Applications of Gait Analysis Data Compression for 3D Character Animation

Robert Lechte

6 November, 2008

Department of Computer Science & Software Engineering

University of Canterbury, Christchurch, New Zealand

Supervisor: Dr R. Mukundan

Abstract

We investigate a streamlined method for compression, approximation and fast interpolation of gait analysis data using Catmull-Rom Splines. We are interested not only in raw compression, but also extracting the most useful data from an animation for subsequent manipulation. Our method allows compression approaching 85 percent while the resulting animation remains indistinguishable by humans from the original animation, resulting in significant memory savings, while the untransformed compressed animation has possible usefulness in gait retargeting.

Contents

1 Background and Related Work

1.1 Three-dimensional Character Animation

The vastly reduced cost and increased power of graphics hardware has lead to a proliferation of animated media. Consumer grades graphics cards now feature real-time graphics pipelines allowing the rendering of increasingly detailed three-dimensional scenes, allowing the creation of many three-dimensional video games featuring realistic characters. Corresponding advances in non-real-time graphics rendering have allowed the creation of movies with incredible detail to the extent that computer-generated animation can be convincingly combined with photographic imagery to produce photorealistic films with computer-rendered characters. Jurassic Park is often considered the pioneer of such advancement, and prominent recent examples include Gollum in the Lord of the Rings, and the eponymous characters in Transformers.

It is instructive to note, however, that the computer-generated characters in the aforementioned examples, and in fully animated motion pictures such as those produced by Pixar, invariably feature non-human characters. Attempts to produce computer-generated photorealistic human motion have met with little success.

This is due to humans having a far great sensitivity towards the motion, expression and body language of other humans, as opposed to other characters such as animals, monsters and robots.

This is because physical human motion and expression plays an important part in communication and more broadly our wellbeing and success as a species.

It has been suggested[13] that our ability to detect subtle changes in gait serves an evolutionary purpose. Their study determined that males generally found the walking motion of women in one particular stage of the menstrual cycle more attractive than in another stage. This suggests that the ability to distinguish subtle changes in gait is a contributor to evolutionary reproductive success.

1.2 Human Motion Perception

Reitsma and Pollard[14] assessed the ability of human perception to detect errors in jumping motion, assessing error by magnitude, variety (horizontal or vertical) and acceleration (negative or positive). They found that errors in horizontal velocity were more readily detectable by human vision than vertical velocity, and added acceleration was easier to detect than deceleration.

1.3 Catmull-Rom Splines

Splines are piecewise-defined polynomial curves. They are typically utilized for interpolation and smoothing. There are many different types of splines, each utilized for a variety of purposes. Cubic Hermite Splines are a type of spline utilizing two control points and two control tangents to derive each piecewise

parametric function. The function derived from these control points consists of four cubic polynomials in Hermite form. If the control points and tangents corresponding to each piecewise parametric curve are chosen sensibly, then these curves may be combined to form an aggregate curve that passes through any arbitrary set of points and remains smooth (that is, continuously differentiable) throughout the path of the curve.

1.3.1 Derivation

To derive the form of a Catmull-Rom spline[4], we begin with the aim of devising a parametric curve whose path crosses one point at t = 0, and another point at t = 1, and whose slope has the desired values when crossing through these points. If we take this curve to be a cubic polynomial, then the resulting parametric equation must be of the form

$$P(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

(If we define $P(0)$ as the point the curve passes though at $t = 0$, and $P'(0)$ as the slope of the curve at that point, then it follows that:

$$P(0) = a_0,\ P'(0) = a_1$$

Likewise with $P(1)$ and $P'(1)$ as the point and slope respectively at $t = 1$:

$$P(1) = a_0 + a_1 + a_2 + a_3,\ P'(1) = a_0 + 2a_1 + 3a_3$$

If we solve simultaneously for a_0, a_1, a_2, and a_3, substitute these back into equation 1, and arrange in terms of P(0), P'(0), P(1), and P'(1), we are left with the following:

$$P(t) = (2t^3 - 3t^2 + 1)P(0) + (-2t^3 + 3t^2)P(1) + (t^3 - 2t^2 + t)P'(0) + (t^3 - t^2)P'(1)$$

Catmull-Rom splines are calculated from this, with the additional proviso that the required slopes, P'(0) and P'(1), are calculated from the points before and after the point in question, as follows:

$$\frac{P_{i+1} - P_{i-1}}{2}, \frac{P_{i+2} - P_i}{2}$$

This definition means that for any given sequence of points, the points slope of the curve entering and exiting a point is the same, making such a curve smoothly continuous. Expressed in matrix terms, we can calculate P(t) from the following:

$$P(t) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & 1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} P_i \\ P_{i+1} \\ \frac{P_{i+1} - P_{i-1}}{2} \\ \frac{P_{i+2} - P_i}{2} \end{bmatrix}$$

Which can be simplified as:

$$P(t) = \frac{1}{2} \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix}$$

1.4 Animation Methods

1.4.1 Keyframes

Existing approaches to character animation can be broadly classified into two contrasting approaches. The more traditional of these is keyframe animation. This entails an animator creating the important frames of an animation and then producing the "in-between" frames necessary to produce a smooth animation. Pioneering animators illustrated key frames by hand, delegating the arduous illustration of the required illustration of the inbetween frames (often called "tweening") to subordinate illustrators.

While computer-based animation now usually involves three-dimensional digitally-rendered models rather than two-dimensional illustrations, and utlizes automated "tweening" algorithms,the essence of the process remains very similar: A digital animator animating a character model must specify key frames (or poses) for the model, as a basis for the interpolation which generates the final animation. While computing power has greatly streamlined this process, the inherent realities of the process have not changed.

Keyframing allows virtually unlimited flexibility: Anything that can be drawn or modeled can consequently be animated. However, the drawbacks of keyframing are the high requirements for talent and time required. In particular, the creation of highly life-like animation is extremely time-consuming, due not only to the exacting attention to detail required, but to the need for a higher raw quantity of keyframes to be used in order to illustrate more of the subtleties of the motion.

1.4.2 Motion Capture

Motion capture is a comparatively new technique for generating animations, having only become practical after the advent of digital photography. A variety of techniques exist, but typically markers are attached to the actor. The position of these markers is then calculated from video from multiple camera angles (using computer vision techniques and manual correction if necessary) or other recording from other sensors. Other techniques include analysis of markerless motion, and the use of inertial measurement systems. The advantages and disadvantages of this system are immediately apparent. These techniques allow for the recording and generation of extremely realistic and detailed animation.

The realism of the resulting animation becomes dependent on the quality of the processing of the motion capture data rather than more fickle human qualities such as artistic talent. However motion capture remains a field in

comparative infancy, and generating motion capture data requires a live actor capable of acting out the motion required. For many motions (such as elite gymnastics) a suitable actor may be difficult to find. Motion capture data can be edited to some extent, often by extracting suitable frames from the original motion for use as keyframes (see Keyframe Extraction, below). However, extensive editing invariably degrades realism as intricacies captured in the original recording are modified and lost.

Motion capture also has use in diagnosing and monitoring the movement of those with physical injuries. However the utility and availability of such facilities is limited by the complexity and expense of the motion capture process, and also by the need for subjects to wear specialized markers or suits: this can be particularly inconvenient or even impossible for those with physical disabilities. This is discussed further in section 1.7.1, below.

1.5 Forward Kinematics

The complexity of animation of detailed objects or scenes with multiple moving parts is typically managed by dividing the subject into a tree of components, with the position of each item in the tree being relative to the position of its tree parent. For instance, a car can be drawn by first drawing the body of the car, and then drawing the wheels in a position relative to the car. Similarly, human motion is stored and modeled as a tree hierarchy, with the center of the body at the base of the tree, with branches for each bodily extremity such as limbs and head.[1]

1.5.1 Skeletons as Joint Hierarchies

Human motion could be stored by recording the position of many points on the body surface. However, this approach generates an excess of data, and makes reuse and adjustment of the motion extremely cumbersome. Invariably then, motion data is stored using a skeletal hierarchy, which is then "skinned" or "meshed" to provide the correct look. This allows the motion and aesthetics to be manipulated largely independently.

1.5.2 Forward and Inverse Kinematics

The motion of the skeleton can be generated either of two broad methods. Forward kinematics defines an angular position for every joint (and a translational position for the root joint) for each time instance. The figure is then drawn by simply traversing the tree, drawing each limb as it is traversed. Inverse kinematics is also known as goal-directed motion. Accordingly, the angles of each joint are not stored, but rather are derived from "goals", or points towards which parts of the figure must move.

A number of hybrid approaches have been developed attempting to combine the advantages of each method. Such methods[18] typically provide movement

constraints from the forward kinematic data within which inverse kinematic algorithms can operate.

1.6 Motion Capture Data Storage

The usual output of the raw motion capture is simply marker positions. This data is unsuitable for the modelling and skinning process, so more usable motion capture formats provide data in terms of a joint hierarchy (skeleton), along with joint angle information for each frame for each instance. For example the popular ".bvh file" format specifies motion in terms of a skeleton, along with multiple lists of angles in "channels"; each channel specifies the angle changes for each axis of a movable joint in the skeleton. Three channels also exist to specify the overall position of the skeleton in terms of translations on each axis.

1.6.1 Mathematical Transformations

To render the character in a 3D graphics library such as OpenGL, we translate to the correct position using the positional channels. We then begin at the root of the skeleton, and traverse the skeletal tree recursively, applying rotations as specified by the channels. Rotations are specified independently for each rotation axis, and are applied in the following order:

$$p_{rotated} = p\,[Y_{rotation}]\,[X_{rotation}]\,[Z_{rotation}]$$

1.7 Gait Analysis

Gait analysis is the study of animal movement, particularly humans. Quantifiable gait analysis is used in physiotherapy and rehabilitation monitoring. It is also used for sports performance evaluation. The advent of motion capture has meant renewed interest in the field as this new source of gait analysis data has the potential for usefulness in several fields.

1.7.1 Medical Applications

One hindrance to the usefulness of motion capture is the need for markers. These markers can hinder natural movement, and the precision configuration involved means that marker-based systems cannot be used for real-time applications. To remedy this, markerless methods have been developed. The method developed by Saboune et al.[16] attempts to convert a two-dimensional camera image of an actor to a 3D hierarchical model of a generic human, by using particle filtering on a foreground image and best-fitting the foreground to various orientations of the figure.

Noble and White[11] implemented a pilot system which potentially aids the clinical diagnosis of joint movement abnormalities. It uses the graph of joint movement on a particular axis, plotted against regular movement, to detect differences and discrepancies. For instance, any movement greater than +or- 2 standard deviations from typical movement is highlighted. Slope, peaks, and

total range are also used to display pertinent information to the health professional.

1.7.2 Redundant Data

As per-axis gait analysis data is merely a waveform, usual waveform compression techniques can be applied, particularly as most human motion (excluding environmental interactions) exhibits an inherent smoothness. This makes the data particularly amenable to compression.

1.7.3 Motion Capture Data Compression

Compression of motion capture data is useful for saving memory space while rendering, and for minimizing the size of motion capture databases, particularly for organisations with large collections of such data such as animation studios.

In terms of compression ratio, the technique developed by Arikan[2] is excellent. Motion capture files can be compressed to less than 5% of their original size: Arikan's test motion database was compressed from 180 megabytes to 5.5 megabytes. This uses a technique whereby joint angles are transformed into positional markers, making them much more amenable to compression using Bezier curves. Environmental interactions are compressed differently, as the positional information must be more precise. The technique also examines similarities between clips. In this way, clips are compressed as a group. This reduces information redundancy, as many clips contain very similar motions, but also reduces flexibility, as many clips must be compressed and stored together. The compression used is also complex, resulting in computation time during decompression.

A different approach[17] applies compression by identifying parts of the skeleton which remain consistent over time.

1.7.4 Environmental Interactions

The accuracy of motion data is most important at the points in the motion where parts of the character interact with their environment. This can include impacts from other moving objects and the character picking up and using objects, but often occurs during footsteps, which are almost universally present in character motion. When motion capture data is modified, imperfections are most often noticable as "footskate", where the feet relative to the ground during contact, instead of being correctly fixed on the surface.

Footskate occurs because forward kinematic motion is not goal-directed: movements are defined only with respect to other parts of the hierarchical skeleton, not the ground.

Footskate must be fixed either by manual adjustment, or using a more automated cleanup technique[8].

1.8 Keyframe Extraction

This is the process of choosing the most "representative" poses of a motion sequence from continuous frames contained in motion capture.

Keyframe extraction has origins in traditional video manipulation. The ability to browse a large set of efficiently video and examine the contents of a clip without watching the full video is important to those working with large collections of video, and increasingly relevant for ordinary consumers with large collections of video media.

Similarly, for those working with large motion capture databases, the ability to quickly search for and review suitable motion capture sequences quickly means much greater efficiency and productivity.

Motion capture keyframe extraction is also useful for provision of suitable keyframes for use in modification of the motion capture data[12]. While this is at a cost of at least some degree of realism, techniques exist for minimizing obvious unrealistic movement such as footskate, as discussed above.

1.8.1 Existing Approaches

The simplest method of keyframe extraction is by regular sampling. This usually provides acceptable results, but is less than optimal as often the important parts of a motion are contained in certain concentrated blocks in the motion. For instance, if the motion capture records the actor waiting for a period of time before beginning acting out a jump or other brief motion, consequent keyframe extraction will feature disproportionate keyframes of the character merely standing still. For better results, less naïve methods are required.

Many techniques exist for intelligent keyframe extraction of video, also known as static video summarization. An early effort used a variety of methods to measure entropy changes between frames, thus dividing a video clip roughly into scenes[5, 6, 9]. Excess keyframes chosen by the entropy metric were then filtered out on the basis of colour uniformity and repetition. A more recent method uses a k-medoid fuzzy clustering algorithm on the luminance channel of video to determine the most central frame for each cluster[7]. Keyframe extraction for motion capture data can take advantage of the richer semantic detail of the motion format, and use this to choose the most significant stages of the motion[3, 12].

1.9 Motion Retargeting

Motion capture records one specific motion, with one specific actor. Motion retargeting techniques attempt to allow recorded motion to be transformed to other characters with different shapes and proportions, thus increasing reusability. One recent technique[10] uses grammatical evolution through a number of intermediate "hybrid models" to translate motion from gradually from one character shape to another.

2 Implementation

2.1 Aim

Our general intention is to determine the most semantically rich data points in forward kinematic joint angle data.

This allows for the most optimal reduction in data set size if we eliminate certain points and subsequently interpolate them. Also, we can simplify manipulations of such data if we have a smaller set of important points with which to represent the characteristics of a motion, rather than a larger volume of largely redundant data.

Existing methods of motion capture data compression rely on extensive transformations of data. This results in very aggressive compression, but means real-time usage of the compressed data is comparatively computationally expensive.

We instead seek to eliminate points which can be considered irrelevant and unnecessary to the overall essence of the motion. The resulting data should then allow computationally simple interpolation.

Our method merely eliminates unnecessary points. This results in a set of remaining points which can be interpolated to form a close approximation of the original motion.

We examined four point elimination techniques which rely on approximations, and two methods which eliminate points by attempting to minimize the value of an error metric. We also tested three interpolation techniques. Catmull-Rom interpolation uses Catmull-Rom splines to interpolate the required values. This results in mathematical and perceptual smoothness of movements, but is computationally difficult. Linear interpolation is the most computationally simple, but the interpolated motion will lack smoothness for long sequences of interpolated data points. Finally, we test an approximation of the Catmull-Rom algorithm which provides some level of smoothness while remaining relatively computationally straightforward.

The key advantage of an interpolation-based approach is flexibility in compression: Because the compression is carried out point by point, a motion capture animation can be compressed to any given requirements: an acceptable quality of animation, or a particular percentage of compression. The resulting data representation should be suitable for storage of any forward-kinematics-based data, since full uncompressed motion capture files can be stored in the same format as compressed data, or data stored as keyframes (in this case, only the angles from each keyframe would be stored.

The locations of missing eliminated data points would be marked in such a storage file and then compressed with run-length encoding.

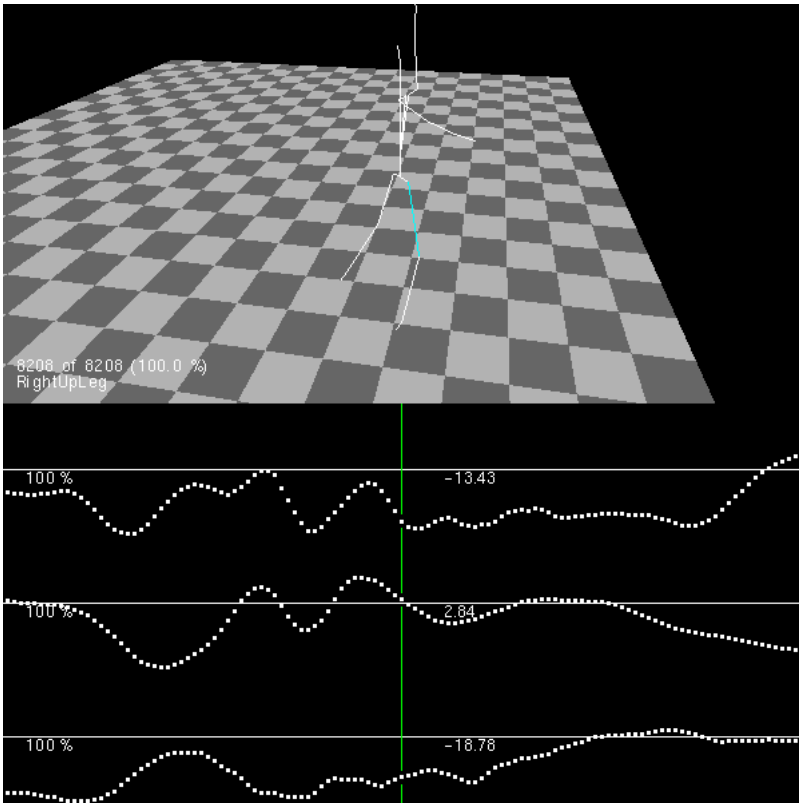

Figure 1: A typical screenshot of the software we used to visualize and examine the gait analysis data of motion sequences.

2.2 Interpolation Methods

2.2.1 Catmull-Rom Interpolation

Interpolation of time-dependendent angle data requires determining a value for fixed values of time. As a Catmull-Rom spline is determined by parametric equation, we must solve a cubic equation in order to determine the correct value for the parameter t.

2.2.2 Approximation Interpolation

In order to avoid the necessity of solving a cubic for interpolation, we can approximate the value of t. We have made this approximation as follows, assuming interpolation of a point p between $p_0$and p_1:

$$t = \frac{p_x - p_{0x}}{p_{1x} - p_{0x}}$$

This means the Catmull-Rom interpolation effectively disregards the position of the outer points along the time axis. This means the differentiable smoothness property no longer holds, but an approximately smooth curve is still produced.

2.2.3 Linear Interpolation

We used linear interpolation as a baseline, to determine the increase in interpolation accuracy provided by the above two techniques.

2.2.4 Time Performance

Our implementation of these interpolation techniques in python reported the following average times for a single interpolation. It is apparent that the approximation allows much faster interpolation than the strict Catmull-Rom technique.

2.3 Decimation Criteria

We evaluated several techniques for deciding between points to remove and points to retain, in order to retain the most important and informative data for a given compression percentage.

Note that for the purposes of consistency, we shall exclude the two starting and ending points in determination of compression percentage, as the values for these four points are not able to be interpolated, and are thus irremovable. For instance, an animation with 104 frames and 24 axes of rotation has a total of (104 - 4) * 24 = 2400 removable data points. Removing 1200 of these points would be considered 50% compression for the purposes of this document.

Several factors must be taken into account when choosing points to remove. To measure and quantify these factors, we have formulated the following definitions.

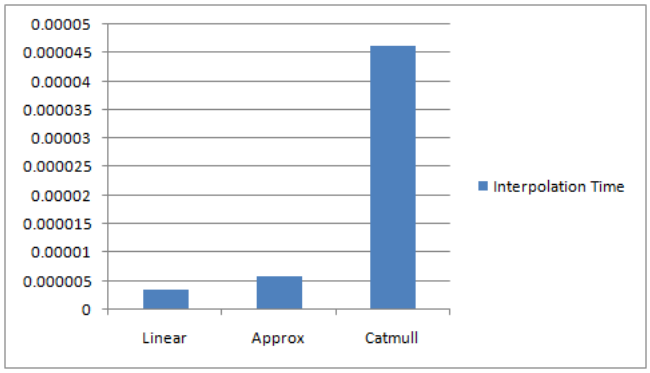

Figure 2: Comparative Time Performance of Interpolation Techniques for a single interpolation

- *Length Factor*: To calculate this, we first take the length of the joint in question added with the length of all child joints of that joint. This value might be called "recursive length". For instance, the "recursive length" for a shoulder joint adds the length of the shoulder, plus the length of the upper arm, lower arm, hand, and all fingers. The length factor then a proportional measure: the recursive length of a joint, divided by the total recursive length of all joints. Thus all length factors together add to one. The length factor can be thought of as a proportional importance of each joint.
- *Amplitude Factor*: The joint angles for each axis have a certain variation. For joints with large angular variation, approximations will result in greater error. Amplitude factor is thus determined from the range of each joint angle axis. This is also a proportional measure: each joint axis in the skeleton is assigned an amplitude factor defined as its range divided by the total range. The total range of the skeleton (one range for each of the three axes for each joint), and each joint axis in the skeleton is assigned an amplitude factor defined as its range divided by the total range.

2.4 Error Metrics

To evaluate the effectiveness of our various point removal algorithms, we formulated two metrics.

2.4.1 Cumulative Angular Error

To calculate the cumulative angular error of any animation, we examine all frames, and add together all the differences between interpolated and original angles, for any interpolated point. Each of these differences is multiplied by the Length Factor.

2.4.2 Total Endpoint Error

The total endpoint error examines every frame, and for every endpoint (end joint) on the articulated figure, determines the original position of each endpoint, and its approximated position. The sum of these differences is the total endpoint error.

2.5 Heuristic Point Removal

2.5.1 Periodic Sampling

Periodic sampling entails removal of all but every n-th data point, with n dependent on the level of compression desired.

To determine the points assigned to each joint axis, we use a combination of length factor and amplitude factor. For each axis of each joint, we multiply Length Factor by Amplitude Factor to give each axis of the skeleton a value. The number of points allocated to each axis is assigned in proportion to this combined value.

2.5.2 Curvature Change

This is a heavily approximating metric. We earlier noted that a point in line with surrounding points carries little information, and can be interpolated accurately, in contrast to points which are inconsistent with the surrounding path and curvature. Thus we can choose points to remove by calculating the resulting change in slope for all points affected for each prospective point to be removed. Specifically, we calculate the change in slope of the linear path entering and exiting the prospective point, as well as for the proceeding and succeeding point. We then interpolate between the slope changes of the proceeding and succeeding point, and assess each point based on the extent to which the actual curvature change of the the point matches the interpolated (expected) curvature change. A greater match means a greater candidate for removal. Here too we multiply the descrepency by the Length Factor to determine a final rating for each data point. Amplitude factor is not used here, as this is taken into account by the level of curvature change.

2.5.3 Maxima and Minima

This method involve retaining only points which are a local maximum or minimum. Unlike the other techniques, this does not in itself allow for a gradual level of compression.

2.5.4 Curve Approximation (Lowe's Algorithm)

With this heuristic method, points are assigned to the skeleton's rotation axes as with periodic sampling, however within each axis, points are distributed using Lowe's curve approximation algorithm[15], rather than being distributed periodically. Lowe's algorithm begins with a single linearly-interpolated line

between the start and end points of a curve. Each step of the algorithm adds a point, choosing the point to add based on the distance of that point from the linearly-interpolated line. Once a point is added, the linearly-interpolated line incorporates the new point, and distances are recalculated. The algorithm continues until the desired number of points have been added.

2.6 Results-based Point Removal

These methods eliminate points one at a time, based on which point will increase the error metric the least.

2.6.1 Angular Error Minimization

At each stage of the point elimination process, the change to the cumulative angular error which would result from that removal is evaluated, and points are ordered based on the resulting change. The point which least increases the error is chosen as the point to remove.

2.6.2 Endpoint Error Minimization

This works the same fashion as the above technique, except the points are evaluated based on their change to the Total Endpoint Error of the approximated figure.

2.7 Human Perception Test

In order to better determine the point beyond which point removal visibly degrades the quality of the resulting animation, we conducted a study of ten people.

Participants were presented with two nearly identical animations displayed concurrently on the screen. One of the two screen areas displayed the original animation, whilst the other displayed a version of the same animation, with a certain percentage of points removed.

The position of each animation on the screen was randomly chosen. Participants were asked to carefully examine the animation, and then "choose the fake". Participants were allowed to play and pause the animation, and also control the camera, such that the animation could be viewed from any angle.

A wireframe of the animation was used, as this makes the experiment more stringent: It is easier to spot changes in motion when angles of joints are directly visible, rather than hidden behind a "skin".

Participants were presented with 20 tests: comparing the original animation with compressed animations with qualities ranging from zero percent of the removable data points remaining, to 95% of the original removable points still existing, in five percent increments. Participants were presented with the test in random order.

The metric used for compression was the Total Endpoint Error. We chose the Walk animation as the candidate animation for this test as walking is a

motion of universally high familiarity, and thus test candidates are likely to be more discriminating when spotting subtle differences.

The nature of the test setup means that even if participants are unable to determine the less realistic motion, they may still guess correctly 50% of the time.

3 Results

3.1 Comparison of Point Removal Algorithms

3.1.1 Angular Error

Overall, point elimination affects animation quality very little until more than half the available points have been removed. This is due to several factors: The excess of points in unimportant joints, the accuracy of interpolation over small ranges, and the presence of many joints with little movement.

Subsequent removals have an increasingly strong impact on animation quality, with the error increasing logarithmically as the final points are eliminated.

We evaluated each point removal algorithm on a motion capture file portraying an actor in a regular walk. We discuss the results in order of quality. We consider Angular Error Minimization to be the benchmark as related to this metric, and other results are judged in relation to this. Accompanying graphs are shown in Figure 3.

- Maxima: While simply choosing local maxima and minima is very fast, results were poor. 23% of data points were local maxima and minima, and retaining these points results in as similar angular error as the retention of only 5 percent of points using the benchmark Minimum Angular Error Method.

- Curvature Change: This showed disappointing accuracy, exhibiting an error of more than double the benchmark, particularly as error begins to increase strongly at the 30% level and below.

- Regular Sampling: Regular sampling performs well, particularly given its comparative simplicity to other methods. Error rates are generally around double the benchmark level.

- Lowe's Algorithm: This performs very similarly to the regular sampling method. It performs slightly worse than regular sampling in the 12-34% range, while outperforming it to a very small extent at high compression levels. Overall, it offers no improvement over regular sampling with Catmull-Rom interpolation.

- Endpoint Error Minimization: This performs nearly as well as the benchmark. Both metrics show smooth, logarithmic increases, and even though the benchmark is tailored to the Angular Error Metric, this result shows Endpoint Error Minimization performing almost as strongly.

- Angular Error Minimization: This metric demonstrates smooth performance, allowing reduction to around 15% of the file size before quality begins to rapidly degenerate.

3.1.2 Endpoint Error

Evaluation of the performance of each algorithm using the Total Endpoint Error Metric gave similar results. The overall ranking and general performance of each technique remained the same. The benchmark metric again showed little increase until approximately 15% of points remained. Figure 4 shows the results for this metric.

3.1.3 Results with other Motions

Due to the nature of the metrics, a direct comparison of errors between different motions is not meaningful. So as to ensure that our evaluation results held more generally, we repeated the point elimination tests with two other sequences with very different animations. Results were proportionately similar, with each method degrading logarithmically and performing similarly in comparison to other metrics.

3.1.4 Lowe's Algorithm Results

Of note here is the relatively poor performance of Lowe's algorithm in subdividing a the path of each graph of rotation angles. We hypothesised that this was due to the fact that Lowe's algorithm is designed for linear interpolation, rather than other techniques such as Catmull-Rom. We thus ran tests using linear interpolation, to see how this affected the performance of Lowe's algorithm.

The resulting data showed that for linear interpolation, Lowe's algorithm does indeed slightly outperform regular sampling, as expected. This is shown in Figure 5.

3.2 Interpolation Comparison Results

The approximated method of interpolation showed very good performance, over a variety of animations. We evaluated the resulting error for the range of compression levels using each technique, and with each metric, and compared the resultant error.

We are most interested in the algorithm's performance in the range where 10-20% of data points remain. Before this, all three methods perform virtually identically, and after this, error rises rapidly for all three techniques. We first examined the performance with angular error. Throughout this range, the approximation either matched, slightly outperformed, or remained much closer in error rate to Catmull-Rom interpolation than Linear. It is evident that for those animations for which Catmull-Rom interpolation decreases error, the approximated version reduces the error nearly as effectively. For instance (as shown

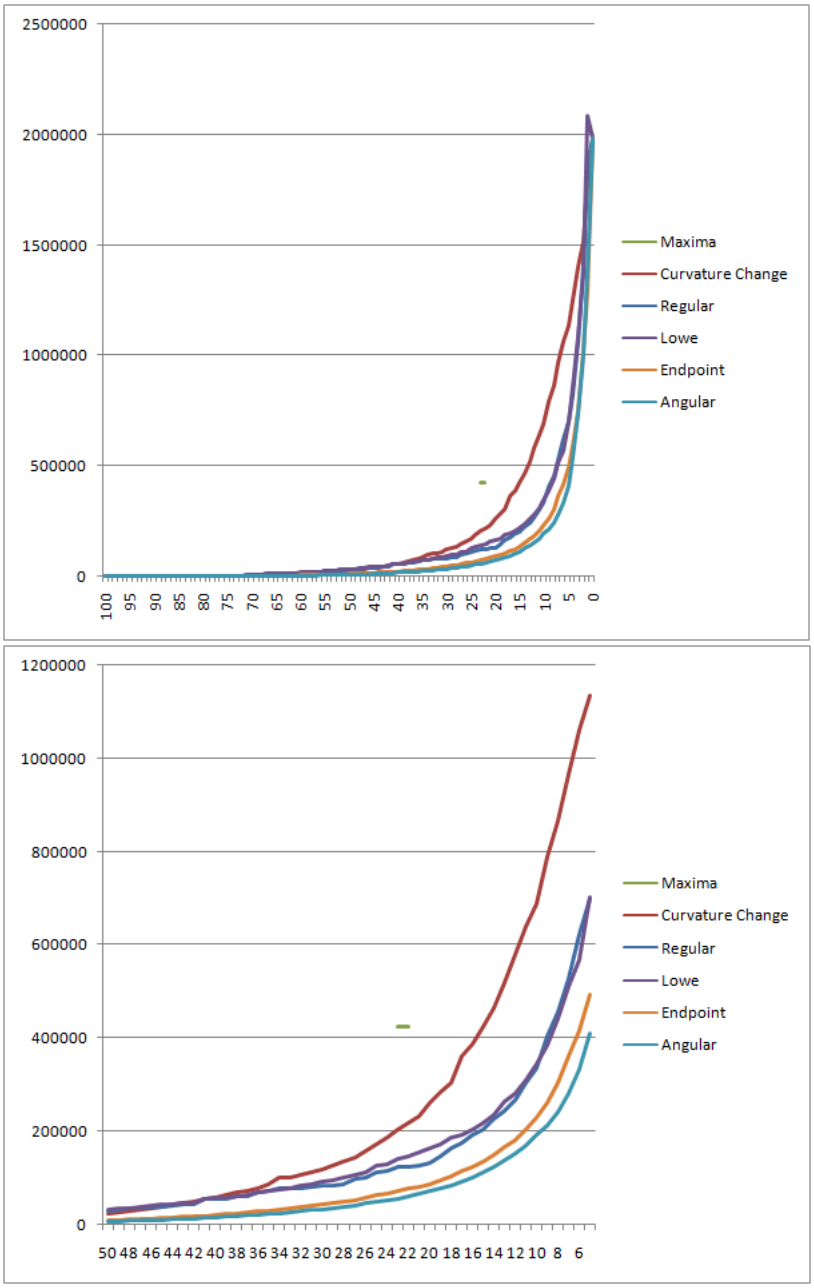

Figure 3: Cumulative Angular Error for various point removal algorithms (Catmull-Rom interpolation)

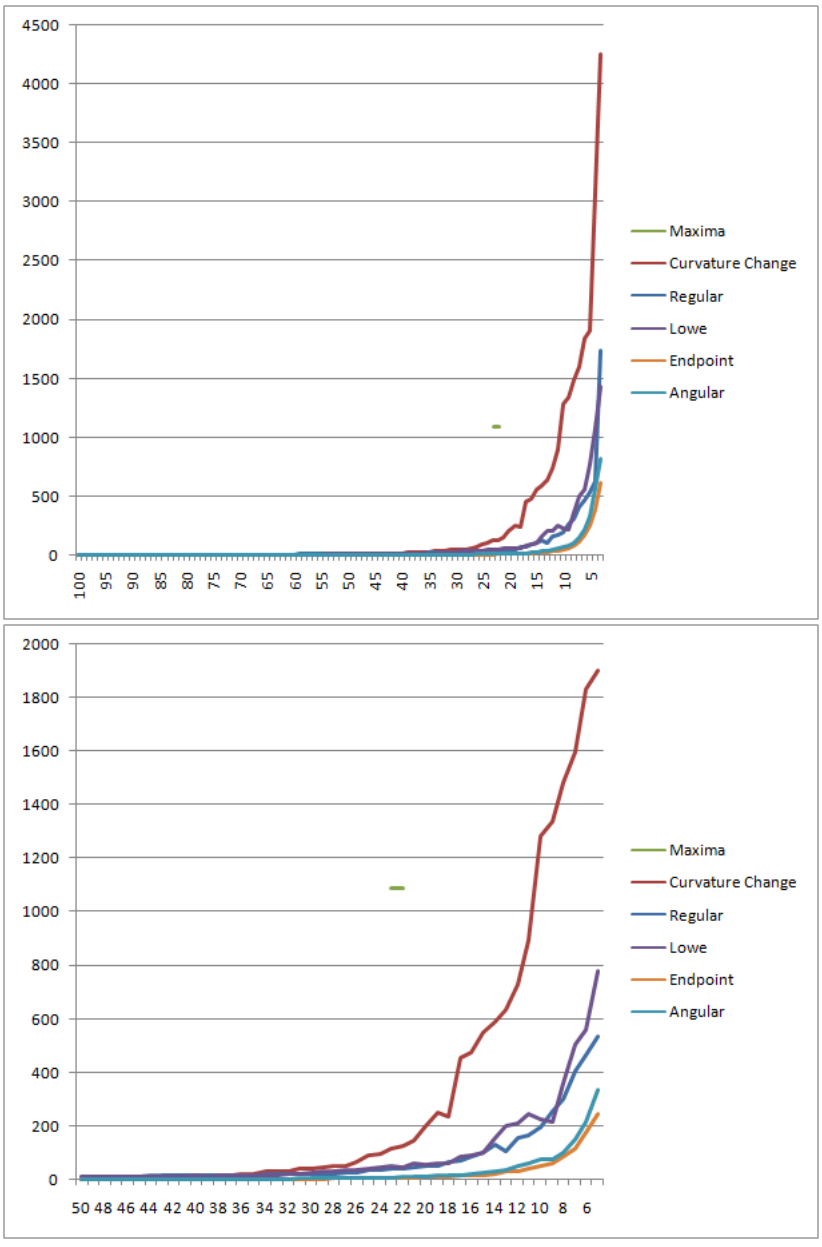

Figure 4: Total Endpoint Error for each algorithm (Catmull-Rom interpolation)

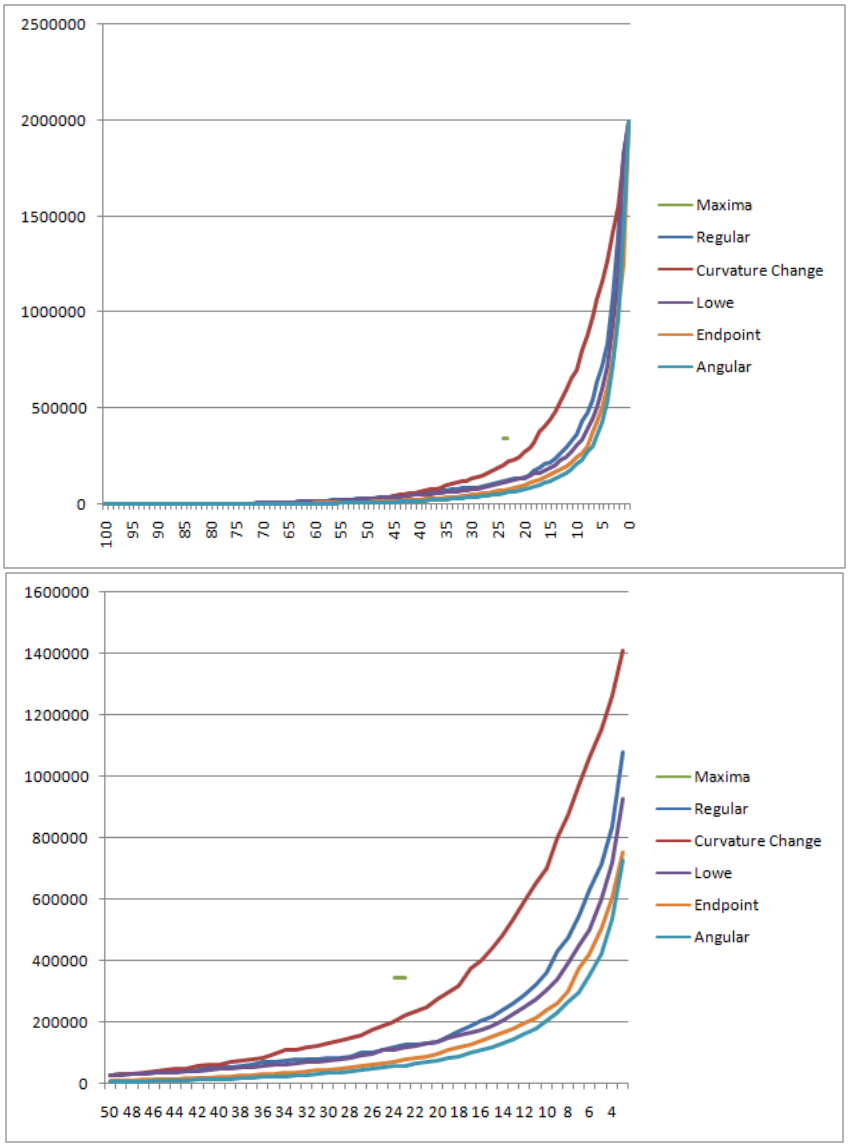

Figure 5: Total Angular Error for each algorithm (Linear Interpolation)

in Figure 6), with 15 percent of joint angle data remaining, linear interpolation resulted in a cumulative angular error of 609073. Using Catmull-Rom interpolation for compression and reconstruction reduces this error to 412153, while the approximated interpolation is almost as effective, reducing error to 453851.

Results with the Total Endpoint Error metric also showed the algorithm performing well (as seen in Figure 7), though this metric exhibits more noise than total angular error, making it harder to draw firm conclusions.

3.3 Human Perception Test Results

Results from this test are promising. We can see from the data (in Figure 8) that animation remains reasonably convincing at least until the the last 10-15 percent of points are removed. This is consistent with our error metric data, which suggested that error begins to increase rapidly when around 15% of the removable data remains.

4 Discussion

4.1 Error Metrics

Both of the error metrics used are reliable, and are consistent both with each other, and with human perception. Minimizing one error metric also mimimizes the other metric nearly optimally. Each demonstrates low levels of error until approximately 15-20% of points remain, and rises rapidly thereafter. This suggests that around 15% of the data points provide most of the data in an animation, with returns rapidly diminishing as further points are added, or with quality rapidly diminishing as further points are removed.

Total Angular Error is of greater practical use as a metric, as the changes in error level exhibited by this metric were smoother. The metric was also faster to calculate than Total Endpoint Error in our implementation.

4.2 Point Removal

Our results suggest that proportional allocation of data between joints and axes is of greater importance than allocation of points within those axes. For instance, Lowe's Algorithm offered little improvement over the Regular Sampling techniques, even though Lowe's Algorithm is a much more advanced approximation technique, and even when using linear interpolation, to which Lowe's Algorithm is tailored.

This also stems from the fact that Lowe's algorithm uses a linearly-interpolated line to choose points to add. When non-linear interpolation methods are used, it would be more appropriate to use a line between points interpolated with the same interpolation technique.

None of the approximated algorithms come close to the results gained when directly attempting to minimize error levels. While these direct methods are effective, their running time is slow (see below). This means more investigation

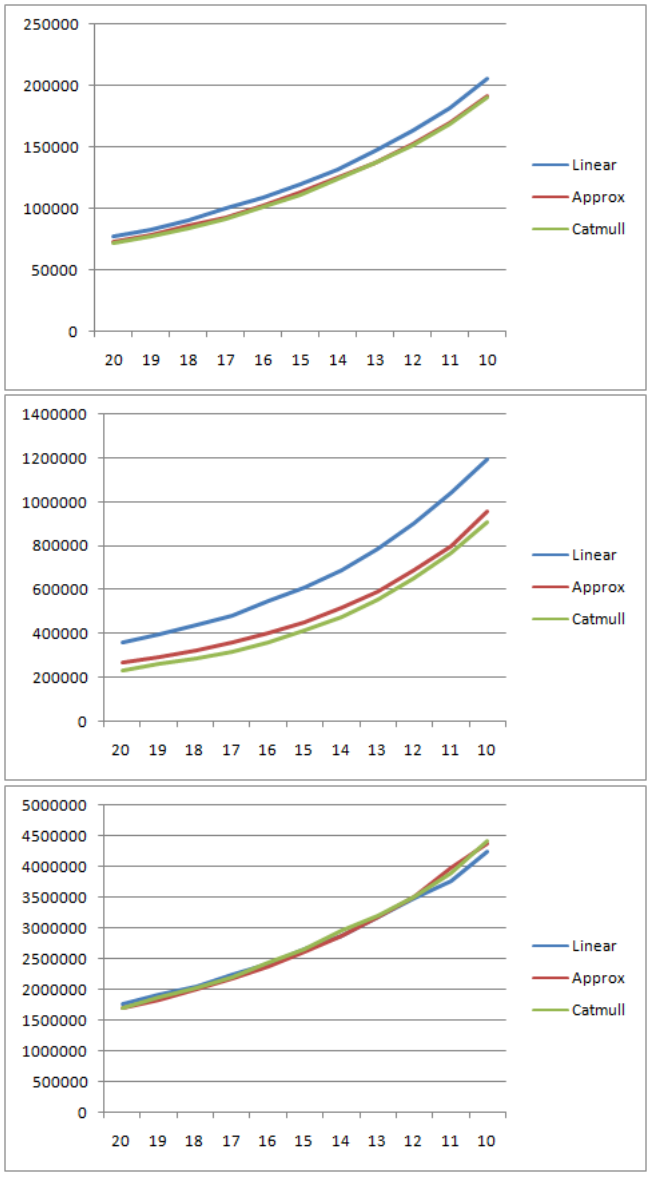

Figure 6: Angular error for each interpolation technique, showing error with between ten percent and 20% of points remaining. The animations tested were (from top to bottom): Walk, Slam Dunk, and Gunshot.

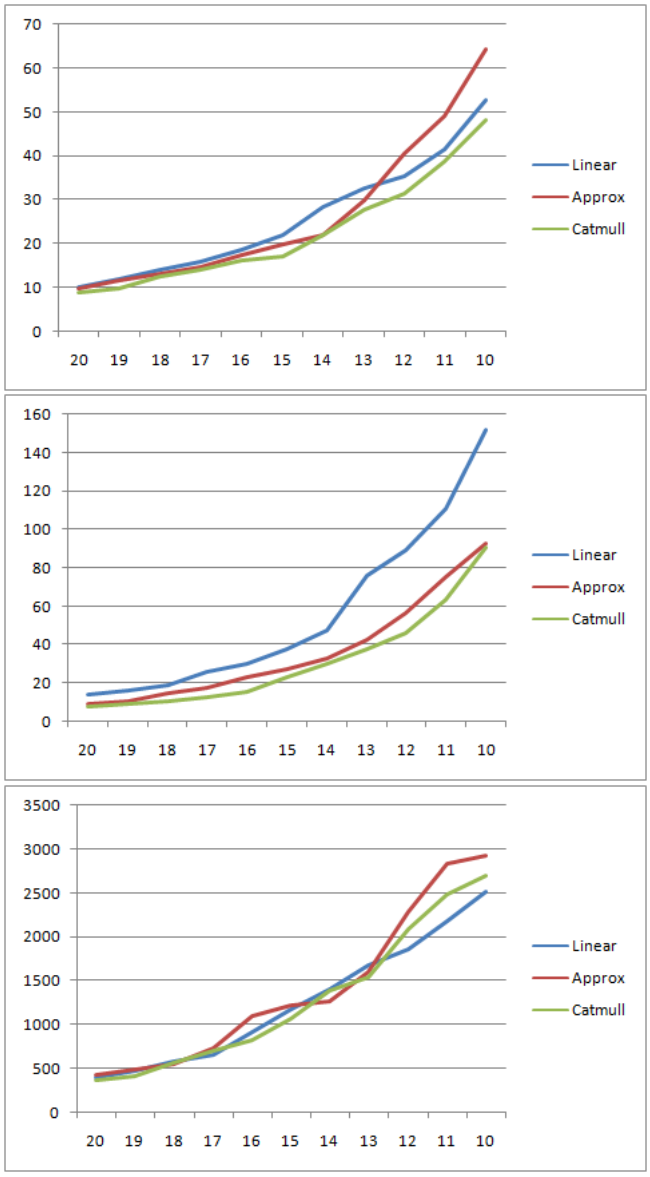

Figure 7: Total endpoint error for each interpolation technique, showing error with between ten percent and 20% of points remaining. The animations tested were (from top to bottom): Walk, Slam Dunk, and Gunshot.

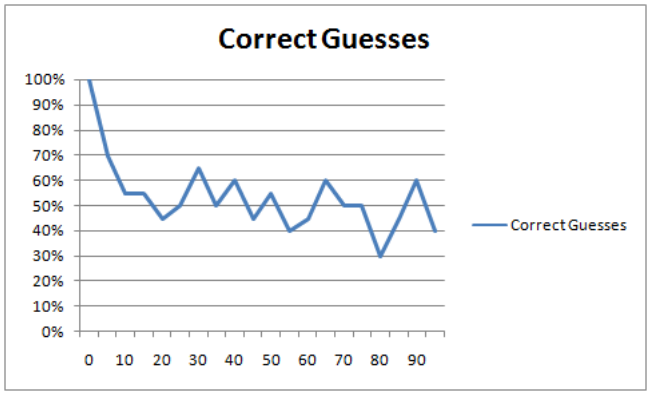

Figure 8: Percentage of guesses made correctly for various levels of compression of Walk animation

into good approximated metrics for determining comparative data importance would be of use.

4.3 Speed of Algorithms

In general, reduction in error requires additional running time for each of the point elimination algorithms we tested (at least in their unoptimized form). Choosing the maxima and minima only is by far the fastest methods, but error results were disappointing. This poor performance is at partly due to the fact that this technique does not place any additional importance on the more "mission-critical" joints and points higher in the skeleton hierarchy tree.

The error resulting from the curvature change algorithm was also excessive, especially given that it nevertheless requires quite a long running time, as for every point removal iteration, all remaining points must be examined. The predominant reason for the high error is an excessively simplified method for calculating curvature change.

Regular sampling was very fast, and provided reasonable results. It simply allocates a number of point to each "channel" or rotation axis, so has no time-consuming repetitive iterations. Error was still much higher than the direct methods, but this is probably the best choice of the existing algorithms for carrying out point reduction if speed is desired at the expense of some accuracy.

Lowe's algorithm is also fast, but the additional complexity lacks any additional usefulness or accuracy over regular sampling.

Both direct methods examine every possible removable point during each iteration, making them very slow, particularly for long or complex motion sequences. However, the speed can be increased several orders of magnitude by removing multiple points (a certain number, or proportion of the total remaining points) every iteration, provided one ensures that the points removed in a given iteration do not affect other points removed in that same iteration.

4.4 Usage of Compressed/Condensed Motion Capture

Each approximated motion sequence produced from these techniques contains a reduced number of information-rich points. The most clear application for information in this form is gait retargeting. Gait retargeting can be thought of as mapping motion from one character to another, differently-proportioned character. Using a much smaller number of points which still retain most of the motion has the potential to simplify, streamline, and speed up gait retargeting algorithms, particularly those techniques which use joint angle data to perform the remapping.

Additionally, the nature of error-minimization during point reduction means that the points retained are likely to be a critical stages of the motion: Turns, reversals, sudden accelerations will all be retained, as they are information-rich. Consequently, the chosen retained points may be of use in identification of equivalent movements between different motion sequences exhibiting similar motions.

A related usage is motion summarization: Analysing which frames contain the most retained points provides a means of determining the most important frames to retain when compiling a summarized version.

4.5 Data Storage

The low overhead of interpolation, particularly with our approximation technique, makes these approximated motions potentially useful for storage where memory is at a premium, but fast rendering performance is still required.

It also has potential for reducing storage space required in a database of motion sequences, especially in the case of previewing. For instance, if an animation professional wishes to browse a large set of motion capture sequences, particularly over a low-bandwidth network connection, using small yet accurate file sizes would be of value for performance and responsiveness.

5 Conclusion

We have demonstrated that retention of only 15% of motion data allows storage and reproduction of an accurate approximation of a motion capture. Decreasing detail to significantly below this level results in increased error, but an identifiable motion is still generated with only five percent of the original data present.

6 Future Work

6.1 Point Elimination Algorithm Improvements

Each of the algorithms could benefit from certain enhancements. Choosing maxima and minima purely proved to be too blunt. The error could be reduced

here if one adopted a more flexible approach and added further data points to joints weighted as important.

Likewise, the curvature change method is too blunt. There is less value in attempting any optimization here as this algorithm is the slowest of the approximations anyway.

Lowe's algorithm could be enhanced for Catmull-Rom interpolation by measuring error distances from a line that is interpolated more appropriately (rather than merely linearly).

The weighting of allocations for Lowe's algorithm could also be enhanced. There is much room for improvement in refining the weighting methodology in order to produce results closer to the direct error-reduction methods.

Optimizations may also exist for the direct methods. While these yield good results, decreasing the running time would make them more useful.

6.2 Environmental Interactions

The error metrics used do not take into account environmental interactions. If these could be incorporated into the point elimination process (possibly by weighting the branches involved with the interaction accordingly), the perceptual and quantifiable accuracy could be further improved.

References

[1] Edward Angel. *Interactive computer graphics: a top-down approach with OpenGL*. Pearson Education, Inc., 2006.

[2] Okan Arikan. Compression of motion capture databases. *ACM Trans. Graph.*, 25(3):890–897, 2006.

[3] Jackie Assa, Yaron Caspi, and Daniel Cohen-Or. Action synopsis: pose selection and illustration. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 667–676, New York, NY, USA, 2005. ACM.

[4] E. Catmull and R. Rom. A class of local interpolating splines. In R. Barnhill and R. Riesenfeld, editors, *Computer Aided Geometric Design*. Academic Press, 1974.

[5] Z. ¯Cerneková, C. Nikou, and I. Pitas. Entropy metrics used for video summarization. In *SCCG '02: Proceedings of the 18th spring conference on Computer graphics*, pages 73–82, New York, NY, USA, 2002. ACM.

[6] Nevenka Dimitrova, Thomas McGee, and Herman Elenbaas. Video keyframe extraction and filtering: a keyframe is not a keyframe to everyone. In *CIKM '97: Proceedings of the sixth international conference on Information and knowledge management*, pages 113–120, New York, NY, USA, 1997. ACM.

[7] Youssef Hadi, Fedwa Essannouni, and Rachid Oulad Haj Thami. Video summarization by k-medoid clustering. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 1400–1401, New York, NY, USA, 2006. ACM.

[8] Lucas Kovar, John Schreiner, and Michael Gleicher. Footskate cleanup for motion capture editing. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 97–104, New York, NY, USA, 2002. ACM.

[9] I. S. Lim and D Thalmann. Key-posture extraction out of human motion data by curve simplification. In *23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 2, pages 1167 – 1169, 2001.

[10] J. E. Murphy, H. Carr, and M. O'Neill. Grammatical evolution for gait retargeting. In Ik Soo Lim and Wen Tang, editors, *EG UK Theory and Practice of Computer Graphics*, 2008.

[11] R.A. Noble and R. White. Visualisation of gait analysis data. *Information Visualisation, 2005. Proceedings. Ninth International Conference on*, pages 247–252, July 2005.

[12] O. Onder, U. Gudukbay, B Ozguc, T. Erdem, C. Erdem, and M. Ozkan. Keyframe reduction techniques for motion capture data. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, pages 293–296, May 2008.

[13] Meghan P. Provost, Vernon L. Quinsey, and Nikolaus F. Troje. Differences in gait across the menstrual cycle and their attractiveness to men. *Archives of Sexual Behavior*, Volume 37, Number 4:598–604, 2008.

[14] Paul S. A. Reitsma and Nancy S. Pollard. Perceptual metrics for character animation: sensitivity to errors in ballistic motion. *ACM Trans. Graph.*, 22(3):537–542, 2003.

[15] Paul L. Rosin. Techniques for assessing polygonal approximations of curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):659–666, 1997.

[16] Jamal Saboune and François Charpillet. Markerless human motion capture for gait analysis. *CoRR*, abs/cs/0510063, 2005.

[17] Mirko Sattler, Ralf Sarlette, and Reinhard Klein. Simple and efficient compression of animation sequences. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 209–217, New York, NY, USA, 2005. ACM.

[18] Andrew Selle and Michael Gleicher. Motion editing with paths and tiles. Technical report, University of Wisconsin Madison, 2003.