Optimal and Interactive Keyframe Selection for Motion Capture

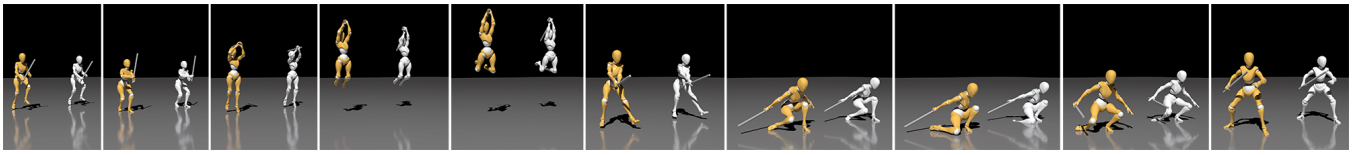
Richard Roberts
Victoria University of Wellington
richard.andrew.roberts@gmail.com

J.P. Lewis
SEED, Electronic Arts and Victoria University of Wellington
noisebrain@gmail.com

Ken Anjyo
OLM Digital and CMIC, Victoria University of Wellington
anjyo@acm.org

Jaewoo Seo
Pinscreen
goongsang@gmail.com

Yeongho Seol
Weta Digital
seolyeongho@gmail.com



Frames extracted from an animation, before (right character in each panel) and after editing (left character) using our technique.

ABSTRACT

Motion capture is increasingly used in games and movies. However, it often requires editing before it can be used. Unfortunately, editing is laborious because of the low-level representation of the data. Existing motion editing methods accomplish modest changes, but larger edits require the artist to "re-animate" the motion by manually selecting a subset of the frames as keyframes. In this paper, we *automatically* find sets of frames that serve as keyframes for editing the motion. We formulate the problem of selecting an *optimal* set of keyframes as a type of shortest-path problem, and solve this problem using efficient dynamic programming. Our algorithm can simplify motion capture to around 10% of the original number of frames while retaining most of its detail. By simplifying animation with our algorithm, we realize a new approach to motion editing and stylization founded on the time-tested keyframe interface.

CCS CONCEPTS

• **Computing methodologies** → **Motion capture**;

KEYWORDS

motion capture, keyframe animation, dynamic programming

ACM Reference Format:
Richard Roberts, J.P. Lewis, Ken Anjyo, Jaewoo Seo, and Yeongho Seol. 2018. Optimal and Interactive Keyframe Selection for Motion Capture. In *SIGGRAPH Asia 2018 Technical Briefs (SA '18 Technical Briefs), December 4–7, 2018, Tokyo, Japan.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3283254.3283256

SA '18 Technical Briefs , December 4–7, 2018, Tokyo, Japan

ACM ISBN 978-1-4503-6062-3/18/12…$15.00
https://doi.org/10.1145/3283254.3283256

1 INTRODUCTION

Motion capture is widely used in games and movies, however it is frequently necessary to edit the mocap for several reasons:

- The motion is "retargeted" to virtual fantasy characters with different proportions, resulting in unrealistic poses.
- Interactions between characters, such as hugging or wrestling, can introduce solving errors due to occlusion.
- The character's recorded motion may not fit their virtual environment, for example, if the height of a door handle does not match the one in the recording studio.
- Especially in the case of games, movements may need to be stylized to feel more rapid, to provide a more responsive experience.
- The director may request other edits for a variety of reasons.

Unfortunately, it is not practical to edit mocap directly, for the same reason that editing a picture by changing individual pixels would be ineffective. A standard approach provided by leading commercial software allows the motion editor to blend changes to a given pose smoothly through the surrounding frames with a spline falloff. This approach is suitable for relatively small and smooth edits, but it raises the question of where to place the control vertices or keyframes on such a spline. Simply placing keys at regular intervals does not provide precise control since the range of influence of each key is not related to the motion. For example, adjusting the motion before a footstep but not afterwards generally requires having a key located at the time of the step. As well, repeated application of this technique risks distorting the character of the motion [Lowe 2016]. In practice, when extensive editing is required motion editors sometimes resort to manually deleting ranges of frames to create an editable representation with a small number of frames that can serve as keyframes [Lam 2017; Shelton 2017].

A number of pioneering motion editing techniques have been introduced in the computer graphics literature. Space-time optimization and sketching interfaces provide alternative and novel interfaces and considerable power. On the other hand, motion editors

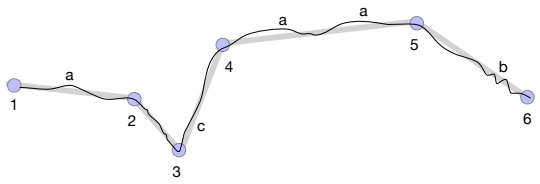


Figure 1: Picking important points on a motion using intuitive principles such as extrema (a) or points of high curvature (b) can easily fail. In this diagram, the curve represents the motion, and points are analogous to keyframes. The points (1)..(6) summarize the curve well, though points (2), (4), (5) are neither points of high curvature nor extrema. Instead of focusing on points individually, our approach considers the error in approximating the motion by *pairs* of points (visualized as the grey line). For any number of points, we find the pairs of points that minimize this error.

and animators are usually trained in traditional keyframe animation, and commercial software has extensive and fluid support for keyframe-based editing. As a result, modern techniques described in the research literature have not yet been widely adopted.

A variety of key point and keyframe selection approaches exist. While it is often assumed that points of high curvature or curvature extrema provide good keypoints for approximating a curve, it is easy to find counterexamples to these simple heuristics (Fig. 1). Greedy algorithms such as variants of the well known Ramer-Douglas-Peucker algorithm [Ramer 1972] often perform well, but generally do not find the best possible solution.

Many existing techniques operate on individual parameter curves (e.g. particular joint rotations) independently. Importantly, a technique suitable for professional use should allow the manipulation of keyframes (KFs), since animation practice generally recommends working in terms of poses when making initial large-scale edits (this is termed *pose-to-pose* animation), followed by editing of individual curves as necessary [Roy 2013; White 2006].

To derive our solution to the keyframe selection problem, we return to the idea of a keyframe. In a keyframe animation, the KFs are a minimal set of poses that can be interpolated to closely approximate the motion of the character or object. In abstract terms, these KFs provide an economical set of parameters that provide complete control over the animation. Given dense mocap data, we thus seek a minimal set of frames that, when interpolated, provide the closest approximation to the entire motion. This is a combinatorial problem as there are $\binom{N}{k}$ potential choices of k KFs that summarize N frames. While this would be intractable, our problem is equivalent to a particular form of the classic shortest path problem and has an efficient dynamic programming algorithm to obtain the optimal solution. In practice we find that optimal solutions can be found in a reasonable time (e.g. less than a second for motion clips of typical length) when implemented on a laptop computer. Our algorithm, *Salient Poses*, provides the optimal solution for *all* numbers of KFs less than or equal to a number requested by the artist at no extra cost, thus allowing the artist to interactively browse solutions with different numbers of KFs and select one that is best for their particular editing purpose.

The resulting keyframe animation closely resembles the original motion but is editable using the standard approaches and polished tools used by industry. The fine-scale detail lost though the keyframe approximation can be saved by subtracting the approximation from the original, and some of this residual can be added on top of the edited animation if desired.

2 METHOD

We discussed our research with seven professional motion editors and animators from three major companies and found both agreement and disagreement about how keyframes are used. From the discussion, we concluded that the qualities of "good" keyframes involve artistic judgement and individual artists do not always agree on some aspects of what constitutes a "good" keyframe. Nevertheless, we found that extremes in the motion tend to be selected as keyframes, and also that there are more keyframes in areas of complex motion.

Considering the discussion from the perspective of approximation, we propose that *keyframes are those frames that best allow the remainder of the motion to be interpolated* (Fig. 1).

To solve our problem, we express it as a particular form of the shortest path problem [Bertsekas 1998], which can be solved with dynamic programming. Refer to our online documentation for a reference implementation.[1]

Each of N frames in the original mocap clip becomes a node in a graph, with directed edges to all temporally subsequent nodes. A node contains the joint positions of the pose that occurs at its corresponding frame, forming a high-dimensional point. The weight of an edge $v_{i,j}$ is the cost of approximating the high-dimensional motion between frames i, j by an approximation that uses nodes i and j as keyframes. In our implementation, we quantify this approximation cost as the maximum perpendicular distance between a linear interpolation of keyframes i and j, and the portion of the motion lying between these two keyframes. Each frame corresponds to a pose that we interpret as a point in high-dimensional space. With this interpretation, the distance measure can be implemented easily using a high dimensional point to line distance.[2]

Our problem differs only slightly from the single source all destinations problem used to motivate the well known Dijkstra algorithm [Dijkstra 1959]: in our problem the nodes have a total, temporal ordering. We seek a single minimum-cost path from the start node (the first frame of the mocap) to the end node (the last frame) that passes through K intermediate nodes while skipping the remaining nodes. K is interactively specified by the artist.

2.1 All-Pairs Table

A table of all $\binom{N}{2}$ edge costs is precomputed.

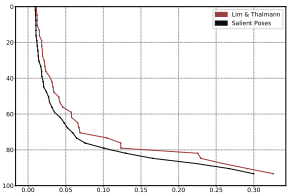
2.2 Successive Keyframe Selections

The error for an optimal approximation of the motion using m keyframes is

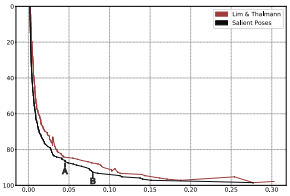
$$E^m_{i,j} = \min_k E^{m-1}_{i,k} + e_{k,j}$$

[1] http://salientposes.com

[2] We experimented with minimizing the integrated squared error between the high-dimensional curve and its approximation. That did not give any clear advantage and so we use the infinity norm for simplicity.



(a) Capoeira, 105 frames



(b) Quarterback Pass, 460 frames

Figure 2: A comparison between our optimal approximation and the greedy-approximation algorithm [Lim and Thalmann 2001]. The vertical axes are the amount of compression, and the horizontal axes depict the distance between the original animation and the motion reconstructed from the keyframes, specifically, the Euclidean distance between corresponding pairs of frames, averaged over both joints and frames, and normalized into meters assuming the character's height is 1.7m. The annotated points A, B in figure (b) are discussed in the text.

with $E^1_{i,j} \equiv e_{i,j}$. In this notation $E^1_{i,j}$ involves two KFs; we always take the first and last frames as KFs. Note that the computation for step m is reused in step $m+1$, thereby resulting in a dynamic programming optimization.

The selected keyframes are identified during this computation as

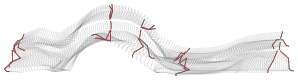
$$\arg\min_k E^{m-1}_{i,k} + e_{k,j}$$

Note that the computation produces keyframe selections for *all* $m \leq K$. This is useful as it allows the artist to interactively browse solutions with various numbers of keyframes and pick one that provides the best trade-off between fidelity and editability.

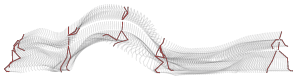
2.3 Run time

The all pairs table has $\binom{N}{2}$ unique entries and hence quadratic cost. The entries can be computed independently and are computed in parallel on the GPU in our implementation. The cost of this step is typically insignificant compared to the time required for common operations such as opening user interface windows. The dynamic programming algorithm runs on the CPU. The cost of each step is approximately quadratic, resulting from the search over k, j, and the overall cost is approximately cubic. The algorithm only needs to proceed as far as K keyframes, however the expected number K generally grows with the length of the mocap sequence, so it is reasonable to summarize the algorithm cost as $O(N^3)$.

In practice, our algorithm provides interactive performance for mocap sequences of typical length. Motion pictures are composed of "shots" that typically last between 4-6 seconds [Kaufman and Simonton 2014, p .126] or often less in action-heavy visual effects sequences. For a shot containing 300 frames, less than one second is required to find all optimal KF solutions of 50% compression or more ($K \geq N/2$) on using a laptop with a low-end current GPU (Intel Iris 550). If the motion is significantly longer than this, it should be split in order to provide interactive performance.

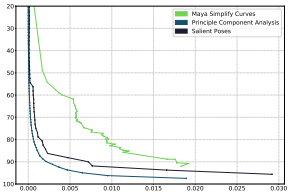


(a) Greedily Selected, 5 Keyframes

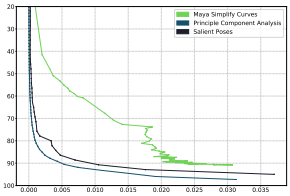


(b) Optimally Selected, 5 Keyframes

Figure 3: These time-lapse renders compare keyframes selected greedily (left) and optimally (right). The keyframes are drawn in red and the in-betweens in semi-transparent grey. An advantage of the optimal approach is that the keyframes are distributed to best approximate the motion in every selection. The distribution is useful for editing, since the control afforded by the keyframes is proportional to significant changes in pose.



(a) Jumping Down, 160 frames



(b) Sword Attack, 140 frames

Figure 4: A comparison between our optimal approximation, our implementation of pose-based PCA, and Maya's "Simplify Curves" algorithm. The axes are the same as those in Fig. 2.

3 RESULTS

The following results were created using animations from Adobe's Mixamo library.[3] The complete set of results can be viewed online.[4]

3.1 Comparison to Greedy Algorithm

Fig. 2 shows the approximation performance of our algorithm versus the algorithm in [Lim and Thalmann 2001]. The performance of [Lim and Thalmann 2001] is sometimes quite good, however our algorithm is always at least as good and often better. Our optimal algorithm is more costly than [Lim and Thalmann 2001], however the cost is not prohibitive (Section 2.3).

Fig. 3 compares the results of the greedy algorithm and our algorithm on selecting KFs from a mocap sequence with 140 frames. The better distribution of KFs provided by our algorithm is evident by inspecting this figure.

3.2 Approximation error vs. compression: Maya, PCA, Ours

Fig. 4 compares the compression (vertical axis) versus average error (horizontal axis) obtained with our algorithm, PCA, and Maya. Note that these curves were obtained by selecting a compression level and plotting the resulting error, thus, the x-coordinate is a function of the y-coordinate rather than the more common $y = f(x)$. We chose this visualization to highlight the amount of compression obtained

[3] https://www.mixamo.com
[4] http://salientposes.com/results

for a particular error. The PCA compression was computed by treating each pose as a datapoint, meaning that each reconstructed frame is a weighted combination of eigenvectors, rather than the alternatives of compressing curves or space-time windows.

In this comparison the simplification performed by Maya does *not* find keyframes, rather it finds keys independently on each curve. The total number of found keys was manually adjusted to match the number of keys in the keyframes (number of keyframes multiplied by the number of degrees of freedom) used by our algorithm. This difference benefits Maya in the comparison, since greater compression can be achieved when choosing keys for each DOF independently (the keys are not organized as KFs across the set of curves). In any case Maya's results are not competitive.

Although our method is not primarily intended for compression, we see that it comes within 5% of this naive application of PCA. This could be of secondary benefit to games that use keyframes for compression, since it eliminates the need to maintain a separate compressed representation.[5] On the other hand if compression is a primary concern then a separate compression algorithm should be employed.

4 DISCUSSION AND CONCLUSION

The representations and approaches preferred by artists are often characterized by a combination of semantically meaningful parameters, high-level control, and an aesthetic appeal that may relate to simplicity and predictability.[6] The keyframe representation of motion is one such representation that has stood the test of time and continues to be used by artists in both 2D and 3D media.

In this paper we introduce a solution to the important problem of editing motion capture, by converting the mocap into a keyframe representation that supports editing using traditional tools and approaches. Our algorithm is designed to satisfy the artists' common preference for keypoints that align on the same frames, i.e. keyframes. The algorithm is both simple and optimal. It produces a range of solutions with differing numbers of keyframes, allowing the artist to intuitively browse the solutions and pick one that offers the desired trade-off between detail and control.

In terms of evaluation, given the same choice of error measure our optimal solution outperforms competing greedy algorithms by definition. It also outperforms a leading commercial tool, even with the handicap that the commercial tool produces a similar number of keypoints without grouping them into keyframes.

We have shown the tool to artists at three internationally known companies from the video game, cartoon animation, and visual effects industries. An initial version of our program was described as "a life changing tool", another artist commented that it makes it "a lot easier to visualize the movement as a series of poses", and another said that it would be ideal for stylizing motion capture for use in action-focused games. More generally, the artists felt that our algorithm reduces the time and cost required for motion editing and stylization.

4.1 Limitations

Our algorithm has several evident limitations.

Asymptotic complexity. As mentioned previously, very long motion capture clips may need to be split if interactive performance is desired.

Sweet spots. The approximation error does not always decay smoothly as more KFs are added. On the contrary, there are error "cliffs" where the addition of a single KF produces a noticeable reduction in error, followed sometimes by "plateaus" where a few additional KFs provide little improvement (points A,B in Fig. 2 (b)). Currently it is up to the artist to interactively browse the solutions for various K and select the best one according to their judgement.

4.2 Future Work

While we feel that our algorithm provides a good general solution for converting motion capture into an editable representation, there remains much work to be done to more fully formalize and support the motion editor's craft.

Animation concepts. A challenging open problem is to more fully express animation concepts such as "leading part" [Lasseter 1987] in a keyframe-based editing framework. We speculate that this challenging problem will, at least, require re-thinking the holistic one-fits-all approximation error we have employed in this work.

Spline fitting. Another topic for future work is that of spline fitting. While spline fitting is often considered to be a solved problem, our experience with published fitting algorithms reveals that clear improvements can often be obtained with manual adjustments to the curve tangents.

ACKNOWLEDGMENTS

Many researchers and artists have contributed important insights to this research. The authors would like to call special thanks to Ayumi Kimura and other staff of OLM Digital, to Johan Andersson, Ida Winterhaven, and Binh Le of SEED, Electronic Arts, and also to Ian Loh and other staff of Victoria University of Wellington's Computational Media Innovation Centre and Virtual World's Lab.

REFERENCES

Dimitri P. Bertsekas. 1998. *Network Optimization: Continuous and Discrete Models.* Athena Scientific.

E. W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1, 1 (1959), 269–271.

J.C. Kaufman and D.K. Simonton. 2014. *The Social Science of Cinema.* Oxford U. Press.

David Lam. 2017. *Personal communication.* Electronic Arts.

John Lasseter. 1987. Principles of Traditional Animation Applied to 3D Computer Animation. *SIGGRAPH* 21, 4 (1987), 35–44.

Ik Soo Lim and D. Thalmann. 2001. Key-posture extraction out of human motion data. In *Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE*, Vol. 2. 1167–1169.

Dan Lowe. 2016. Animation Bootcamp: The 'Animate' Button. (2016). Game Developers Conference.

Urs Ramer. 1972. An Iterative Procedure for the Polygonal Approximation of Plane Curves. *Computer Graphics and Image Processing* 1, 3 (1972), 244 – 256.

Kenny Roy. 2013. *How to Cheat in Maya 2014 : Tools and Techniques for Character Animation.* Focal Press, Burlington, MA.

Damon Shelton. 2017. *Personal communication.* Electronic Arts.

Tony White. 2006. *Animation from Pencils to Pixels : Classical Techniques for Digital Animators.* Focal Press, Burlington, MA Oxford.

[5] In games it is generally desirable or necessary to compress the motion, since the expected movement sets of all characters must either be stored in limited GPU memory (along with the geometry and textures of all objects in the scene), or the motion of every character must be streamed to the GPU. In either case the resource (memory or bandwidth) is in demand and must be optimized.

[6] For example, meshes that have a desirable distribution of polygons are referred to as having "edgeflow".