Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2005)
K. Anjyo, P. Faloutsos (Editors)

An Efficient Search Algorithm for Motion Data Using Weighted PCA

K. Forbes & E. Fiume

University of Toronto

Abstract

Good motion data is costly to create. Such an expense often makes the reuse of motion data through transformation and retargetting a more attractive option than creating new motion from scratch. Reuse requires the ability to search automatically and efficiently a growing corpus of motion data, which remains a difficult open problem. We present a method for quickly searching long, unsegmented motion clips for subregions that most closely match a short query clip. Our search algorithm is based on a weighted PCA-based pose representation that allows for flexible and efficient pose-to-pose distance calculations. We present our pose representation and the details of the search algorithm. We evaluate the performance of a prototype search application using both synthetic and captured motion data. Using these results, we propose ways to improve the application's performance. The results inform a discussion of the algorithm's good scalability characteristics.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation

1. Introduction

As the corpus of information regarding virtually every human endeavour grows exponentially, the importance of computer-based indexing and searching becomes correspondingly important. The ubiquity of the Web could not have occurred without the coincident rise of the search engine. There is little value in information unless one can explore it. Once a collection of data grows to a certain size, its index becomes almost as important as its content.

Digitized motion data is expensive to create and manipulate. Its creation requires the talents of a skilled animator using specialized software, or exotic and finicky motion capture hardware. The motion data that goes into the production of a feature animation represents an investment of millions of dollars. As an animation studio accumulates more such data, it is in its best interest to leverage this investment. In order to do so, however, they need an efficient way to search the data.

In this paper, we present a method for performing a similarity search over a database of sampled motion data. This method operates independently of mark-up, and can find similarities in motion subject to non-linear temporal warping. We develop a flexible and efficient representation for motion that is amenable to optimization, and the application of existing searching and sorting techniques. The method is tested with both synthetic and sampled motion data, and we analyse the results. We conclude the paper by discussing the potential applications of the method, and mapping out avenues for future improvement.

1.1. Motivating Example

Animators often save time when creating new animations by working from prior examples. It is often more productive to modify a walk cycle to match the requirements of a particular situation than to start from scratch on every scene. As individual production studios accumulate 3D character animation, the possibilities for motion reuse at once grow and diminish. Reuse becomes potentially more fruitful, since there are more examples to choose from, but the act of actually finding useful clips gets considerably more difficult. Motions, whether they are key-framed or motion-captured, are high-dimensional objects that are hard to compare numerically. Two motions that look similar to a human observer may in fact be numerically very dissimilar using certain representational schemes. In most cases, searching though a cat-

alog for a particular type of motion quickly becomes an exercise in patience, memory, and hard work. Clearly, a method for quickly searching a database of motions is a prerequisite for large-scale motion reuse. Furthermore, it is important to develop similarity measures that can be more readily adapted to user needs.

1.2. Problem Statement

We present a method for querying a skeletal motion database with example clips. The motion database is constructed from one or more long motion sequences. These sequences can be taken from previously finished animations, unsegmented motion capture trials, or manually keyframed motion tests. All motions must be expressed over the same skeleton. As a preprocessing step, all sequences in the database are resampled to a uniform rate, and spliced together to form a single long motion. The database is queried with a short example clip, which ideally expresses one distinct motion, such as a single reach, step, punch, or jump. The search algorithm finds the subsegments of the database which are most similar to the query, subject to a nonlinear time warping. These subsegments are ranked and returned as the search results.

2. Related Work

Principal components analysis is a widespread technique that has been used before in the analysis of motion data. In several papers [PG04, GBT04], Glardon et al. construct PCA spaces based on motion clips. The spaces represent entire normalized motions as a single points. This representation allows for style-based interpolation and classification, but the motions used must be segmented first. These spaces are most useful when dealing with cyclic motions such as walking. Faloutsos et al. [BSP*04] use a motion PCA representation similar to our own in order to automatically segment motion data into separate behaviours. This is done by detecting changes in the inherent dimensionality of test motions embedded in spaces trained to particular motions. The spaces are built from raw quaternion data. Since quaternions are not closed under addition, an arbitrary point in such a space does not necessarily map back to a valid set of quaternions. This precludes the space from being used for pose interpolation or synthesis.

We use the motion data preprocessing steps outlined in Johnson's Ph.D. thesis [Joh03] to guarantee closure for our PCA pose space. This involves a linearization step, as described by Grassia [Gra98], as well as the calculation and subtraction of the sample mean pose [Joh03]. We extend Johnson's formulation by combining it with the PCA weighting scheme from Skocaj and Leonardis [SL02]. Solving for a weighted PCA space requires the use of the EM algorithm, the particulars of which are described by Roweis [Row98].

There are two routes that can be taken with rich media querying. The easier option is to perform some kind of a textual attribute mark-up, and then search using the resulting meta-data. The MPEG-7 standard [MKP02] represents the content industry's progress in this direction. There are several drawbacks to such a scheme. The quality of queries made under such a system depends upon the quality of the mark-up. In addition, the process of manually marking up data in the first place can be tedious, and it requires subjective judgment. While manual annotation will always be required to classify the qualitative or emotional aspects of motion, it should be possible to automate quantitative classification. Finally, issues can arise when multiple data sources that use different mark up schemes are merged. In our particular domain of motion data, this could happen to a studio should it acquire the rights to motion libraries from others.

The alternative to meta-data based querying is to use an automatic comparison technique. If a distance measure can be provided for the data type in question, a similarity-based search engine can be created. Unfortunately, efficient and robust distance measures are hard to design for many types of media. Salesin and Finkelstein present a wavelet-based search method for static images in [JFS95]. Their method transforms an entire image into a robust and much more compact signature. This strategy works well for discrete entities, like whole images, but is not applicable to motion data, where potential matches take the form of subintervals within a much larger time-series. In [KG04], Kovar and Gleicher create an exhaustive table of the inter-pose difference between two motion sequences of arbitrary length. With some post-processing, this table can be used to quickly find matches for segments from one motion in the other. While useful for certain applications, such as the parametric extraction task which is the major focus of their paper, the long pre-processing time precludes it from use with novel or real-time queries. In our technique, we calculate multiple smaller, carefully-targeted distance tables to reduce the complexity of the query. Given these distance tables, we use the dynamic time warping algorithm to actually calculate the distance.

Dynamic time warping (DTW) is a technique that is traditionally associated with speech recognition. Bruderlin and Williams applied it to animation parameters in [BW95]. Subsequent authors have used it to align motion clips before interpolation [KG03], and there is active research within the data mining community to improve upon the basic algorithm [CKHP02, KP99]. There are several alternatives to performing matching with DTW. One that has been used to perform motion queries [CVB*03, KPZ*04] is the LCSS-based multidimensional trajectory comparison measure proposed by Gunopulos et al. [VKG02].

A related task to motion database querying is gesture recognition. In a typical gesture recognition system, real-time input (vision or mocap-based) is compared to a library of predefined target gestures. This differs from our task in that the targets are neatly segmented, while the query is

continuous. Many authors have successfully employed Hidden Markov Models in order to detect and classify gestures [CBA*96, vHB01].

3. Search Algorithm

In this section we describe the motion search method and its related algorithms and data structures. We explain the representational framework in which the method operates. Next, we give an overview of the system itself. This is followed by a detailed description of each step in the process.

3.1. Motion Representation

In its raw form, motion data is not easy to work with. Much of the difficulty stems from the lack of an inherent distance function between poses. Researchers have used many different approaches in their own motion work, such as the deformed point-cloud method described by Kovar et al. [KGP02], or the weighted sum of quaternion distances proposed by Johnson [Joh03]. We present a weighted-PCA based representation for poses that has a Euclidean distance metric. The simple distance metric allows for the direct application of standard data processing techniques. Being PCA-based, our representation also benefits from having a coarse-to-fine interpretation, which allows for less accurate, but quicker distance calculations. A similar technique was used in [AFO03], albeit without the use of weighting.

Principal Components Analysis is a change-of-basis transformation that imposes a structure upon the resulting space that models the dimensions in which the greatest correlations and variations occur [Bis96]. The result of performing PCA on a given dataset is a vector space with the same dimensionality. Each axis in the space represents a principal component vector. Any point in the space is thus a weighted combination of the principal components. If the principal components are ordered according to the amount of variance that they describe in the original dataset, the variances show an exponential drop-off. This is what allows for dimensionality reduction: a full data point can be represented with a predictable degree of fidelity by using some subset of its PC coordinates.

With some massaging [Joh03, LWS02], motion data can be used to construct a PCA space. Such a space, however, will not take into account the hierarchical nature of the pose data. Perceptually speaking, a few degrees of change in the angle of a shoulder changes the shape of a pose much more than a similar change in a toe. In fact, 'noisy toes' can threaten to dominate the PCA space, and lead to an inefficient distribution of the motion's degrees of freedom over the principal components. This in turn increases the number of dimensions that must be used to produce acceptable looking motion. In order to prevent this, we use weighted PCA. Skočaj and Leonardis present a wPCA formulation for vision applications, wherein weights can be applied to both subsections of individual frames, and to entire frames

Algorithm 1 Creating the wPCA space

Ensure: $X \leftarrow$ linearized Range of Motion data
  Find the mean pose
  for all samples in ROM **do**
    for all DOF **do**
      Rotate out the mean quaternion
      Linearize the result
      Accumulate in matrix X
    end for
  end for

Ensure: $X = wU \times A$
  $U \leftarrow$ random values
  while $(its < maxIts) \bigwedge (reconError < \varepsilon)$ **do**
    E Step: QR Solve for projection A
    M Step: LU Solve for space vectors U
  end while

  return the orthogonalized columns of U as the PCs

[SL02]. We use only the former, and apply a real-valued weight to each joint. The specific weights used can be manipulated to change the properties of the resulting space, as we will show later. In the general case, we use weights that are derived from an approximation of the relative amount of body mass that is influenced by the movement of each joint. Pseudocode for the wPCA construction algorithm is given in algorithm 1.

When projected into the wPCA space, motions become high-dimensional parametric curves, with each sample point representing a single pose. This projection is lossy if the projected motion was not used in the construction of the space, but such losses can be minimized by using a good range of motion trials for training. The distance between two poses can be calculated by taking the scaled L2 norm of their PC coordinates, or equivalently by scaling the points and taking the regular L2 norm. The scale factors for these calculations are the eigenvalues which are given as a by-product during the creation of the PCA space. Since the magnitude of these scale factors drops off exponentially, the distance calculation can be truncated after a small number of dimensions if speed is required before accuracy. The weighting scheme also serves to warp the space, changing the nature of the pose distance metric. Because of the exponential eigenvalue decay, the first few principal components dominate the distance metric. Variations across degrees of freedom that are heavily weighted will be reflected in the earlier principal components. By changing the weighting scheme, the user is able to target the pose distance metric to different types of motions. For example, when comparing walking motions, it may be advantageous to use a space that is heavily weighted toward the legs. Different spaces can be trained off-line and saved to disk, allowing the user to quickly select a relevant space at work with at run-time.

Various operations that are complex to perform with the

Figure 1: *A 27 second clip of boxing motion projected into a wPCA space.*

original quaternion-based motion representation are greatly simplified using the wPCA representation. Pose means can be calculated by simply averaging the the PC coordinates of the target frames. Interpolation is likewise simplified – linear interpolation in wPCA space gives results similar to spherical linear interpolation in the original representation. In addition, geometric operations such as scaling and translating can be used to modify motions. These operations can be coupled with a 3D display of the first three dimensions of a projected motion's pose-points to present a compelling motion editing interface. An example of such a projection is shown in figure 1.

The motion data with which we work is joint-based. A single sample consists of a list of quaternions corresponding to each joint's angular position. When applied to a hierarchical skeleton, these quaternions (along with a pelvis offset vector) fully describe a full-body pose. We assume a constant sampling rate across all clips in the database. We treat the position and heading of the skeleton's root joint as something external to the pose. This allows for poses to match regardless of orientation. One limitation of the linearization step that we used is that individual joints can not travel a full 360 degrees. This is not a issue for most internal joints, but is problematic for the root joint. Acrobatic motions such as cartwheels would necessitate breaking the root joint's X and Z rotation out of the PCA space representation, much like the heading.

3.2. System Overview

Given our motion representation and pose distance metric, we will now describe our motion search strategy. Our search application is similarity-based. This means that the user must have an existing motion clip with which to query the system. This clip could be from a library of pre-segmented, canonical actions, the results of a previous query, or even from real-time motion capture. As a preprocessing step, both the query clip, and the database are projected into a user-specified wPCA space. Projecting the database is an expensive operation, but it only needs to be done once for each wPCA space, and the results can be placed in permanent storage. After the query clip is projected, its characteristic pose is found. An efficient spatial sorting data structure is then used to find the indices of all similar poses in the database. These indices are clustered to reduce redundancy, and then a variant of the dynamic time warping algorithm is used to warp the database subregions surrounding the cluster means to match the query clip as closely as possible. The resulting warps are ranked according to fit, and returned as the search results. We will now describe each step in this process in detail. Pseudocode for the querying operation is given in algorithm 2.

Algorithm 2 Performing a Query

Require: projected database and query clip, and offset to characteristic pose in query

```
Perform an Approximate Nearest Neighbour search query
with the characteristic pose

for all ANN results r do
  if r can be joined with an existing cluster c then
    Grow cluster c, join with neighbours if necessary
  else
    Create a new cluster initialized with r
  end if
end for

for all Cluster min points do
  Calculate the forward and backward distance tables
  Find the min forward and backward paths
  Join the two half paths
  calculate the mean warp distance
end for

return the sorted warp paths
```

3.2.1. Finding the Characteristic Point

Queries represent single, coherent motions. Such motions can often be expressed using single poses [McC94]. We call these poses *characteristic points*, and we will use them as starting points in our motion search. First, however, we must come up with a workable definition of "characteristic".

A good characteristic point for a punching motion would be the moment of maximum arm extension. All punching motions contain some element of arm extension. Most verb-level action descriptions, such as stepping, jumping, or ducking imply some similar common element. In a step motion, the characteristic point could be moment when the legs are farthest apart. The characteristic point of a jump might be at its apex. Likewise, a ducking motion might be characterized by its lowest point. The common thread in all of these examples is that the characteristic point represents a moment of maximal extension or deviation from some neutral pose.

This concept fits well with our motion representation. If we define the neutral pose to be some point in the wPCA space, we can find the characteristic point with respect to that point by searching for the most distant pose from that point. The neutral pose can be defined in any number of ways. If the

action phase of the query is proportionally short, the mean pose of the whole motion provides a good approximation. If the query is nicely segmented, a reasonable assumption may be that the subject begins and ends in a neutral pose. Either boundary pose could be used directly, or alternately the mean of the two could be taken. The origin of the PCA space represents the mean pose of the (probably significantly longer) motion used in its creation, so it can also be used as the neutral point.

Our objective in choosing a characteristic point is to find class of poses that is guaranteed to have a close analogue in all possible matching motions, but is unlikely to exist in non-matches. If the pose is too common, spurious matches will drown out the actual results during the next step of the algorithm. For this reason, finding a suitable characteristic point is a crucial task. Of course, if the query is quite short, it is not unreasonable to require the user to specify a characteristic point directly. For certain types of queries, this gives better results than the automatic methods.

3.2.2. Generating Seed Points

An exhaustive solution to our search problem would be to use DTW to rank all possible alignments of the query clip and the database. This is analogous to the technique in [KG04], but is slow because the DTW operation is expensive. In order to provide interactive response rates to the user, we must cull the search space before the DTW step. We refer to this culling as finding the seed points in the database. Seed points are the indices of poses in the database that are similar to the characteristic point of the query.

Our measure of similarity is the euclidean distance within the scaled wPCA space, so we can use algorithms from computational geometry to speed our search. We also have to choose the number of dimensions within which we will operate. The weighting scheme used to construct the wPCA space greatly influences the results of a search, which can be exploited to considerable advantage in searching selectively. The weights should be picked by the user to reflect the constraints of the animation for which s/he is searching.

There are several different search structures that would work for our implementation. We chose to use Approximate Nearest Neighbour search because of its quick running time, flexibility, and readily available source code [Mou05]. The effects of varying the various parameters of the ANN software are discussed in the results section.

3.2.3. Seed Point Clustering

Motion in the database takes the form of contiguous, time-ordered strands of pose-points. Nearest neighbour searches within such a context result in sequences of temporally adjacent points. Since we will be subjecting the seed points to the DTW algorithm in the next step, all of these points will return valid, yet similar results. We cluster the seed points in order to avoid overwhelming the user with hundreds of very similar results, and to improve the search's run time.

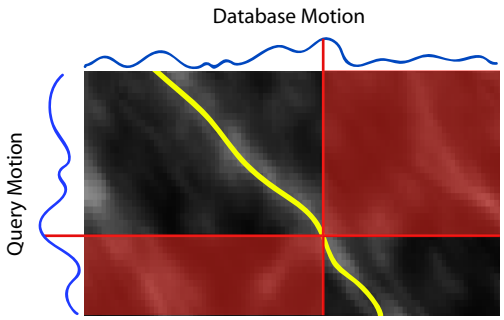

Figure 2: *The DTW constraints. The warp must pass through the intersection of the characteristic and seed points, and contact both horizontal edges of the table. The search is constrained by causality, so distances in the shaded areas of the table need not be calculated.*

The clustering is performed on the seed points' time indices. This data, since it is one dimensional, integer-valued, and mostly sequential, is very well-behaved and easy to cluster. Data points are simply collected into contiguous (to within a noise term) intervals as they are found. The final results are given as the closest points within each cluster.

3.2.4. Dynamic Time Warping

The query signal has a well-defined start and end point, but we have no such luxury when looking for a subsequence within the database. The search is further complicated by the fact that motions tend to be performed slightly differently each time they occur. Changes in motion timing which are subtle to a human observer may cause an enormous numerical difference.

Both of these issues are surmounted through the use of dynamic time warping. DTW is a signal processing technique that finds a non-linear alignment minimizing the error between two signals. It returns a time displacement function that compresses and dilates one of the functions to match the other. DTW has been used extensively on sound signals for speech recognition, and is often used to improve the interpolation of multiple motion clips [BW95, KG03].

Each clustered index represents a single moment of similarity between some point in the database, and the characteristic point of the query. A valid time warp must pass through this point. The warp is also constrained to run from the beginning to the end of the query. These constraints are visualized in figure 2.

We can divide the time warp into two subproblems: one running forward in time and one running backward. The method for solving each subproblem is identical. First, a distance table is computed involving the pertinent half of the query and the corresponding section of the database. A slope limit imposed on the final warp provides a bound on the size of the distance table. Starting from the characteristic point,

each cell in the table is filled with the sum of the pose distances between the indexed animation frames and the minimum of its previously filled neighbours. Once the table is filled, the minimum value along the query's boundary frame is found. The DTW path is then found by greedily searching through the table toward the characteristic point. This search is subject to causality, so there are only at most three possible steps to take at any given point. The slope limit is enforced to prevent degenerate warps. Degeneracies in the warp are still possible around the characteristic point, but these can be culled out during the results ranking.

When the characteristic point is in the middle of the query, splitting the DTW into two problems halves the number of distance calculations required. There are several methods available to further reduce the number of calculations, and/or improve the warp quality [CKHP02, KP99]. Our pose distance metric is quick enough that this has not been necessary to achieve interactive rates with the test data that we have used. Another desirable feature of the distance metric is that it is possible to trade accuracy for speed, and use less than its full dimensionality in the calculation.

3.2.5. Results Ranking

The time warps must be scored before they can be returned as ranked search results. Any of a number of motion distance measures can be used. We define the final score of a warp to be the average pose distance of each cell in its path. This measure does not penalize warping, so it is more forgiving of timing differences in the results. Alternative measures could take into account the effects of outliers along the path, or put a premium on time distortions. It may also be useful to cull results that have large degeneracies about the critical point, or at least penalize them so that they are lower ranked.

4. Results

4.1. Synthetic Data

We first used synthetically-generated data to verify the functioning of our system. This allowed us produce clean motion clips with controlled variations in movement parameters. We used the physically-based animation system designed by Neff and Fiume to generate this data procedurally [NF04].

Two synthetic motion sets were generated. The first is approximately 300 seconds long, sampled at 50 fps. The figure begins by raising its right arm 15 times. The exact position of each raised hand was selected from a $10cm^3$ cube using a uniform random distribution. The posture of the figure was randomly set along the Alberts axis from .25 to .75 [NF04]. Finally, the overall timing of each motion was scaled, ranging from .7 to 1.5 times the normal length. The figure then performs 20 similarly varied left arm raises, and 20 double-arm-raises. It finishes by performing 10 identical shrugs at different speeds, and 10 slouches. The second dataset only contains arm raises, but the bounds for the arm targets are increased in the x and y directions by a factor of three.

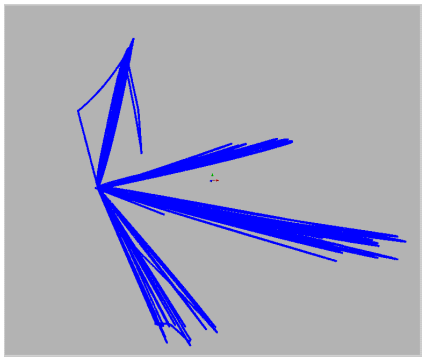

Figure 3: *PCA projection of synthetic motion, showing low inherent dimensionality.*

Neff and Fiume's animation system uses an SD-Fast derived physical model [HRS94]. Using the measurements provided in the SD system definition file, a similar skeleton definition was created. The mass definitions from the file SD file were used to set the wPCA weights: each joint was weighted with the amount of mass under it in the skeleton hierarchy. No range of motion trial was available to train the PCA space, so we used the longer of the two samples.

A 3D projection of the long motion clip is shown in figure 3. The individual motion classes stand out as the path extremities. The sample mean of the training data is the same as the rest position in this clip, and is represented by the cluster of points at the origin. More complicated motions embedded within spaces generated from richer training data take on a much less angular appearance when projected in three dimensions. This is consistent with the fact that the synthetic data was designed to have a low inherent dimensionality.

4.2. Validation

Validation was performed using the synthetic data. The first of each type of arm raising motion was manually segmented from the longer motion clip, and then used to query both motion clips. The quality of the results of the queries were highly dependent upon the characteristic point used, and the size of the initial ANN search.

The automatic characteristic point finder did not work well with the arm movements, because the time that the hands are raised is very long in relation to the length of the whole clip. This shifted the clip's mean point away from the rest position, and put the characteristic point down near the rest pose. Using the rest pose as the query point in the ANN search lead to mostly spurious results, with the time warping algorithm left trying to match essentially random segments of the database to the query. Manually setting the characteristic point to the moment of maximum arm extension and re-querying allowed the algorithm to proceed as designed.

The size of the initial ANN search determines the broadness of the results. The structure of the motion data is such

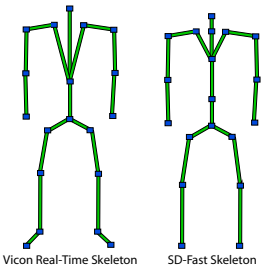

Figure 4: *The two skeletal structures used.*

Figure 5: *wPCA space projection of a martial arts move.*

that a spatial proximity search returns long chains of time-adjacent points for any query. We simplify the results in the clustering phase, but in order to find all valid target clusters, the initial search must be set to return a very large number of points. For example, when querying the long clip with the raised left hand motion, the ANN search must be widened to 1130 points before all of the raised left hand targets are found. If the search is widened even further, other types of motions start to creep in to the results. At 1900 points, several instances of the raise both hands action are returned after all of the raise right hands. This is good behaviour, since the 'raising both hands' motion is perceptually closer to raising only the right hand than any other action in the database. The motion distance measure also holds up, since it consistently ranks the second tier of matches below the actual matches.

Interestingly, querying the database with one of its own motions does not always most highly rank that motion. This is due to several factors. During the manual segmentation of the query clips, the designated motion range is re-sampled, introducing small numerical differences. With synthetic data, the figure holds absolutely static poses at several points during its performance. These quiescent points tend to occur at the characteristic points, so the ANN search often finds several identically closest points. The clustering technique has no way to distinguish among these points, so it picks the first. This causes some noise in the alignment of the database ranges with the query clip, which is corrected by the time warping step. This correction has a non-zero cost, so self-matching does not necessarily hold.

4.3. Motion Capture Data

After verifying the system with synthetic data, we tested its real-world applicability with motion capture data. The data was collected using a Vicon optical motion capture system, and post-processed into joint angles using the Vicon IQ software. This data uses a different skeleton than the synthetic data, having the same dimensionality (19 joints), but a different arrangement (see figure 4). The Vicon samples at 120 Hz.

Many individual motion trials were captured, with each being on the order of a couple minutes in length. Specific range of motion trials were made to train the PCA spaces. Trials included walking, cyclic motions, martial arts moves, and others. As with the synthetic data, query clips were manually segmented from the longer trials. An example projection is shown in figure 5.

Searching with the real data worked well, but the results were not as clean as those from the synthetic tests. An illustrative example is a search done using recorded Aikido movements. The actor in the Aikido motion trial performed a specific script. By tinkering with the location of the characteristic point, it is possible to cause the system to return movements in the wrong order. The matches are still consistent, however, with the bodies being in similar, if not identical poses. The ranking that our search method provides is somewhat arbitrary, much like the ranking of web pages returned by most web search portals.

Differently-weighted PCA spaces can be used to modify the results. Re-doing the previous example with a weighting scheme that emphasizes the arms improves the ranking of the results. This is because the position of the arms is what most clearly differentiates the moves. An animator can use different spaces to accomplish specific goals. For example, when animating a character picking up an object, it would be a good idea to use a space weighted heavily toward the hand the character is using.

4.4. Scalability

One of the most important features of a search algorithm is its scaling behaviour. In this section we discuss the theoretical complexity of our algorithm and present some search heuristics. We then provide experimental results that demonstrate the efficacy of these techniques.

The complexity of the overall search algorithm is best evaluated in individual sections. The complexity of the ANN k-nearest neighbour search in d dimensions over n points and with error bound ε is shown to be $O((c_{d,\varepsilon}+kd)\log n)$ in [AMN*94], where $c_{d,\varepsilon}$ is a constant dependent upon d and ε. This gives good performance with large databases, but is quite sensitive to dimensionality. The clustering technique that we used has a worst-case complexity of $\frac{r(r+1)}{2}$, and returns c clusters, where r is number of results returned by the ANN search. Each application of our DTW algorithm

Length (s)	Num. Results	Query Time (ms)
54	46	580
100	21	263
130	30	662
230	40	854
300	41	877
440	25	534
670	27	580

Table 1: *Query Time vs Database Length*

requires at most $s \times q^2$ and at least $\frac{(s \times q^2)}{2}$ pose distance calculations to build the distance table, and $3s$ comparisons to trace a path through the table (where q is the length of the query and s is the slope limit). Sorting the c warps takes $c \log c$ comparisons. The aggregate complexity of all of the steps is

$$(c_{d,\varepsilon} + kd)\log n + \frac{r(r+1)}{2} + c(s \times q^2 + 3s + \log c).$$

We tested our system with motion capture databases of various sizes to experimentally evaluate its scalability. The results of the tests are given in table 1.

The data shows that our implementation's real-world performance largely depends upon the number of results passed on to the DTW step, rather than the size of the database. The databases used for the last two trials contained exact matches for the query. It follows that the area around the characteristic point would be dense with consecutive poses, which would lead to more clustering, and fewer returned results. This emphasizes the importance of a having both a unique characteristic point and a good clustering result. It also indicates that the best performance can be gained by tweaking the DTW parameters. All tests in this paper were performed using a 3GHz Pentium 4 computer.

4.5. Performance Optimization

There are several ways to improve the running time of the algorithm. Most involve a trade-off of either search breadth or accuracy for query time performance. In order to build a basis for comparison, we set up what could be considered an average query. We used the Aikido data described in the previous section, and the search parameters listed in table 2. Using the automatically generated characteristic point, the average query time (from 10 trials) was 347 ms, and the search returned 12 results. Using a manually specified characteristic point, the average query time was 130 ms, with 9 results returned. The results from the manual characteristic point trial were subjectively much closer to the query than those from the automatic trial, so the manual point was used in all subsequent trials.

4.5.1. Pre-smoothing the data

The Vicon system samples motion at a default rate of 120 Hz–a much higher rate than is necessary to capture most large muscle movements. Reducing the sampling rate obviouly reduces search time. We resampled the database and queried at a progression of sampling rates to illustrate the effects of this reduction on average query time. The results are shown in table 3. As expected, using fewer samples greatly speeds up the algorithm. The quality of the results is consistent until the low sampling rate conflicts with the clustering algorithm. At very low rates, the gaps between clusters all get filled, and the system fails by returning a single result.

Data Sampling rate	120 Hz
Database length	70 seconds
Query Length	2.8 seconds
ANN dimensionality	10
ANN search size	1000
ANN epsilon	0
DTW dimensionality	5

Table 2: *Baseline Parameters*

Hz	Num. Results	Query Time (ms)
233	9	453
166	10	158
58	9	29
29	9	7

Table 3: *Query Time vs Sampling Rate*

4.5.2. Adjusting the ANN parameters

The scalability results indicate that the performance barrier in the the system lies with the DTW stage. That being said, an investigation of the effects of the ANN parameters is important, if only just to verify the earlier result.

The number of results returned by the ANN search affects both its own running time, and the number of final results after the clustering step. Using the default setup as a baseline, we varied the number of neighbours to be returned by the ANN search. The results are given in table 4. With a low number of neighbours, many potential results are missing. As the number of neighbours increases, there tends to be more results returned. After a point, the seed points' neighbourhoods grow too large, and incorporate points that they should not, which in turn causes the clustering algorithm to create improper clusters. The 'sweet spot' between too few results and over-clustering depends upon the specific nature of the database and query being used.

One of the most desirable features of ANN search is that it can deliver improved performance if a measure of error is accepted. Paradoxically, increasing this error tolerance leads to reduced performance in our system. Non-exact results caused gaps in the runs of nearest neighbour poses, which leads to an increased number of clusters. Adjusting the dimensionality of the data used for the ANN search had a negligible effect upon the running time of the overall system.

Neighbours	Num. Results	Query Time (ms)
100	4	54
250	5	70
360	6	80
490	8	109
600	8	118
1000	9	132
1500	9	137
2500	12	193
3010	17	256
5000	12	219

Table 4: *Query Time vs Neigbours*

Dimensions	Num Results	Query Time (ms)
1	9	70
3	9	101
5	9	130
20	9	200
54	9	372

Table 5: *Query Time vs DTW Dimensions*

4.5.3. Adjusting the DTW parameters

The running time of the search algorithm directly corresponds with the number of clustered results that make it to the DTW phase, so the time warping algorithm is good area upon which to focus optimizations. An easy optimization is to exploit the flexibility of our motion representation by reducing the dimensionality of the pose distance metric. As the shown in table 5, reducing the accuracy of the distance comparisons does improve the system's run time. For the types of motions tested, the quality of the warp is not subjectively affected by reducing the dimensionality. This of course depends upon the distribution of the principal components. The user is required to pick a set of weighted PCA bases that reflect the content of the motions that s/he is using.

5. Applications and Future Work

Our search system was designed to be integrated into a larger motion editing system utilizing our pose space representation. In this context, the search system can be used to find motions that are similar to a specified clip, so that fine adjustments can be made via interpolation. It can also be useful as a motion exploration tool. A clip created using the various editing tools can be used as a query in order to find a similar, but more realistic motion. The time warping code prototyped for the search system is also useful for improving the quality of arbitrary interpolations.

A possible use of the system would be to quickly apply markup to a large motion database. This procedure would start with a small set of manually marked-up clips. The mark-up would take the form of a ⟨descriptor,value⟩ tuples, where descriptor describes the motion's action, and value indicates how well the descriptor fits. For example, if the descriptor is 'step forward', an unambiguous step forward might have the value of 1, while a step forward and to the left would have a lower value. Queries could be performed using each of the motions in the marked-up set. The ranges that are returned from each query would then take on the descriptors of the queries, with values set to be a function of the query's values and the result's ranking score. After mark-up, semantic queries could be made to database very quickly.

The system could be extended to work with more complicated motions by adding support for more than one characteristic point. If a query were determined to have multiple characteristic points, the modified algorithm would start by finding and clustering seed points for them all. The dynamic time warping phase of the algorithm would have to be modified to take into account multiple constraints. It would work by finding seed points in the same order as their corresponding characteristic points. Warps would then be found between adjacent matching pairs. The adjacency information could then be represented as a directed graph, and all possible traversing paths enumerated.

6. Conclusion

In this paper we have presented a search algorithm for use with sampled motion data. In doing so we have also developed a representation for motion data that introduces a meaningful distance metric for poses. We have shown how an animator can control the properties of the wPCA space through its weights, and how this may be used to direct the search results. We have demonstrated the use of the search algorithm on both real and synthetic data, and have analyzed its performance. Finally, we have experimented with the algorithm's settings in order to gauge its scalability to large databases.

7. Acknowledgments

Financial support for this work was provided by NSERC. The software prototype used ANN code written by David Mount and Sunil Arya from the University of Maryland. Michael Neff helped to create the synthetic motion data used to validate the system.

References

[AFO03] ARIKAN O., FORSYTH D. A., O'BRIEN J. F.: Motion synthesis from annotations. *ACM Trans. Graph. 22*, 3 (2003), 402–408.

[AMN*94] ARYA S., MOUNT D. M., NETANYAHU N. S., SILVERMAN R., WU A.: An optimal algorithm for approximate nearest neighbor searching. In *SODA '94: Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms* (1994), Society for Industrial and Applied Mathematics, pp. 573–582.

[Bis96] BISHOP C. M.: *Neural networks for pattern recognition*. Oxford University Press, Oxford, UK, UK, 1996.

[BSP*04] BARBIC J., SAFONOVA A., PAN J.-Y., FALOUTSOS C., HODGINS J. K., POLLARD N. S.: Segmenting motion capture data into distinct behaviors. In *GI '04: Proceedings of the 2004 conference on Graphics interface* (2004), Canadian Human-Computer Communications Society, pp. 185–194.

[BW95] BRUDERLIN A., WILLIAMS L.: Motion signal processing. In *SIGGRAPH* (1995), pp. 97–104.

[CBA*96] CAMPBELL L. W., BECKER D. A., AZARBAYEJANI A., BOBICK A. F., PENTLAND A.: Invariant features for 3-d gesture recognition. In *FG '96: Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition (FG '96)* (1996), IEEE Computer Society, p. 157.

[CKHP02] CHU S., KEOGH E. J., HART D., PAZZANI M. J.: Iterative deepening dynamic time warping for time series. In *SDM* (2002).

[CVB*03] CARDLE M., VLACHOS M., BROOKS S., KEOGH E., GUNOPULOS D.: Fast motion capture matching with replicated motion editing. In *SIGGRAPH 2003, Sketches and Applications* (jul 2003), ACM Press.

[GBT04] GLARDON P., BOULIC R., THALMANN D.: Pca-based walking engine using motion capture data. In *Computer Graphics International* (2004), pp. 292–298.

[Gra98] GRASSIA F. S.: Practical parameterization of rotations using the exponential map. *J. Graph. Tools 3*, 3 (1998), 29–48.

[HRS94] HOLLARS M. G., ROSENTHAL D. E., SHERMAN M. A.: *SD Fast User's Manual*, 1994.

[JFS95] JACOBS C. E., FINKELSTEIN A., SALESIN D. H.: Fast multiresolution image querying. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1995), ACM Press, pp. 277–286.

[Joh03] JOHNSON M. P.: *Exploiting Quaternions to Support Expressive Interactive Character Motion*. PhD thesis, Massachusettes Institute of Technology, 2003.

[KG03] KOVAR L., GLEICHER M.: Flexible automatic motion blending with registration curves. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 214–224.

[KG04] KOVAR L., GLEICHER M.: Automated extraction and parameterization of motions in large data sets. *ACM Trans. Graph. 23*, 3 (2004), 559–568.

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs, 2002.

[KP99] KEOGH E. J., PAZZANI M. J.: Scaling up dynamic time warping to massive dataset. In *PKDD '99: Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery* (London, UK, 1999), Springer-Verlag, pp. 1–11.

[KPZ*04] KEOGH E., PALPANAS T., ZORDAN V., GUNOPULOS D., CARDLE M.: Indexing large human-motion databases. In *VLDB 2004* (2004).

[LWS02] LI Y., WANG T., SHUM H.-Y.: Motion texture: a two-level statistical model for character motion synthesis. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM Press, pp. 465–472.

[McC94] MCCLOUD S.: *Understanding Comics*. Perennial Currents, 1994.

[MKP02] MARTÍNEZ J. M., KOENEN R., PEREIRA F.: Mpeg-7: the generic multimedia content description standard. *IEEE Computer Society* (2002), 78–87.

[Mou05] MOUNT D. M.: *ANN Programming Manual*, 2005.

[NF04] NEFF M., FIUME E.: Methods for exploring expressive stance. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2004), ACM Press, pp. 49–58.

[PG04] P. GLARDON R. BOULIC D. T.: A coherent locomotion engine extrapolating beyond experimental data. In *CASA 2004 Proceedings* (2004).

[Row98] ROWEIS S.: Em algorithms for pca and spca. In *NIPS '97: Proceedings of the 1997 conference on Advances in neural information processing systems 10* (1998), MIT Press, pp. 626–632.

[SL02] SKOČAJ D., LEONARDIS A.: Weighted incremental subspace learning. In *Workshop on Cognitive Vision, proceedings* (Zurich, Switzerland, September 19-20 2002).

[vHB01] VON HARDENBERG C., BRARD F.: Bare-hand humancomputer interaction. In *Proceedings of Perceptual User Interfaces, 2001.* (2001).

[VKG02] VLACHOS M., KOLLIOS G., GUNOPULOS D.: Discovering similar multidimensional trajectories. In *In Proc. of 18th ICDE, San Jose, p. 673Ű684, CA, 2002.* (2002).