Fast Motion Capture Matching with Replicated Motion Editing

Marc Cardle[*] Michail Vlachos[†] Stephen Brooks[*] Dimitrios Gunopulos[†]

[*]*University of Cambridge* [†]*University of California, Riverside*

1 Introduction

The possibility of storing large quantities of human motion capture, or 'mocap', has resulted in a growing demand for content-based retrieval of motion sequences without using annotations or other meta-data. We address the issue of rapidly retrieving perceptually similar occurrences of a particular motion in a long mocap sequence or unstructured mocap database for the purpose of replicating editing operations with minimal user-input. One or more editing operations on a given motion are made to affect all similar matching motions. This general approach is applied to standard mocap editing operations such as time-warping, filtering or motion-warping. The style of interaction lies between automation and complete user control.

Unlike recent mocap synthesis systems [1], where new motion is generated by searching for plausible transitions between motion segments, our method efficiently searches for similar motions using a query-by-example paradigm, while still allowing for extensive parameterization over the nature of the matching.

2 Motion Capture Matching

The animator selects a particular motion, by specifying its start and end time, and the system searches for similar occurrences in a mocap database (Fig 1). For maximum usability, our mocap matching engine must provide near real-time response to user queries over extended unlabeled mocap sequences, whilst allowing for spatial and temporal deviations in the returned matches. To this end, an unsupervised non-realtime pre-processing phase is required to automatically build up an efficient search index capable of guaranteeing *no false dismissals*. Our novel technique works by splitting up the each joint curve and its associated angular velocities in multi-dimensional Minimum Bounding Rectangles and storing them in an R-Tree. Support for Dynamic Time-warping and Longest Common Subsequence similarity measures caters for noisy motion with variations in the time axis thus producing robust and intuitive correspondence between keyframes. Using our index, we rapidly locate potential candidates within a dynamically user-defined temporal range; that is, matches that are up to *k* percent shorter or longer. Spatial match precision can also be dynamically altered to selectively retain dissimilar matches.

The animator has the crucial ability to interactively select the body areas utilized in the matching, so that, for example, all instances of a grabbing motion are returned, irrespective of the lower body motion. Negative matches can also be defined to preclude certain motions in the returned matches. For instance, if the user selects a kicking motion as a positive example and a punching motion as a negative example, then only matches with a kick, without a simultaneous punch, will be returned. Finally, in the case where many potential matches exist, the query results are clustered to identify the representative medoid of each cluster. These are simultaneously displayed so that the animator can rapidly dismiss undesirable classes of matches.

3 Replicated Editing Operations

As well as being a practical and efficient mocap search mechanism, our system can repeat an editing operation performed

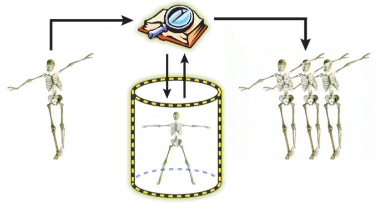

Figure 1 Matches to the user query motion are returned in realtime using an efficient search index, previously extracted from the motion database.

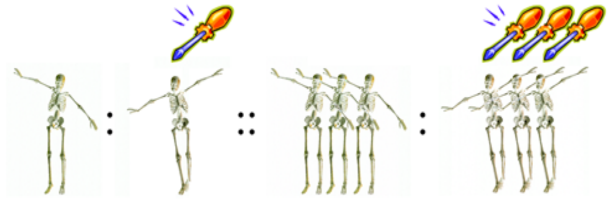

Figure 2 The animator's motion edit performed on the query motion is automatically replicated over all found matches.

on the query motion onto all selected matches (Fig 2). For example, one could quickly exaggerate every instance of a punch in the whole database by altering the EMOTE *Effort* parameters [2], as well as blending in a head-nodding motion. A variety of motion editing operations are supported such as time-warping, motion displacement mapping, motion-waveshaping, blending, smoothing and various other linear and non-linear filtering operations. Our index supports updates in sub-linear time so searches including the newly edited matches, or completely new sequence inserts, are possible with minimal wait.

For certain editing operations, such as displacement mapping, it is preferable to locally tailor the original transformation before applying it to each match. To this end, dynamic time-warping is used to automatically establish the optimal sample correspondence between the query motion and each match. The original transform, such as the displacement map, is then remapped in time according to the calculated warp. This automatic alignment ensures that edits are meaningful and useful across all matches. Additionally, the strength of applied transforms can be made proportional to the match strength, so that dissimilar motion will be less affected by the edits.

4 Conclusion and Future Work

Animators should find this tool particularly valuable since editing motion capture is still very laborious and time consuming. The benefits are especially manifest when dealing with exceedingly long mocap sequences or large mocap databases. Replicated editing can further be improved by investigating more ways of locally adapting the original edit to each motion match.

References

[1] ARIKAN, K, AND FORSYTH, D, 2002. Interactive motion generation from examples. *ACM SIGGRAPH 2002*.

[2] CHI, D, COSTA, D, ZHAO, L, BADLER, N, 2000. The EMOTE model for Effort and Shape. *ACM SIGGRAPH 2000*.

email: { [*]mpc33 | [*]sb329 }@cl.cam.ac.uk, { [†]mvlachos | [†]dg }@cs.ucr.edu
video: http://www.cl.cam.ac.uk/users/mpc33/sketch03.html

Appendix: Fast Motion Capture Matching with Replicated Motion Editing

Marc Cardle[*] Michail Vlachos[†] Stephen Brooks[*] Dimitrios Gunopulos[†]

[*]*University of Cambridge* [†]*University of California, Riverside*

1 Introduction

This appendix delineates the underlying motion capture (mocap) matching techniques used in our sketch. The search index format, the supported similarity measures and the steps necessary to process a user query are detailed.

The reader is invited to view our demonstration video on the attached CD or online[1].

2 Indexing Motion Capture Sequences

Linear search on a large multi-dimensional mocap database is not feasible for real-time query response. Consequently we perform extensive dimensionality reduction to obtain a fast searching process. For that reason, we 'split' the captured sequences into a number of equi-length Minimum Bounding Rectangles (MBRs). Specifically to index and query the mocap database we follow these steps (Fig. 1):

1. The individual mocap sequences are segmented into MBRs of length m (user defined).
2. The resulting MBRs are stored in a multi-dimensional R-tree [3].
3. User queries are similarly segmented and probed into the index.
4. Based on the MBR distance (or overlap), similarity estimates are calculated between the query and the indexed sequences. These estimates 'guide' our search as to which sequences are most likely similar to the given query. For the most promising sequences, the exact distance function is calculated on the raw data, and the top-*k* results are kept in a priority queue.

The Euclidean Distance between the segmented 1D sequences *Q* and *S* (which underestimates the distance on the raw data) is given by the following equation:

$$D_{LB} = \sum_{i=1}^{k} \begin{cases} 0 & \text{if no MBR overlap} \\ D(MBR_{Qi}, MBR_{Si}) * \|MBR\| & \text{otherwise} \end{cases}$$

Figure xxx demonstrates the above steps when the Euclidean distance is used for describing the similarity between sequences. In following sections we will provide further details on the index construction. We will also show how this methodology can be extended in order to support more flexible matching using the Longest Common Subsequence (LCSS) or Dynamic Time Warping (DTW) distance.

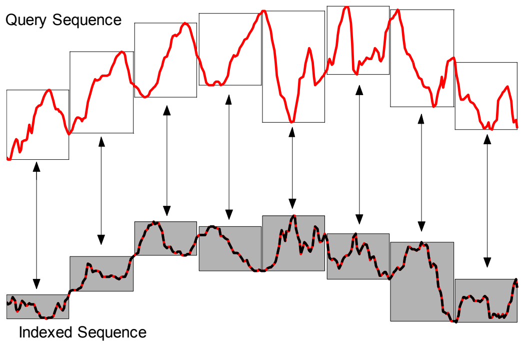

Figure 1: Index Matching. The index stores rectangles (*gray*) that approximate the sequence motion. The query is segmented, too, and the resulting MBRs are probed in the index.

3 Indexing Structure

The distance computed between the sequence MBRs is a *lower bound* of the actual Euclidean distance. Therefore, according to the GEMINI framework [1], our index structure will guarantee no false dismissals. The fundamental result of this observation is that our index will retrieve the *same* top-*k* matches, as the ones that the sequential scan would have returned. The mocap data used for this research consisted of 57 Euler joint curves. It is well known that the performance of the R-tree degrades rapidly for increasing dimensionalities [2]. Our solution to this problem is to utilize a 2-dimensional R-tree, where different joints (e.g. upper-leg X rotation or left-arm Z rotation) are stored at different temporal positions as illustrated in Fig. 2. Supposing that the maximum length of the captured data and the user queries' does not exceed the 500 frames, we can place each type of motion at different, non-overlapping time intervals.

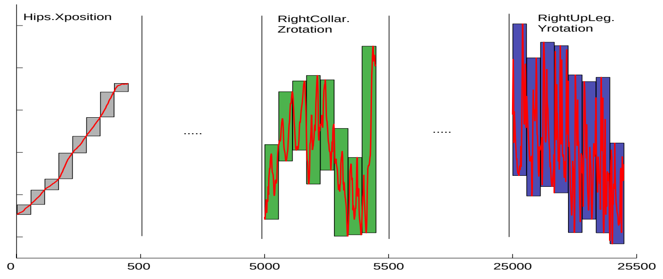

Figure 2: The different types of motions are stored at different temporal positions of the R-tree. This approach allows for efficient low-dimensional indexing.

The user queries that (possibly) contain multiple joint motions, are segmented and subsequently each motion is probed at the corresponding R-tree location. The estimates for each motion are calculated and finally a cumulative estimate over all motions is returned.

[1] video: http://www.cl.cam.ac.uk/users/mpc33/sketch03.html

4 Matching Weights

The storage of different motions at distinct parts of the index facilitates a simple weighting scheme. The ability to interactively select the body-areas utilized in the matching is accomplished by multiplying the distance estimates for each joint by its associated weight value. Important joints, such as the upper leg and knees, are proportionally favored as opposed to joints with less impact such as the foot joint. Unwanted joint motion can also be defined in a similar fashion (Fig 3). This negative matching is achieved by weights inferior to 1.

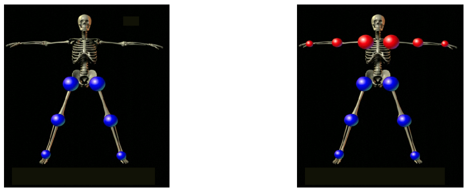

Figure 3: Body Area Weighting. (*Left*) The user can limit the matching to only the legs by assigning weights to individual joints proportionally to their importance. (*Right*) Negative matching areas can be defined so that only motions dissimilar to the query's arm motion, and similar in leg motion, are returned.

5 Subsequence Matching

Until now, only whole sequence matching has been described. Extending our framework to additionally perform subsequence matching is straightforward. This is done by shifting the query MBRs (each of length *m*), by $\lfloor m/2 \rfloor$ time instants and recording the new distance (that is the next position that could potentially have smaller distance). Using the raw data we compute the actual *best-so-far* match. However, we only need to calculate the distance on the raw data, if the lower bound distance is smaller than the *best-so-far*. Therefore, distant subsequences can be efficiently pruned. The procedure continues until the whole sequence is scanned. This is illustrated in Fig. 4.

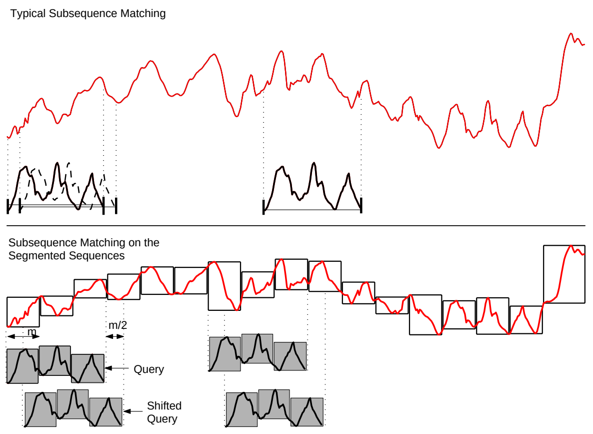
Typical Subsequence Matching
Subsequence Matching on the Segmented Sequences
m
m/2
Query
Shifted Query

Figure 4: Top: Typical Subsequence Matching using a sliding window. Bottom: Using the rectangles to perform matching, we can sufficiently speedup the produce and at the same time avoid trivial matches.

By segmenting the sequences into MBRs, not only do we perform a fast and efficient dimensionality reduction, but we also eliminate the trivial matches that we would have gotten upon operating in the original dimensionality. Trivial matches are the matches returned by sliding a sequence 'left' or 'right' by a small amount, which result to almost identical series.

6 Support for LCSS and DTW

The LCSS is a variation of the edit distance and has been extensively used for matching discrete values. In our formulation, we extend the LCSS model in order to allow for a matching when the sequence values are within a certain range in space and in time. Suppose that we have sequences A and B, where A=(a1,…,an), B=(b1,…,bm) and Head(A) = (a1,…,an-1).

DEFINITION: Given an integer δ and a real number 0<ε<1, we define the $LCSS_{\delta,\varepsilon}(A,B)$.

$$LCSS_{\delta,\varepsilon}(A,B)=\begin{cases} 0 & \text{if A or B empty} \\ 1+LCSS_{\delta,\varepsilon}(Head(A),Head(B)) & \text{if } |a_n-b_n|<\varepsilon \text{ and } |n-m|<\delta \\ \max(LCSS_{\delta,\varepsilon}(Head(A),B), LCSS_{\delta,\varepsilon}(A,Head(B))) & \textit{otherwise} \end{cases}$$

An example of the new LCSS definition is provided in the following figure.

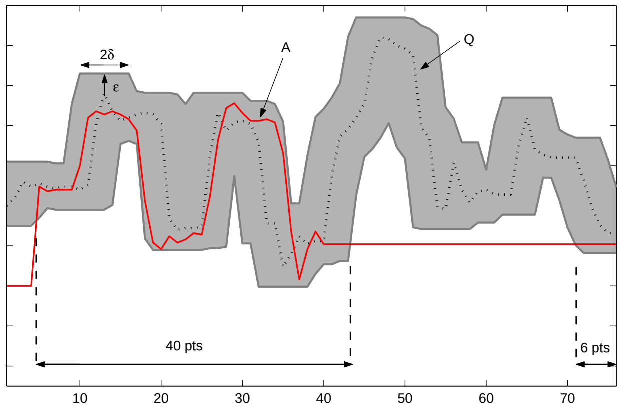

Figure 5 : The notion of LCSS matching within a region of δ in time and ε in space. The gray area defines the Minimum Bounding Envelope (MBE) of a sequence. Anything that lies outside this envelope can never be matched.

The value of the LCSS is unbounded as depends on the length of the compared sequences. We need to normalize it in order to support sequences of variable length. The distance defined from the LCSS similarity is defined as**:**

$$D_{\delta,\varepsilon}(A,B)=1-\frac{LCSS_{\delta,\varepsilon}(A,B)}{\max(\|A\|,\|B\|)}$$

The proposed framework can efficiently support more flexible matches by using the Longest Common Subsequence (LCSS) [5] or Dynamic Time Warping (DTW) models. As opposed to Euclidean distance, DTW and LCSS take into account variations in the time axis. LCSS also has the advantage of being more robust to noise.

It can be shown that if we surround the query by a bounding envelope indicating the areas of possible matching, the lower bound distance computed between this envelope and the indexed MBRs, guarantees no false dismissals [4]. By allowing constrained time warping, we can achieve faster execution time and at the same time avoid distant and degenerate matchings. Fig. 6 shows an example of this approach for the *LCSS* model.

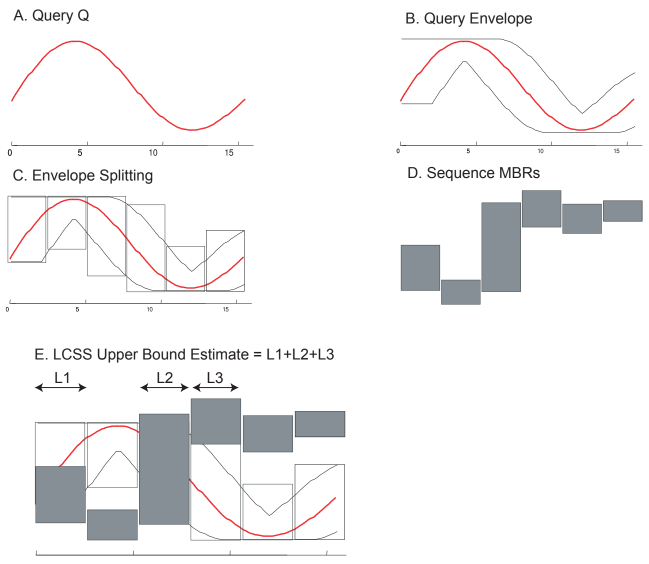

Figure 6: A demonstration of how our framework can support flexible matches. (*A*). User Query, (*B*). Areas of possible matching are surrounded by the Minimum Bounding Envelope (MBE). (*C*). The MBE is segmented into MBRs, (*D*). Indexed MBRs, (*E*). Estimates between the query and the index MBRs are computed.

Since the *LCSS* model describes the similarity, we don't calculate the distance between MBRs, but their degree of overlap. Now our estimates provide an *upper bound* on the actual similarity, and since similarity is inversely analogous to distance, the claim of no false dismissals still holds.

Below we depict an example of the quality of time warped matching for multidimensional sequences.

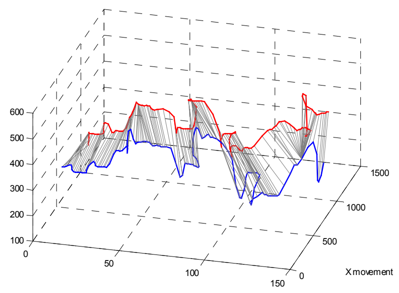

Figure 7: An example of time warping between multidimensional time-series.

7 Further Optimizations

In order to provide better sequence approximation, one can alternatively use a greedy split algorithm to generate the sequence MBRs. The complexity of this operation is now O(nlogn) (for sequences with n points), however the time-series approximation is significantly tighter (Fig 8). Our experimental results indicate that the approximation is very close to the optimal one, which is achieved using dynamic programming and requires a costly quadratic execution time.

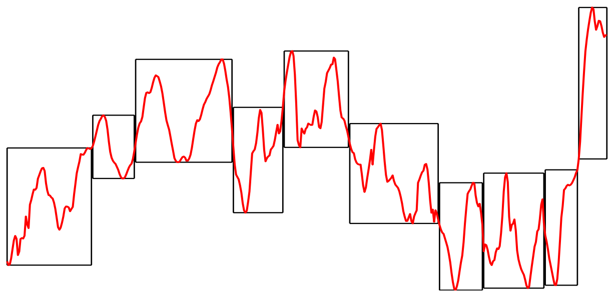

Figure 8: Using a greedy MBR generation approach we can provide a very tight sequence approximation

9 References

[1] Agrawal, R., Faloutsos, C. & Swami, A. (1993). 'Efficient similarity search in sequence databases'. In Proc. *of the 4th Int'l Conference on Foundations of Data Organization and Algorithms*. Chicago, IL, Oct 13-15. pp 69-84

[2] Böhm C., Berchtold S., Keim D.A. (2001) 'Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases', ACM Computing Surveys 33(3): pp 322-373

[3] Guttman A., (1984), 'R-trees: A Dynamic Index Structure for Spatial Searching', In Proc. of ACM SIGMOD International Conf. on Management of Data,

[4] Keogh E., (2002) 'Exact Indexing of Dynamic Time Warping' In Proc. of the *28th International Conference on Very Large Data Bases.* Hong Kong. pp 406-417.

[5] Vlachos M., Kollios G., Gunopulos D (2002). 'Discovering Similar Multidimensional Trajectories'. In Proc. of the 18th International Conference on Data Engineering,