was implemented. This allowed fixed weights to be associated with more important joints, and provided a far smoother interpolation over a series of frames.

Within (Park *et al.* 2002) a different time warping method is introduced. This incremental time warping employs a notion of keytimes which are important instances of a motion, such as heel strikes and toe-offs. Based on these keytimes, time warping is defined as a piecewise linear mapping of actual time onto generic time.

The basic idea is to blend the change rates of actual times with respect to the generic time and to accumulate the weighted change rate for incrementally updating the current actual time of the blended motion. This technique guarantees the monotonicity of the blended time warping function.

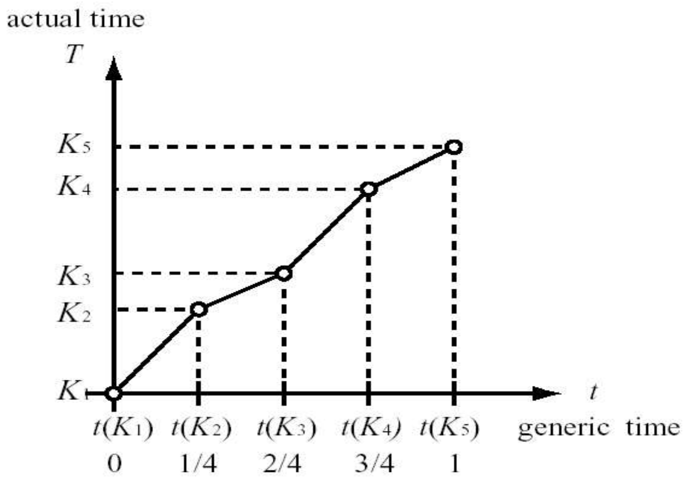


Figure 3:

The piecewise linear mapping between the actual time T and the generic time t, used Online Locomotion Generation Based on Motion Blending

4.2.3 Algorithms And Explanations

4.2.3.1 Creating the grid:

Once the 2D rigid transformation in section [4.1.3] has been implemented, the optimal sum of squared distance between corresponding points of the two poses is calculated.

A grid is created where columns correspond to frames from the first motion (m) while rows correspond to frames from the second motion (n). The grid is therefore size m,n.

Using the average distance from each frame, every corresponding frame of the two motions occupies a cell within this grid. The cell value is calculated by measuring the average distance between the two frames, it is then normalized.

Within the application, each cell is assigned a grey scale value depending on the similarity of the two frames. A dark cell reflects high similarity, while a light cell reflects poor similarity.

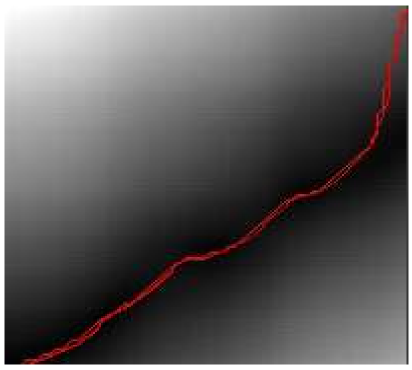


Figure 4:

The dynamic time warp path of a walking and sneaking motion

4.2.3.2 Dynamic Time Warping – Creating a DTW path

Once the distance matrix grid has been created, the least cost optimal path though the grid must be computed. This application uses a *priori* , whereby the endpoint of the path must be time aligned – that is, the last frames must correspond. Other, less restrictive paths may also be computed, allowing a path which simply minimizes the total cell cost.

There are several methods to determine the least cost path. All possible paths could be computed, but this is extremely inefficient as the number of possible paths is exponential to the length of the inputs.

Several conditions are imposed on the path. Continuity, causality and slope limits. These are explained in more detail later in this section.

The resulting algorithm is referred to as *Dynamic Programming*. The key idea to *Dynamic Programming*, is that at a point (i,j) the next chosen direction is the lowest cost choice $(i-1, j-1)$, $(i-1, j)$ or $(i, j-1)$. This algorithm is guaranteed to find the lowest distance path through the matrix, while minimizing the amount of computation. Therefore, if $D(i,j)$ is the global distance up to (i,j) and the local distance at (i,j) is given by $d(i,j)$:

$$D(i,j) = min [D(i-1, j-1), D(i-1,j), D(i, j-1)] + d(i, j)$$

The implementation of actually creating the Dynamic Time Warp within the application initially sets the cell $D(1,1) = 0$, with a new multi-dimensional array containing the least global cost to each cell on the grid.

Leading on from this, the least global cost to every cell in the grid is calculated. Once this has been evaluated, the process of back tracking is executed to create a pattern with minimal total distance.

The total global cost is stored in the top most cell of the last column.

The following is the algorithm outline for this process:

```
• Initialization
        $D(1,j,K) = \sum n=1\ d(1,n,K)$
• Warping
        for $2 \le i \le I$ , calculated
            for $1 \le k \le K$ , calculated
                $D(i,1,k) = d(i,1,k) + min\ [D(i-1, l,k), min\ [D(i-1, J(k'), k']]$
                    for $2 \le J \le j(k)$ , calculated
                        $D(i,j,k) = d(i,j,K) + min\ [D(i-1, j,k), D(i-1, j-1,k), D(i, j-1, k)]$
                    End j
            End K
        End i
• Path Back Track
        $Rend = min\ k'\ [D(I,J(k'), k')]$
```

Continuity

This defines the legal increments in a path through the distance matrix. It is defined to ensure that the path does not skip any information.

Extra parameters could be set so a move by more than one consecutive cell could be enabled. This is applicable mainly to some forms of speech recognition and was not necessary for this application.

Causality/Monotonicity Conditions

A path is not allowed to go back in time. This corresponds to not allowing the path to go left or down in the distance matrix.

Slope Limits

This defines the maximum consecutive horizontal or vertical step which are legal within the application. Each path move is checked to ensure it has not broken the user defined slope limit. If the slope limit has been reached the next least cost move is undertaken, and the slope limit count of the illegal move is set to zero.

The graphical user interface of the application allows the user to determine a slope limit, which can be interactively adjusted. Through testing a slope limit of 3 seems most stable, although certain (more rare) conditions had a better result with a higher slope limit (up to a maximum of fifteen)

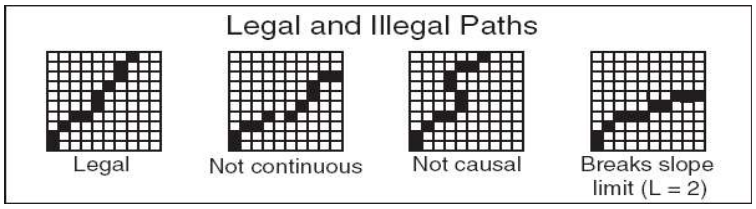


Figure 5:

Legal and illegal dynamic time warp paths

4.3 Constraint Matching

4.3.1 Introduction

With Alignment and Dynamic Time Warping, there exists a high probability that all constraints on the blend may not be satisfied. As a result a certain amount of 'Foot Sliding' or 'Hopping' may exist. It is therefore desirable to determine what the constraints should be. Using implementations presented in certain papers (Kovar et al. 2002 a) or using an application such as MotionBuilder or Maya, these constraints can be enforced. Even if there are no aesthetic issues with the blend, constraints may be used to further edit the results.

Different motions may have different constraint types. A run and a walk, even after Dynamic Time Warping won' have matching constraints. As a walk always has one foot planted, while a run sequence contains flight stages. Adding to this, it might be desirable to blend two motions that have a different number of constraints.

The algorithms presented within (Kovar, Gleicher 2003) aim to create constraint matches, where each constraint match contains a set of related constraints, one from each motion.

4.3.2 Related Papers

Within (Gleicher, Litwinowicz , 1996) a technique to allow the adaptation of motions to meet new goals, while retaining much of the quality is presented.

Numerical constraint techniques alter motions to meet new goals while preserving as much of the original motion as possible. Work from previous techniques is combined to achieve this, including motion signal processing methods (Bruderlin, Williams 1995) and constraint based direct manipulation (a generalization of Inverse Kinematics)

The idea behind this approach is to target the smallest change to a motion in order to meet a set of specified goals.

Inverse Kinematics uses a solver to compute configurations to meet specific goals, yet as constraints are treated independently, they do not take each other into consideration. The method presented within (Gleicher, Litwinowicz , 1996) integrates a displacement curve that uses cubic interpolation. This gives the illusion of the character planning its motion ahead.

Constraints developed within registrations curves can be adapted to fit several papers specializing in controlling avatars. In (Kovar et al. 2002 a) a series of operations are necessary to implement footpath constraints. The paper though ,works from the assumption that the frames where the feet are to be constrained are known, and the points where the feet are to be constrained is known. Both of these are delivered by registration curves.

An amount of Inverse Kinematics and position averaging are used to achieve the foot goal. While root adjustment is necessary on certain constraints. Even after this implementation, the final data must go through a degree of clean up to eliminate foot floor penetration, and to smoothen any discontinuities which may appear.

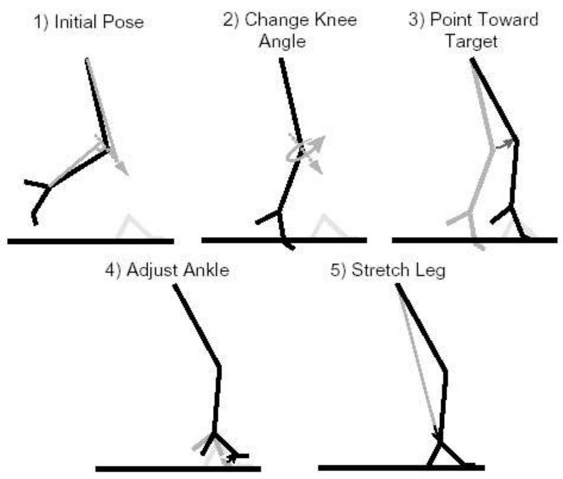


Figure 6:

The five steps for attaining a target ankle position and orientation

4.3.3 Algorithms

The constraint matching algorithm concentrates on footplant constraints, judging a constraint as a static heelplant over a series of frames. Each kind of constraint is treated independently, in this case the left and right footplants.

After the dynamic timewarp grid has been completed, the duration of each constraint is computed. These constraints are then compared against and connected to corresponding constraints of the second motion which fall within the same time range. This logic is based on the heuristic that corresponding constraints ought to occur at similar points in the motions.

Once connections have been established a series of constraint elimination and splitting may occur. Any constraints which do not belong to a constraint union are deleted, while constraints with two or more matches are split. This split is implemented through the constraint splitting algorithm explained in section [4.3.3.1]

After all constraints have been matched, exactly one match from each motion per union must exist. These constraints are exported from the application as IK reach markers.

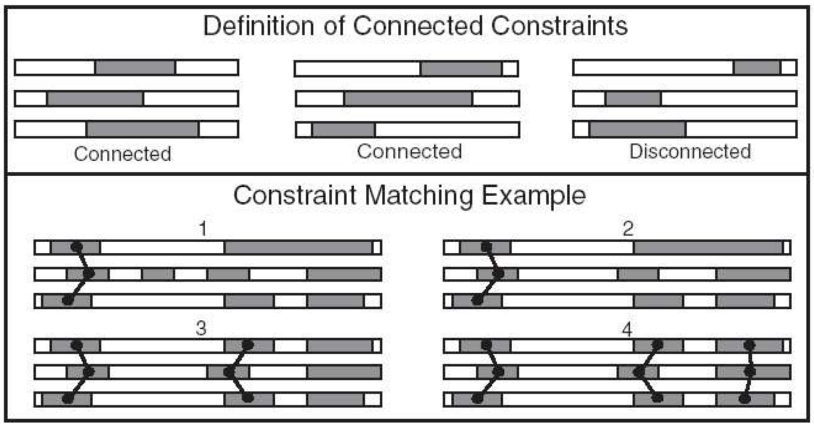

Definition of Connected Constraints
Connected
Connected
Disconnected
Constraint Matching Example
1
2
3
4

Figure 7:

Top : Examples of connected and disconnected constraints.

Bottom: The steps taken by the constraint matching algorithm

4.3.3. 1 Constraint Splitting

This algorithm is essentially based on determining what constraints need to be split, and which gap should be introduced to a constraint which needs to be split.

The decision on what needs to be split is based on subsumption. Subsumption works on the principle:

Ci,j subsumes *Cj,i* if:

$$C^e_{i,1} > C^s_{j,2} \text{ and } C^s_{i,2} > C^e_{j,2}$$

Where Ci,j is the jth constraint of the ith motion, which is active over the interval [Csi,j, Ce i,j

In other words:- a subsuming constraints endpoint , overlaps the second constraints start point, and the end point of the second constraint does not overlap the start of the next constraint in the second motion.

To satisfy this need for handling the splitting within the application, all constraints were partitioned into two arrays. The first array S1 contains all those constraints which subsume others, while the second array S2 contains all those constraints which are subsumed.

Following this, an iterative process of checking through array S2 to ensure it does not subsumes any other elements in S2 is carried out. If the constraint is found to subsume another elements it is switched to S1.

It should be stated that if S2 ends up empty, then nothing should be split and all elements should be returned to S2.

After this process, it must be determined where and by how much each gap needs to be positioned. Within the application, each constraint in S1 is treated individually. Each one of its corresponding constraints in S2 determines through voting, how its parent in S1 should be split.

The elements of S2 are examined to determine their overall ratio, compared to the total length of the constraints, including the gaps separating them. This ratio of gaps to constraints is directly applied to S1. Through experimentation, this approach has given quite satisfactory results, with corresponding sizes of constraints matching quite well.

The array containing the original elements (with the exception of those eliminated where no matches were found), is updated with the new split constraints within S1.

This final array, provides fully matched elements which can now be exported with the blended sequences, allowing further editing if desired.

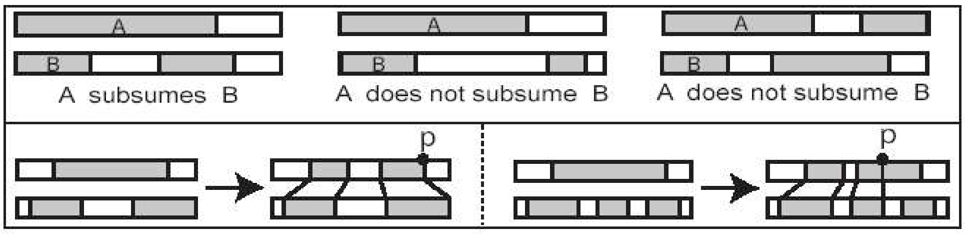
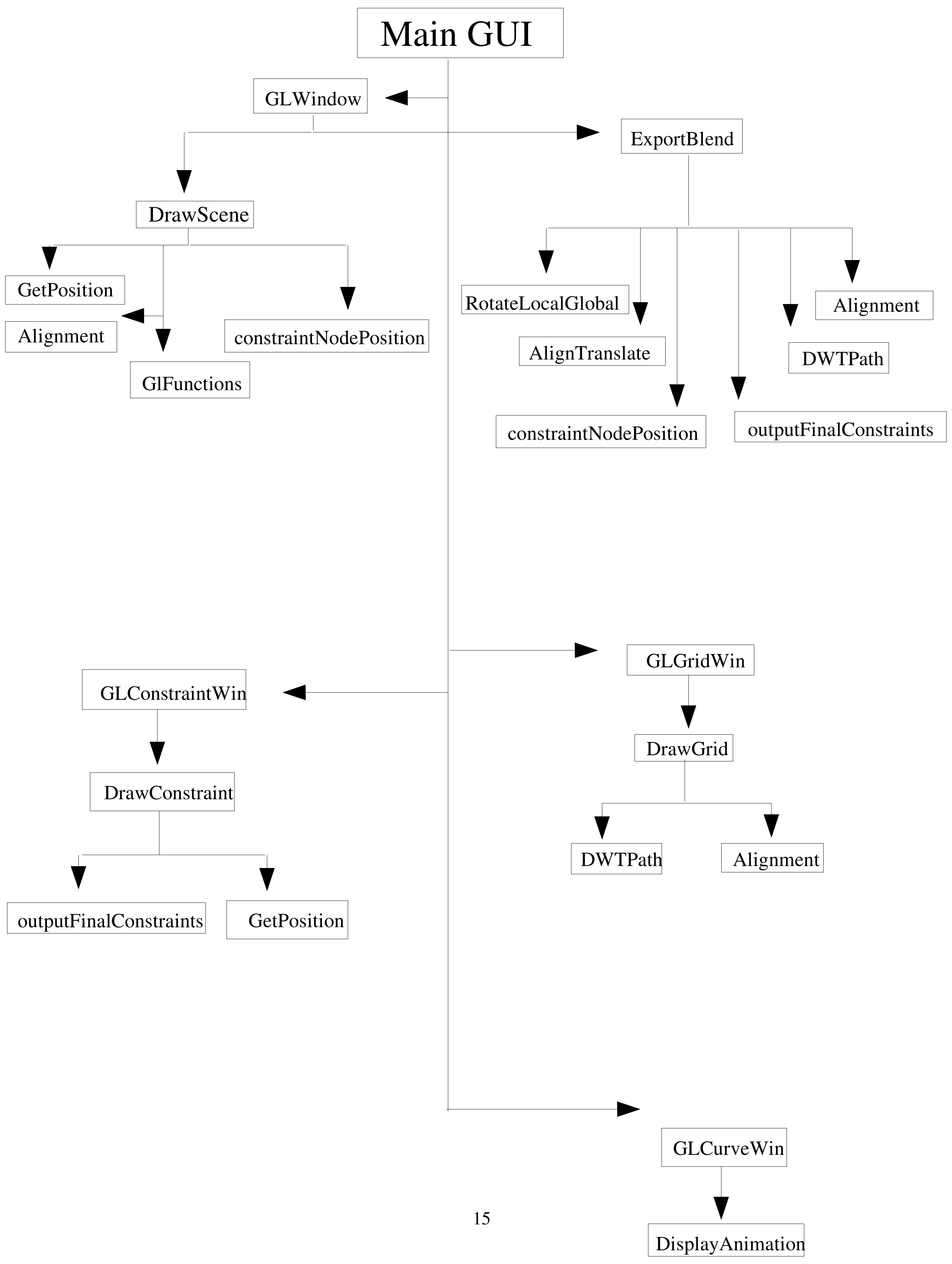


Figure 8:

Top section: Gives examples of subsumption. Bottom section: Adding a gap to a constraint

5. Implementation

5.1 Class Structure



5.2Classes

GLWindow

Objective: Provides an OpenGL viewer for Motion Capture, within the GUI container. Calls the QT and FBX libraries. Also handles request for the User Interface, such as Zoom, Play, Import and Export

Methods: scaleDown(), scaleUp(), changeFilename(QString),secondFilename(QString), changeSceneStatus(); animate(), exportBlendMocap(), rotateY(int newYValue), rotateX(int newXValue), flipRot(void), exportBlendOption (int blendOption), exportBlendWeight(int newWeight);

AlignTranslate

Objective: Sets up the alignment vector for the skeleton root

Methods: writeAnimationAlign(), writeChannelsAlign(). void writeCurveAlign(), float returnAlignTranslate(int axis);

constraintNodePosition

Objective: Creates 2 arrays, holding the global positions of node 4 and Node8, which are the left and right ankle nodes respectively

Methods: addConstraintNodePos(), getGlobalPosition()

DrawConstraint

Objective: Takes in the scene information of the two segments of motion capture, Creates a Global and Local Position, Passes Node and Time information to the GetPosition class

Methods: DrawNodeRecursiveConstraint(), SumOfFunctionConstraint(), SumOfFunctionEveryFrameConstraint(), AnglePerFrameConstraint(), DrawNodeRecursiveAtPoseConstraint(), addToSumOfConstraint(), addToSumOfEveryFrameConstraint(), calculateFrameAngleConstraint(), DrawNodeConstraint(), DrawSkeletonConstraint(), DrawSecondSkeletonConstraint(), DrawSecondSkeletonTranslateConstraint(), DrawConstraintQuads(), DrawSplitConstraintQuads(), drawConstraintLines(), DrawNewSplitConstraintQuads4(), DrawNewSplitConstraintQuads8(), splitEachConstraintInS14(), splitEachConstraintInS18()

DrawGrid

Objective: Creates and displays the distance matrix, between the two motion capture segments

Methods: SumOfFunctionGrid(), SumOfFunctionEveryFrameGrid(), addToSumOfGrid(), addToSumOfEveryFrameGrid(), calculateFrameAngleGrid(), translateSecondPosition(), DrawGridQuads()

DrawScene

Objective: Takes in the scene information of the two segments of motion capture, Creates a Global and Local Position , Passes Node and Time information to the GetPosition class

Methods: DrawNodeRecursive(), SumOfFunctionEveryFrame(), DrawNodeRecursiveAtPose(), addToSumOfEveryFrame(), DrawNode(), DrawNode(), DrawMarker(), DrawSkeleton(),DrawSecondSkeleton(), DrawSecondSkeletonTranslate()

DrawPath

Objective: Creates, returns and displays the Dynamic Time Warp Path through the distance Matrix

Methods: DrawDWTPath(), getGridDistance(), traceBack(), drawLine(), createKeyArray(), returnPathCount()

ExportBlend

Objective: Takes in an ASCII .fbx file, and then creates an export script based on the information from these import files. This export script uses registration curve algorithms created within the application to create a blend sequence

Methods: ExportBlend()DisplayContent(), DisplayContent() , DisplayTarget(),DisplayTransformPropagation(; , DisplayGeometricTransform(), DisplayMetaData(),WriteSkeleton(), closeWriteScene(), openExportFile(), closeExportFile(),writeNodeRoot(),writeNodeChild(),writeAnimation(), writeAnimationRootHierarchy(), writeAnimationHierarchy(), writeChannels(), writeCurve(),resolveRotationLocal(),createHipArray(), closeWriteAnimation(),setUpAlignment(), determineConstraintFrames()

GetPosition

Objective: Takes in the 2 Motion Capture Nodes, and timing/keyframe information, Then return the global position of each node

Methods: GetLocalPosition(), GetSecondLocalPosition(), GetLocalPosition(), GetGlobalPosition()

GLConstraintWin

Objective: The GlConstraintWindow and initialization of Constraint matching

Methods: initializeGL(), changeFilename( ), secondFilename( )

GLCurveWin

Objective: Displays the curves of the animation channels in an OpenGL Window. Called by the main GUI

Methods: initializeGL(), changeFilename( ), secondFilename( )

Glfunctions

Objective: Class is passed each skeletal node glocal position and draws each point, for the resulting point cloud. Also draws foot plant Constraints

Methods: GlPointCloud(), GlPointCloudTranslate(), GlDrawMarker(), GlDrawLimbNode(), GlFootPath()

GLGridWin

Objective: The GlGridWindow and initialization of the distance matrix grid, and dynamic time warp path

Methods: initializeGL(), changeFilename( ), secondFilename( )

outputFinalConstraints

Objective: Calculate and output the final constraint matches

Methods: outputFinalConstraints(), checkFrameConstraint()

editBlendFile.pl

Objective: Edited file to override MotionBuilder bug, New File is called main.cxx and in bigblendExport folder, Compile and run ExportBones followed by the filename you wish the FBX file to have, e.g. ExportBones newFBXBlend

6. Results and Discussions

6.1 Capturing and selecting the input motions

A series of segments were selected for experimentation with the application. As the paper (Kovar , Gleicher, 2003) concentrates mainly on human locomotion, this became the main focal point for the testing.

Each sequence has the mocap actor moving across the floor plane of the motion capture frame rig. The distance covered is about 3.5 meters, so each sequence is relatively short.

A standard walk cycle was recorded from different angles (in order to test the co-ordinate alignment). Following this a much slower sneaking motion was captured, once again from different directions. A skip and jog motion were also captured to allow experimentation with quicker movements. Due to the restricted physical space available, these sequence are quite short. The skip motion resulted in fifty five frames, while the jogging motion was forty five frames. The standard walking motion was eighty frames in length.

Several other motions were also captured for the purpose of experimentation. A large swinging boxing movement, and a sharp quick hit were captured. Several different kicking styles were also acquired

All the motion capture for this application was captured at AccessMocap, Bournemouth University.

Figure 9:

Directing the actor at the motion capture session.

The motion capture area was 3 x 3 x 3 Meters, leaving only a small area for movement

6.2 Splitting the motion capture

Once the motion capture sequences were cleaned, the process of splitting the files into specific sequences was undertaken. This splitting was achieved by creating a short extension to the application, which creates new individual FBX files with information taken from the larger sequence. The segments to be split are specified by the user. Once the individual motions have been segmented they can be imported back into the application for motion blending.

6.3 Registration curves compared to Linear blending

Several different experiments were carried out to investigate and compare the blended motions of the registration curves to that of regular linear blending. Both transitional and weighted interpolations were tested.

Motions of some reasonable similarity were tested, as it must be appreciated that no blending method is perfect, so it is important to understand when a blend is likely to succeed and when it will probably fail.

As a general rule, the difference between the types of captured motion had a direct relation with the success rates of the blends created from the application.

The initial tests were based around creating a 50:50 interpolation between two walking motions from different directions. The linear blend between the two movements had quite poor results, whereby the resulting motion seemed to wave and stumble uncomfortably. The translation along the floor plane was also poor as the motion path was a direct interpolation between the two segments traveling in independent directions.

The blend created with registration curves resulted with a far more aesthetically pleasing resolution. The path of motion between the two sequences was based solely on the motion path of the first input motion (co-ordinate alignment), while the joint interpolations were handled by the Dynamic Time Warping. The DTW, with its optimal path, warped the timing of the two sequences on top of each other in order to maximize the amount of similar frames occur simultaneously.

This worked best when the sequence came closer to its midpoint. At the start and end of the blend corresponding frames are forced to match, giving a poorer similarity. Overall, the test proved quite successful in matching the two motions, creating a more fluid and realistic movement.

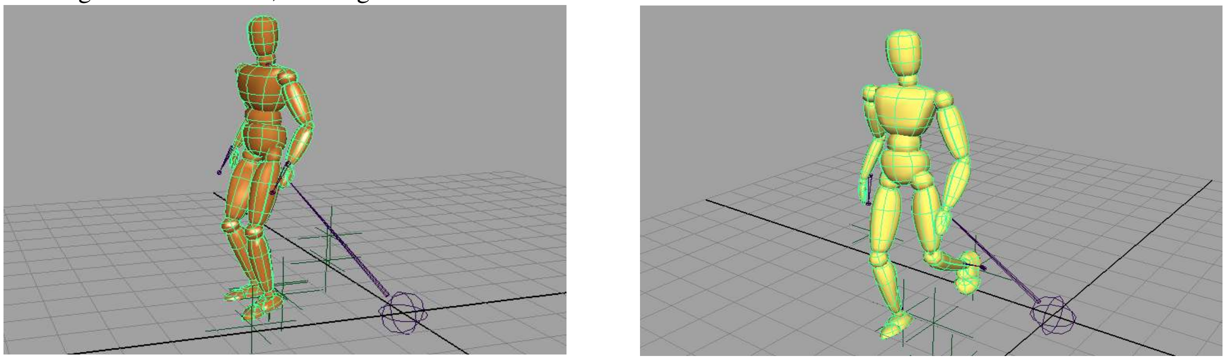


Figure 10:

Taken from a corresponding frame of both blends. The legs on the linear blend (left) float, while the dynamic time warped blend (right) maintains a walking motion

A transition between the two segments was also tested. The linear blend in these tests floated uneasily during the transition phase. Registration curves however gave good results, matching the foot plants and walk cycle of the two motions. Therefore provided continuity during the transitions.

With both tests though, a degree of foot sliding and foot hops were evident. This required constraint editing within the 3D animation package Maya.

The constraint matching within the application created a series of markers positioned at areas where the feet should be constrained. These markers can be weighted at certain frames to force the left or right ankle onto that marker. Restrictions within Maya though, only allow a weighting to be effective when it is matched against another point constraint.

To solve this issue an extension to the application was written. This extra feature attaches another marker to the ankle joint in the skeletal rig. This marker in turn, which follows the path of the original motion, can be used to weight the ankle between a foot plant constraint and the original position of the ankle.

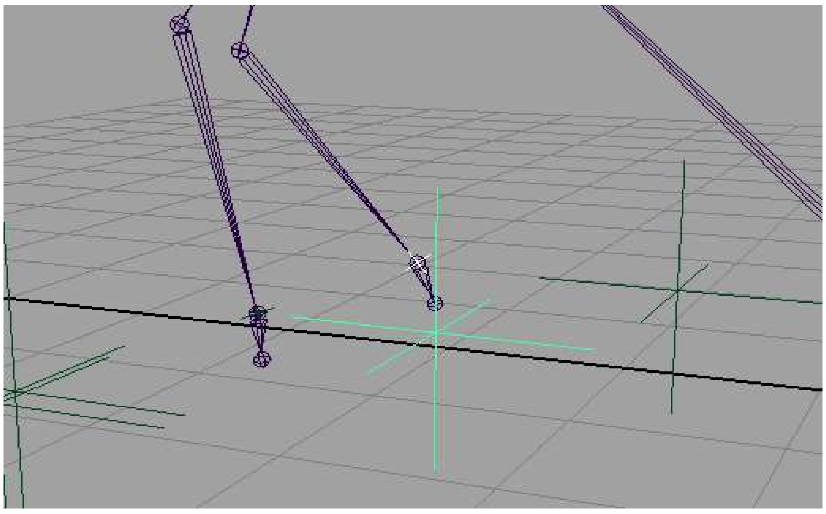


Figure 11:

A ground constraint marker, and a marker attached to the skeletons ankle

As a result of this extra feature, the cleanup time of the blended motions was substantially reduced.

6.3.1 Blending different types of motion

Walking and sneaking blends provided good results, with effective transitions using registration curves. Linear blending with the same base motions can remain believable for a short period, but fall out of sync quite regularly. This creates results which are sometimes floating or occasionally erratic.

Once constraint matching was applied to the walk/sneak blend, an array of believable transitions and weighted interpolations were created.

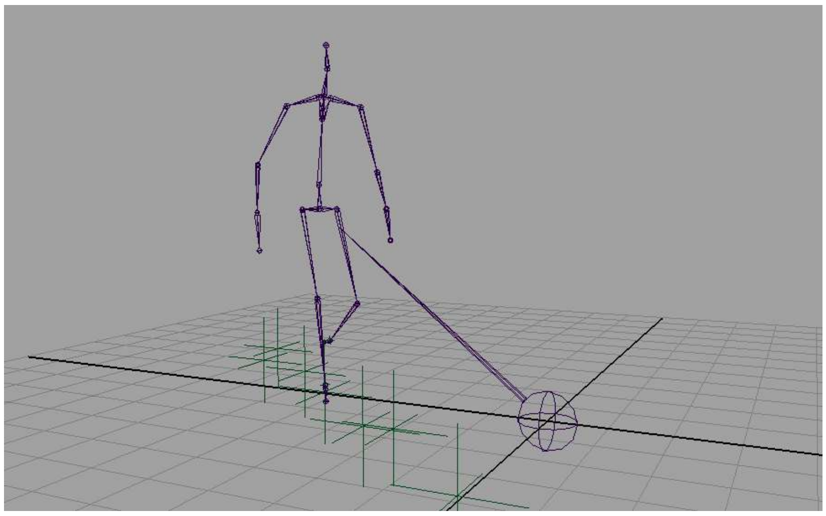


Figure 12:

A walk sneak blend exported from the application. The ground markers represent foot constraints at certain frames.

Further tests were carried out on skipping to sneaking transitions and interpolations. Although registration curves provided far better matches than its linear counterpart, manual editing was needed to eliminate irregularities and apply constraints increased.

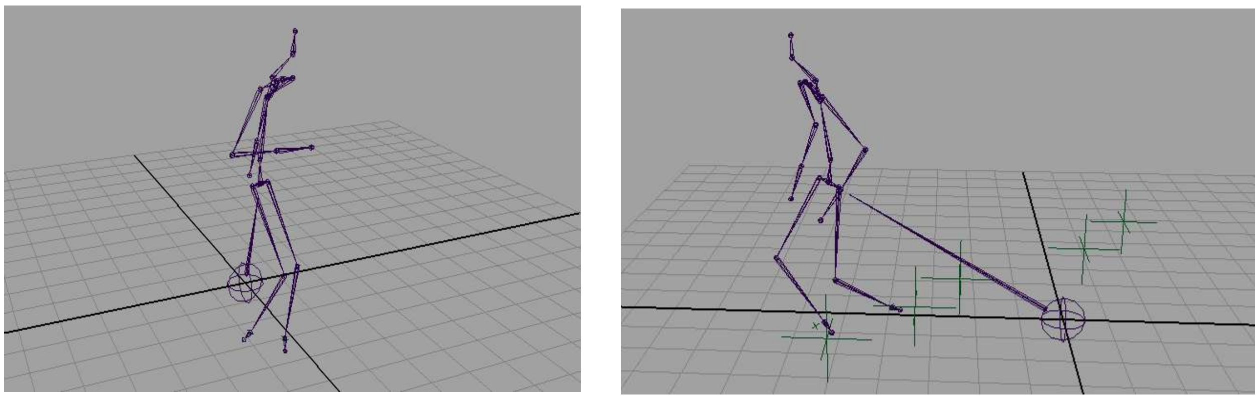


Figure 13:

A linear skip sneak transition (left) with floating feet and poor motion. A skip sneak transition using registration curves (right) with feet planted and realistic motion maintained

6.3.2 Blending more static motions

The ability to create different motion through interpolating raw input motions is another use of registration curves. To test this a large swinging boxing motion was blended with a small sharp boxing motion. Different weights were applied to this to create new types of hitting actions.

The interpolated weights were 50:50 and 20:80 respectively. The resulting swings were good, yet despite both input motions appearing to have reasonably static foot plants, the blended feet slid and waved.

This issue was solved using the constraint techniques explained earlier, but highlighted how subtle differences in seemingly similar motions can cause poor results.

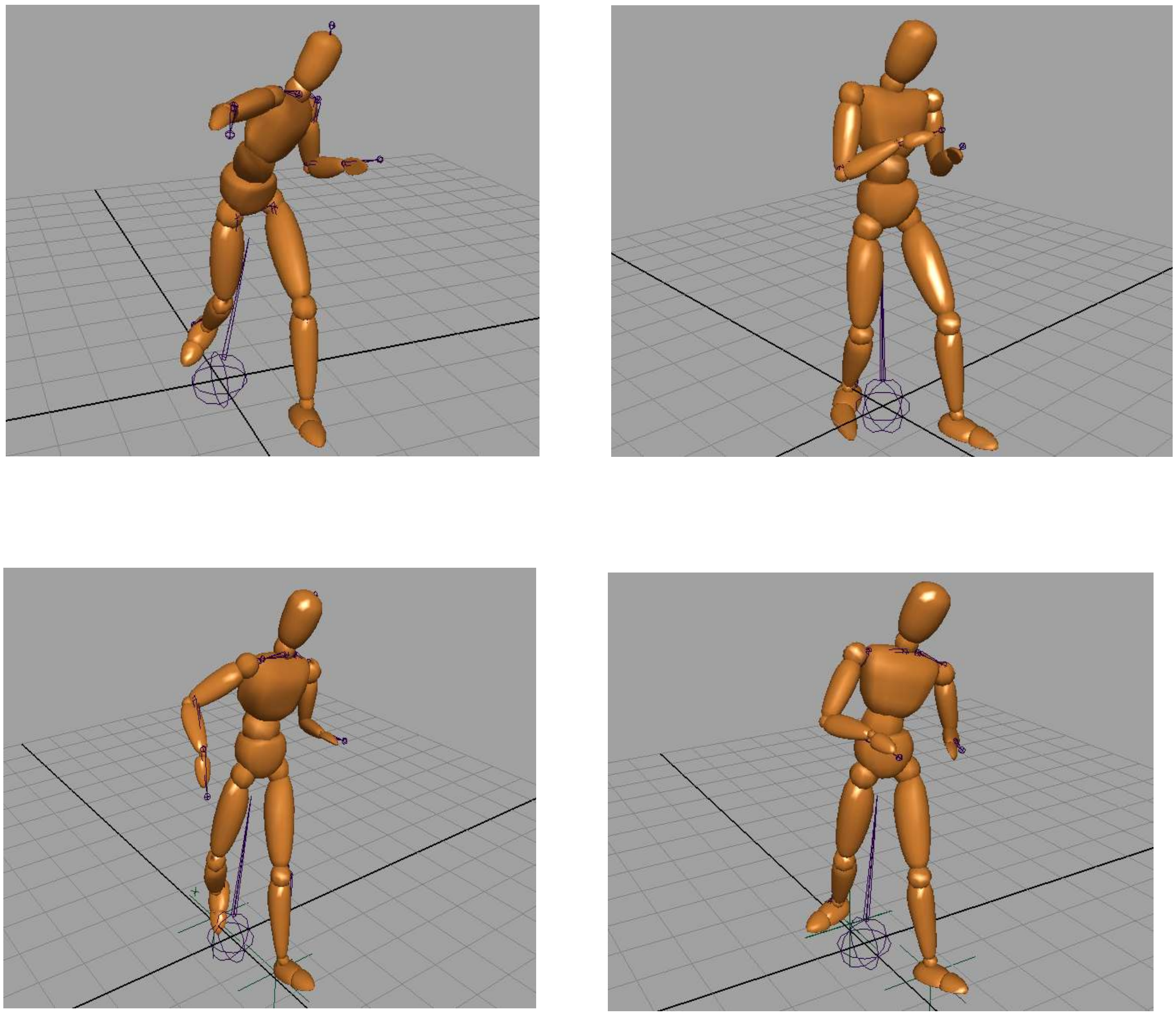


Figure 14:

The top left image is taken for the original large swinging motion, the top right is the far smaller jabbing motion.
Bottom left is a 50:50 interpolation using registration curves. Bottom right is a 20:80 interpolation using registration curves.

6.3.3 Binding a mesh to the blended rig

To complete the testing of the blended sequences, a mesh was bound to the animated skeleton.

Two more issues became evident once this process was complete. The first issue was the resulting animated mesh was self -intersecting on certain blended motions. For example, on the transition between the two walking paths, during the turn, the right foot goes through the left foot, occurring in both the linear and time warped versions. This is due to registration curves not taking balance into account. If balance was integrated then the feet would position themselves to provide support for the body. Therefore avoiding self-collisions.

The second issue, is due to constraints being forced on the skeleton at certain frames. Some of the limb nodes during these periods are slightly deformed in order to satisfy the constraints. This has a knock-on effect of deforming the mesh attached to the ankle joints at these frames. As a result, there needs to be careful manipulation of the influence of each ankle joint on its surrounding mesh to minimize the effects of deformation.

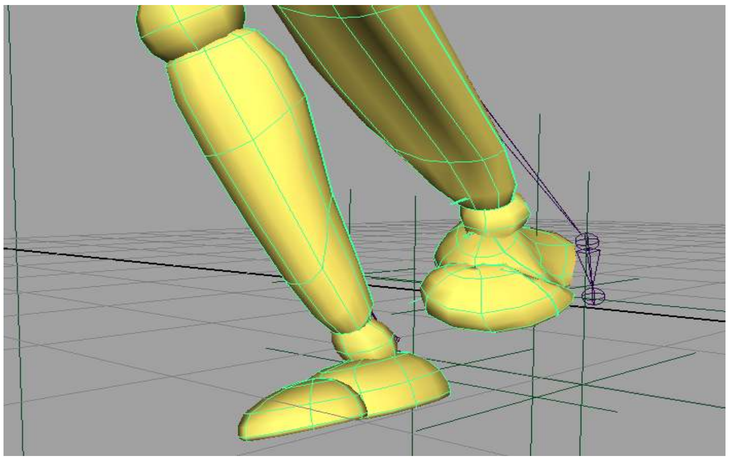


Figure 15:

One of the more extreme examples of deformation, caused when a limb node stretches to maintain a constraint

7. Conclusion

Automatic motion capture blending is a logical progression in enhancing entertainment technologies. It could provide major cost benefits to the industry. The research to date has developed varying techniques, however several limitations exist, in particular with the manual input required for blending.

Registration curves appear to adopt major elements required to implement automatic motion capture blending, and this thesis studied their relationship. The results of this study demonstrate that the implementation of registration curves won' t offer a full automatic blending solution. Although integrating several other techniques, for example foot skate cleanup or constraint based motion adaptation, provides a far more comprehensive solution.

As a result, the use of the term automatic is open to question, and would not gain universal acceptance. As this research thesis demonstrates the need for manual editing of the resulting motion in order to achieve a comprehensive visually acceptable blend necessary for the discerning human eye.

8. Future Work

There still remains a lot of scope for research and novel approaches towards creating an improved resolution to motion capture blending. To improve on the work carried out on this project, a method to create automatic constraint reaching using inverse kinematics would show the most significant improvement.

Further work, based on longer sequences of input data should also create a more stable blend. More motions can be created by extending the program to handled more than two inputs simultaneously. Also, taking into account more physical constraints such as balance will add an extra dimension to the resulting motion.

Other work being carried out at present (Hsu *et al.* 2005) is looking at real time solutions to motion blending, this is an area registration curves could be optimized for.

Finally, this application structure could be implemented to run within an application API, allowing full blending within 3D programs such as Maya or more specific applications such as MotionBuilder.

Acknowledgments

This author would like to thank Andy Cousins of Access Mocap Bournemouth University. Also Zhidong Xiao, Jon Macey, and John Vince of Bournemouth University, and Mark Doherty for their help in producing this masters thesis.

References

[1] Barbič J, Safonova A, Pan J, Faloutsos C, Hodgins J, Pollard N, 2004 , *Segmenting Motion Capture Data into Distinct Behaviors*, Proceedings of the 2004 conference on Graphics interface, 185 – 194, Ontario, Canada

[2] Bruderlin A, Williams L, 1995, Motion Signal Processing, Proceedings of the 22nd annual conference on Computer Graphics and interactive techniques, p 97 – 104

[3] de Jong R , van der Spek H, 2003, Voice Authentication, Thesis

[4] Ellis D, 2005, Dynamic Time Warp (DTW) in Matlab, Columbia University, available from http://www.ee.columbia.edu/~dpwe/resources/matlab/dtw/ [Accessed 5/7/2005]

[5] FBX SDK Overview, March 2005, Alias Quebec, Retrieved 20/04/05 from www.alias.com/eng/products-services/fbx/file/FBXSDKOverview.pdf

[6] Gleicher M, Litwinowicz P, 1996, Constraint-Based Motion Adaptataion, ATG Graphics Group, Apple Computers

[7] Gregersen T, Harsfort H, Hausmann L, Korsgaard P, Nielsen M, Stepien R, Thomas C, 2000, Automated Lip Synchronization of Animated Characters, Thesis, Aalborg University

[8] Hsu E, Pulli K, Popovic J, 2005, Style Translation for Human Motion , Proceedings of ACM SIGGRAPH 2005, p 1082 – 1089, San Diego

[9]Kovar L, 2003, *Registration Curves,*   Retrieved 23/04/05 from *http://www.cs.wisc.edu/graphics/Gallery/Kovar/RegistrationCurves/*

[10] Kovar L , Gleicher M, 2003, *Flexible automatic motion blending with registration curves*, Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer animation, July 26-27, 2003, San Diego, California

[11] Kovar L, Gleicher M Pighin F, 2002 , *Motion Graphs,* Proceedings of the 29th annual conference on Computer Graphics and interactive techniques, p 473 – 482, San Antonio, Texas

[12] Kovar L,Schreiner J, Gleicher M, 2002 , *Foot skate cleanup for motion capture editing, Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, July 21-22, 2002, San Antonio, Texas*

[13] Mandel M, 2003, *Motion Capture and Physical Simulation for Character Animation,* Retrieved 8/05/05 from http://www.city-net.com/~amandel/portfolio/masters.html

[14] Ménardais S, Kulpa R, Multon F, Arnaldi B, 2004 , *Synchronization for dynamic blending of motions*, Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation, Grenoble, France

[15] Ni K. 2004, *Motion Blending and Warping*, Retrieved 1/05/05 from *http://www.cc.gatech.edu/grads/n/nikai/research/7001_2_report.pdf*

[16] Parent R, 2002, *Computer Animation Algorithms and Techniques,* Academic Press, San Diego, USA

[17] Park S, Shin H, Shin S, 2002, On-line Locomotion Generation based on Motion Blending, Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, July 21-22, 2002, San Antonio, Texas

[18] Rose C, Cohen M, Bodenheiner B, 1998, Verbs and Adverbs: Multidimensional Motion Interpolation , IEEE Compuetr Graphics and Applications, p32 – 40, IEEE Computer Society Press, California USA

[19] Sederberg T, Greenwood E, 1992, A Phsically Based Approach to 2-D Shape Blending, Proceedings of the 19th annual conference on Computer Graphics and interactive techniques, p 25 – 34

[20] Troy, 2004, Film. Directed by Wolfgang Peterson. USA: Warner Bros.

[21] Witkin A , Popovic Z, *Motion warping*, 1995, Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, p.105-108, September 1995

[22] Wrigley S, 1999, *Speech Recognition by Dynamic Time Warping* [Accessed 20/7/05] http://www.dcs.shef.ac.uk/~stu/com326/sym.html

[23] Zordan V , Hodgins J, 2002 , *Motion capture-driven simulations that hit and react*, Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, July 21-22, 2002, San Antonio, Texas

Appendix A

Appendix A:Tool Overview / Operational Instructions

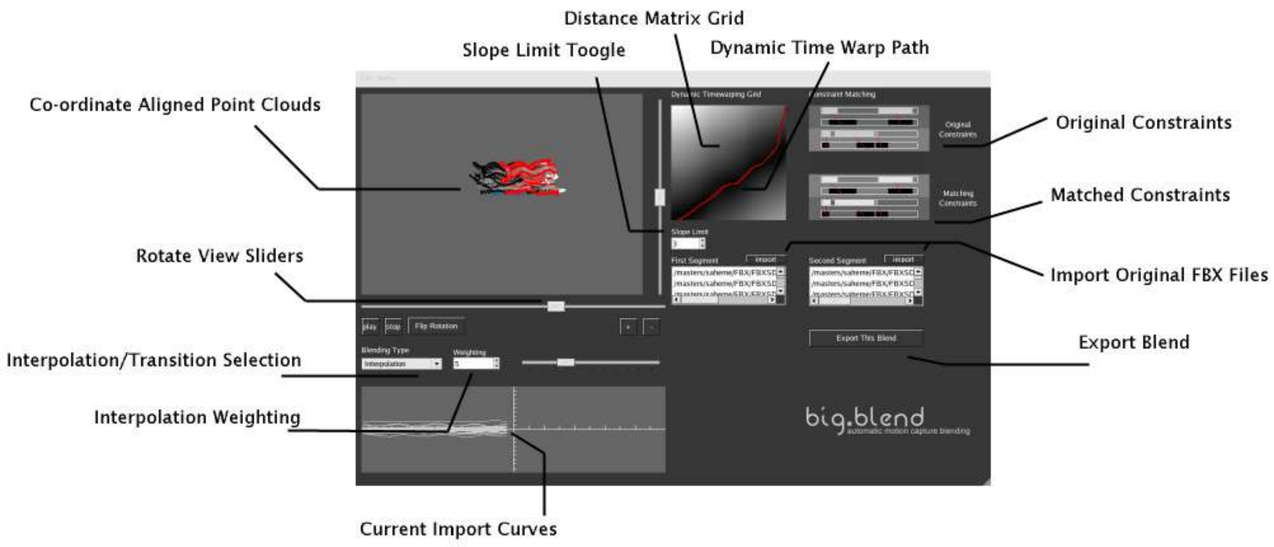


Figure 16:

Tool overview of ' big.blend' , the automatic motion capture blending tool implementing registration curves

Operational Instructions

To run the application load the file bigblend from the code folder.

Once the application is open segments of motion capture can be imported from the motion capture library.

The user has the choice of determining if the blend is to be a transition between the two segments or an interpolation of the two. If interpolation is chosen the weight of the interpolation can be amended via the drop down box. It is set at a default value of 5, but can be adjusted between 0 - 10

The slope limit of the Dynamic Time Warp path can also be adjusted via the toggle box underneath the distance matrix (dynamic time warp grid).

To export the blend press the *'Export this blend'* button

In your shell window (which is at the same directory) type the following

```
perl editBlendFile.pl
```

This will amend a bug with motion builder, which attaches unwanted artifacts to the file rig names, and will transfer the new file to the export folder.

Within this folder call *make*, and run the program *ExportBones nameOfFile*
For example:
ExportBones myNewBlend
Which will create the file myNewBlend.fbx

It should be noted that to make this file, the FBX SDK must be installed on the users System

This can then be imported into an application such as Maya or XSI.