A Magic Wand for Motion Capture Editing and Edit Propagation

Christopher J. Dean
Victoria University of Wellington
christopherjosephdean@gmail.com

J.P. Lewis
SEED, Electronic Arts
noisebrain@gmail.com

Andrew Chalmers
Victoria University of Wellington
andrew.chalmers@ecs.vuw.ac.nz

ABSTRACT

This paper introduces a new method for editing character animation, by using a data-driven pose distance as a falloff to interpolate edited poses seamlessly into a motion sequence. This pose distance is defined using Green's function of the pose space Laplacian. The resulting falloff shape and timing are derived from and reflect the motion itself, replacing the need for a manually adjusted falloff spline. This data-driven region of influence is somewhat analogous to the difference between a generic 2D spline and the "magic wand" selection in an image editor, but applied to the animation domain. It also supports powerful non-local edit propagation in which edits are applied to all similar poses in the entire animation sequence.

CCS CONCEPTS

• **Computing methodologies → Animation**;

KEYWORDS

Character Animation, Motion Editing, Motion Retargeting

ACM Reference Format:
Christopher J. Dean, J.P. Lewis, and Andrew Chalmers. 2018. A Magic Wand for Motion Capture Editing and Edit Propagation. 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/10.1145/3283254.3283268

1 INTRODUCTION

Motion capture (mocap) performances provide detailed and realistic animations, but both editing of motion capture and custom animation are routinely required for retargeting, adjusting motion to new virtual environments, and other reasons. A typical skeletal rig may have 150 degrees of freedom (or more) that need to be specified across hundreds of frames. If changes are needed, editing these animation sequences is time consuming.

Many existing production techniques rely on spline interpolation of corrections made by the artist across the timeline. These approaches have two disadvantages: First, the appropriate temporal region of influence of a spline-based edit differs greatly for different joints. Thus the splines need to be manually defined on a case-by-case basis—a cumbersome task for the artist. More fundamentally, splines always have the same generic smooth shape, and may not appear natural for realistic motion. Research in motion editing is often founded on space-time optimization approaches. In general, these methods are similarly *data agnostic* – indeed, an optimization minimizing an integrated squared derivative is just an underlying formulation of a spline.

Instead of relying on splines, we propose to let the data "speak for itself" by basing the falloff directly on the change of pose of the underlying motion. A data-driven falloff should have these characteristics:

- It should be proportional to the change in pose and correspond to distance on the manifold of movement.
- it should be smooth if and only if the change in pose is smooth.
- It should be reasonably efficient to compute, thereby allowing interactive edits.

Unfortunately, defining a distance on poses has well known difficulties: Euclidean distances are not appropriate due to the rotational nature of an articulated body, while on the other hand, distances founded on concatenated rotational degrees of freedom (dofs) have the problem that major joints such as the shoulder have a much larger influence than the distal joint of a finger. This problem has been addressed by weighting each dof differently, though (depending on the underlying representation of rotations) these weights may need to be recomputed at each pose.

In this paper we instead start from the idea that the Laplacian reflects the geometry of a signal or manifold [Levy 2006]. With this in mind, we show that the Green's function of the Laplacian can be used to provide a natural similarity measure between poses.

Our data-driven falloff is somewhat analogous to the difference between a generic spline and the intelligent region selection available in popular image editing software such as Photoshop (e.g. algorithms such as [Gleicher 1995; Mortensen and Barrett 1995]), but applied to the animation domain. The result is intelligent interpolation that blends new poses into the existing motion. For cyclical movement, the data-driven falloff can be applied across all similar poses (with strength depending on their relative similarity), providing non-local edit propagation. This is particularly powerful since human and animal motion often involves cyclical gaits.

2 RELATED WORK

Research on motion editing spans several decades, but most of this work is data agnostic and thus not directly related to our approach. [Grochow et al. 2004] pioneered research on data-aware motion editing, drawing on a manifold learning technique. In Section 4 we compare our work to classic [Tenenbaum et al. 2000] and more recent [Coifman and Lafon 2006] manifold learning methods.

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

ACM ISBN 978-1-4503-6062-3/18/12…$15.00
https://doi.org/10.1145/3283254.3283268

3 METHOD

Consider a linear differential equation $\Delta u = c$ where c expresses some "forcing function" or constraints, and Δ is a differential operator (the Laplacian in our case). The Green's function solution of the differential equation expresses the unknown solution u as a weighted sum or integral

$$u(\mathbf{x}) = \int_M G(\mathbf{x}, \mathbf{x}')c(\mathbf{x}')dM$$

of the forcing function, where the *Green's function* $G(\mathbf{x}, \mathbf{x}')$ between points $\mathbf{x}, \mathbf{x}'$ serves as the weights, that is, it expresses how much the solution at $\mathbf{x}$ is influenced by the forcing function at location $\mathbf{x}'$. In our application, a sparse set of artist edits play the role of the forcing function, and $G(\mathbf{x}, \mathbf{x}')$ will serve as weights to intelligently propagate those edits across the motion.

Body poses are represented as a vector containing the concatenated rotational degrees of freedom of the skeleton. Our approach is inspired by the Lie algebra/exponential map idea (see [Anjyo and Ochiai 2017; Tournier et al. 2009] for details) – the changes to pose resulting from artist edits are represented using log quaternions and results are converted back to drive the final poses using the exponential map [Grassia 1998].

3.1 The Pose Laplacian

Following [Belkin and Niyogi 2002], we use the graph Laplacian as an approximation to the Laplace-Beltrami operator. We define a pose adjacency matrix using a kernel

$$\mathbf{K}_{i,j} = \exp\left(-\frac{d(\mathbf{p}_i, \mathbf{p}_j)^2}{2\sigma^2}\right) \tag{1}$$

where $\mathbf{p}_i, \mathbf{p}_j$ are pose vectors that describe a skeletal configuration (e.g. joint angles). The underlying distance $d(\mathbf{p}_i, \mathbf{p}_j)$ between pose vectors should reflect the editing purpose, for example, it may be desirable to include the temporal distance between poses. On the other hand, it is only important that this distance be *locally* representative. The width σ should be adjusted so that all large distances are near zero under the kernel; these will be completed by the Green's function (below).

To create the Laplacian, let $\mathbf{r}$ be a vector of the row sums of $\mathbf{K}$. With $\mathbf{D} = \text{diag}(\mathbf{r})$ the normalized adjacency matrix $\mathbf{A}$ is

$$\mathbf{A} = \mathbf{D}^{-1/2}\mathbf{K}\mathbf{D}^{-1/2}, \tag{2}$$

This results in the Laplacian $\mathbf{L} = \mathbf{D}_A - \mathbf{A}$ where $\mathbf{D}_A$ is the corresponding degree matrix (identity). Next we compute the eigenvalues and eigenvectors λ_k, ϕ_k of $\mathbf{L}$. This computation occurs only once and requires a fraction of a second for a motion sequence of the typical length of several hundred frames.

3.2 Green's function

The Green's function similarity between poses $\mathbf{p}_i$ and $\mathbf{p}_j$ is obtained in terms of the eigen-decomposition of the Laplacian

$$G(\mathbf{p}_i, \mathbf{p}_j) = \sum_{k>0} \frac{\phi_k(\mathbf{p}_i)\phi_k(\mathbf{p}_j)}{\lambda_k}. \tag{3}$$

We precompute the matrix $\mathbf{P}$ of all pose similarities $\mathbf{P}_{ij} = G(\mathbf{p}_i, \mathbf{p}_j)$. Typically only a few terms of the sum (3) are needed since it is dominated by the smaller eigenvalues. While the kernel (1) also provides a similarity measure (indeed, the Green's function matrix $\mathbf{P}$ is a straightforward modification of the eigenvalues of the $\mathbf{K}$), the Green's function has a natural interpretation as the effect of a change at pose $\mathbf{p}_i$ on pose $\mathbf{p}_j$, which is directly related to our purpose.

3.3 Edited Pose Interpolation

Here we illustrate a simple method for applying the Green's function distance from equation (3) as an animation falloff. The artist has edited frame i, changing the pose from $\mathbf{p}_i$ to $\mathbf{p}'_i$. We wish to propagate this change to the surrounding motion. The difference between these poses represented in the log map as the delta-pose

$$\delta\mathbf{p}_i = \log(\mathbf{p}'_i) - \log(\mathbf{p}_i). \tag{4}$$

Let $\mathbf{w}$ denote a copy of the ith row of $\mathbf{P}$ that has been scaled to lie between 0 and 1 across the edited region of the timeline. That is, the normalized pose distance must be 0 at the edited pose and rise to 1 at the most dissimilar pose in the region to be edited (Fig. 1).

For all frames j in the editing sequence, the Green's function-weighted edit is then computed as

$$\mathbf{p}'_j = \exp\left(\log(\mathbf{p}_j) + \mathbf{w}_{ij}\,\delta\mathbf{p}_i\right). \tag{5}$$

At $\mathbf{p}_i$, the original location, the weight is maximum and fully activates the delta-pose. Further down the timeline, the weight tapers off, eventually deactivating the edit. Note that the weight can *increase* after decaying to zero, particularly when the motion is quasi-periodic. The artist selects the region of the motion to edit, and they can choose to include such repetition or discard it and focus only on a single movement.

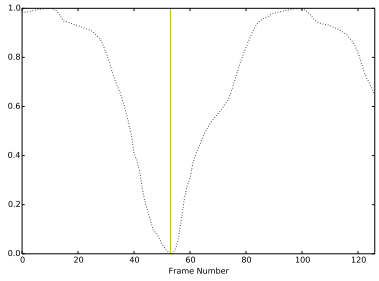


Figure 1: Green's function distance from frame 53 in a walk cycle to all frames in the timeline. We see that the pose distance is zero at the current frame *(yellow bar)*, as the pose is compared to itself. Please enlarge figures to see details.

4 RESULTS

We compared the Green's function distance to two other manifold learning techniques: ISOMAP [Tenenbaum et al. 2000] and diffusion maps [Coifman and Lafon 2006]. Diffusion maps have not been historically applied to animation, but yield results of comparable quality to MDS and ISOMAP in our experiments. In these experiments, we process mocap data from [CMU 2016]. For illustration, we have chosen animated run cycles due to their comprehensible and periodic nature.

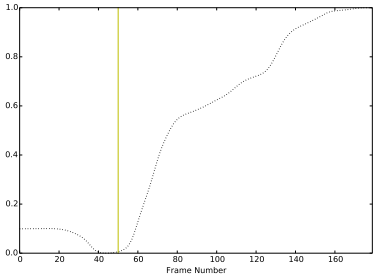


Figure 2: Green's function pose distance for a different animation. In this animation, a character is turning around mid-walk. It is non-periodic, so there are multiple phases to the action (reflected by bends in the curve): walk, stop, turn, and walk.

4.1 Pose Distance

Using the normalized pose distance calculated from Equation 3, Fig. 1 shows the distance from a control pose to every other pose in a 130-frame animation. This is periodic motion, as an actor steps right-left-right-left and so on. Small dents, bends, and plateaus indicate the prominent characteristics of the skeletal motion.

In Fig. 2, we see the pose distances for a non-periodic motion. This actor abruptly turns around in the middle of a walk. There appear to be points of transition in the motion, but the beginning and end of a falloff is not well defined from this distance measure. Inflection points and bends serve as good guidelines for falloff boundaries, but an artist may want customized control over the falloff scaling for blending their edit into this motion.

We compared our results with those obtained with two alternate algorithms, ISOMAP and diffusion maps. In these manifold learning methods, poses lie in an embedded space in which the L_2 distance approximates distance on the manifold of movement. Fig. 3a depicts an overlaid comparison of all three distances. Overall, the three methods detect the same animation characteristics and reveal periodic motion behaviors across the entire timeline.

Fig. 3b takes a closer look at the distance falloff near the edited pose. Zooming in, the local pose information is very rounded for Green's function, somewhat tapered in the diffusion map result, while ISOMAP produces a sharp corner. The sharp ISOMAP falloff results in an undesirable quick "snap" into and out of the edited pose. This effect is noted in the accompanying video results. A smooth falloff causes the animation to ease gently through the targeted pose, demonstrating the effectiveness of Green's function weights for animation editing.

4.2 Edited Animation

Fig. 4 demonstrates the edit propagation in practice. For repetitive actions, a single edited pose and falloff animation can automatically propagate to all instances of similar poses in the timeline. In Fig. 4, the artist wishes for the character to lift their knees while running. With one simple edit, the entire run-cycle is changed.

The efficacy of the animated falloff can be observed in the accompanying results video. As seen in Fig. 3b, pose edits using the Green's function distance are smoothly incorporated and preserve the character of the motion. However, large edits or physics-altering edits introduce inaccuracies in the animation. For example, editing a foot that is planted on the ground will produce the infamous 'skating foot' if not treated as a constrained problem.

As seen in Fig. 5, the Green's function distance changes significantly on a global scale for different values of the kernel width σ. Local information near the edited frame is more stable, but a slight offset of 2-3 frames is sometimes introduced for bad values. This parameter should be adjusted by the artist for desired results.

In summary, our approach requires the artist to adjust only σ and the desired beginning and end of the edit range – and the latter decision is informed by inspecting the pose distance curves. Thus our approach requires much less work than manually editing a number of spline curves, while simultaneously being "data aware".

4.3 Evaluation

Fig. 6 validates that for a simple low-dimensional case, our method produces similar results to spline animation. We look at the vertical movement channel of the ankle in a walk cycle. First, five keyframes were used to create the motion on a foot using IK. The animation was "baked" into a mocap-like format with 40 frames. Next, each of the two animations were edited, forcing the character to lift her feet higher at one frame. Fig. 6 shows the resulting four motion graphs: the original keyframe animation, the baked 'mocap' version of the animation, the keyframe animation with an edit to raise the foot, and the Green's function falloff-edited motion.

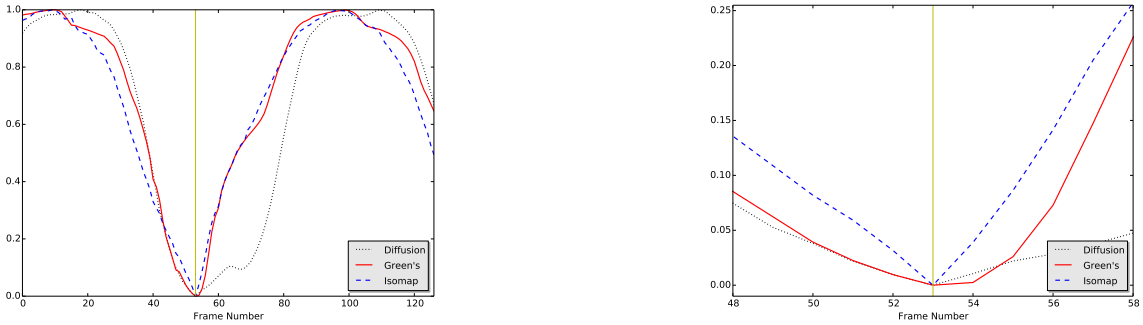
Unsurprisingly, the original animations are nearly identical, with some minor detail loss attributed to the baking from an intricate IK rig to a simplified FK rig. In this simple case, the spline falloff matches the Green's function falloff in both shape and timing extent. On the other hand, in the case of more complex motion the Green's function falloff curves are not "generic" and have interesting detail that reflects the underlying motion (Figs. 1,2).

5 SUMMARY

In this research we have introduced the Green's function similarity for animation editing. It can be thought of as a data-driven spline, designed to operate in the animator's usual workflow. We used the Green's similarity as a natural falloff for propagating edits to the motion. This general idea slightly resembles label propagation in semi-supervised machine learning [Zhu et al. 2003]. Until now, this problem has been either solved tediously by hand with splines, or automated using space-time optimization-based approaches. Both approaches produce generic smooth/low energy falloff curves. Our method proved to yield better falloff curves than diffusion maps and ISOMAP. Comparisons against spline editing in a simple case show that it behaves predictably, while generalizing to nuanced data-driven falloffs in more complex cases. Please see the authors' websites for additional supplementary material.

ACKNOWLEDGMENTS

JPL acknowledges helpful discussions with Ken Anjyo, Hiroyuki Ochiai, Eitan Grinspun, and Mathieu Desbrun.



(a) (b)

Figure 3: Comparison of Green's distance vs. diffusion distance (10 iterations) vs. ISOMAP distance for frame 53 in a run cycle. Fig. 3b is a zoomed-in comparison of Fig. 3a near the control pose. Note the smoothness of the Green's function at this point.

REFERENCES

Ken Anjyo and Hiroyuki Ochiai. 2017. *Mathematical Basics of Motion and Deformation in Computer Graphics* (2 ed.). Morgan & Claypool.

Mikhail Belkin and Partha Niyogi. 2002. Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. In *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani (Eds.). MIT Press, 585–591.

CMU 2016. CMU Graphics Lab Motion Capture Database. http://mocap.cs.cmu.edu, Accessed 7 January.

Ronald R. Coifman and StÃ©phane Lafon. 2006. Diffusion maps. *Applied and Computational Harmonic Analysis* 21, 1 (2006), 5 – 30.

Michael Gleicher. 1995. Image Snapping. In *Proc. 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. ACM, 183–190.

F. S. Grassia. 1998. Practical parameterization of rotations using the exponential map.. In *Journal of graphics tools, 3(3), 29-48.*

Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popovic. 2004. Style-based inverse kinematics. *ACM Trans. Graph.* 23, 3 (2004), 522–531.

B. Levy. 2006. Laplace-Beltrami eigenfunctions: towards an algorithm that understands geometry. *IEEE Int. Conference Shape Modeling and Applications (SMI)* (2006).

Eric N. Mortensen and William A. Barrett. 1995. Intelligent Scissors for Image Composition. In *Proc. 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. ACM, 191–198.

J. B. Tenenbaum, V. de Silva, and J. C. Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science* 290 (2000), 2319–2323.

M. Tournier, X. Wu, N. Courty, E. Arnaud, and L. Reveret. 2009. Motion compression using principal geodesics analysis. In *Computer Graphics Forum 28(2)*. 355–364.

Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. 2003. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *Int. Conf. Machine Learning*. 912–919.

Figure 4: An artist edits the pose *only* at frame 20 *(left)*, lifting the knees higher. A propagated edit produces a near-identical pose adjustment in frame 110 *(right)*, which is a similar pose in the run-cycle. Character model courtesy of ©copyright Blender Foundation | www.sintel.org.

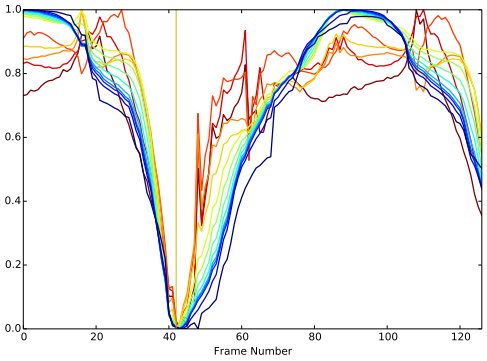


Figure 5: The Green's function distance from a control pose over several different sigma values (lower values are more blue). The range of choices allows the artist to highlight different motion features while still respecting the original character of the data.

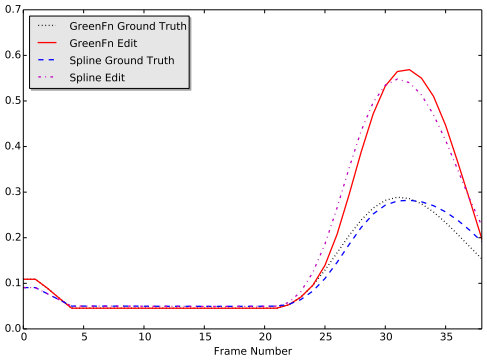


Figure 6: In a simple case, our algorithm's falloff curves resemble generic splines. This is a foot's z motion channel in a keyframe-animated walk-cycle. The animation was edited to raise the foot. A similar edit was made using our method, after converting to a mocap-like dense set of keys.