A Multi-resolution Approach for Adapting Close Character Interaction

Edmond S. L. Ho*
Department of Computer Science
Hong Kong Baptist University

He Wang
School of Informatics
University of Edinburgh

Taku Komura
School of Informatics
University of Edinburgh

Abstract

Synthesizing close interactions such as dancing and fighting between characters is a challenging problem in computer animation. While encouraging results are presented in [Ho et al. 2010], the high computation cost makes the method unsuitable for interactive motion editing and synthesis. In this paper, we propose an efficient multiresolution approach in the temporal domain for editing and adapting close character interactions based on the Interaction Mesh framework. In particular, we divide the original large spacetime optimization problem into multiple smaller problems such that the user can observe the adapted motion while playing-back the movements during run-time. Our approach is highly parallelizable, and achieves high performance by making use of multi-core architectures. The method can be applied to a wide range of applications including motion editing systems for animators and motion retargeting systems for humanoid robots.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation I.3.8 [Computer Graphics]: Applications—;

Keywords: Character animation, close interaction, spacetime constraints

1 Introduction

Close interactions of characters such as dancing and fighting are widely seen in computer animations, interactive games, movies and TV commercials. However, automatic synthesis of such motions is not an easy task as the close contacts between the body parts can easily result in collisions and interpenetrations. Imagine a situation where the body parts of the characters are tangled with one another such as dancing. Collisions and penetrations can easily happen. Since such kind of artifacts strongly affect the realism of the animation, a huge amount of computation is spent on detecting and handling the collisions to alleviate this problem. As a result, designing and creating such kind of interactions still require a lot of manual work by experienced animators.

While recently proposed methods based on the spatial relationships between the characters and objects in the environment [Ho et al. 2010] have shown a promising direction for solving this problem, the computational cost required makes it difficult to be applied to interactive applications. The problem lies in the design of the spacetime optimization in the Interaction Mesh [Ho et al. 2010] motion adaptation framework. In [Ho et al. 2010], all frames of the motion are edited at once by solving a spacetime optimization problem. The advantage of such an approach is the smoothness of the produced motions as we can take into account the influences among different frames when solving them simultaneously. However, the drawback of this method is that solving such a large problem will usually result in slow convergence, especially when the number of variables and constraints is large. As a result, a huge amount of computation is required.

In this research, we accelerate the performance of the method proposed in [Ho et al. 2010] by introducing the idea of multiresolution optimization in the temporal domain. We first produce a multiresolution structure of the motion by extracting keyframes from the original motion. A coarser version of the motion is first computed by solving a smaller scale spacetime optimization by optimizing the keyframes. Next, starting from the motion computed at the coarser level, the fine details of the motion between the keyframes are adjusted by solving another series of smaller scale spacetime optimization problems. Our multiresolution approach improves the convergence of the optimization problem and thus speeds up the motion adaptation process. In addition, the highly parallelizable nature of our approach enables the motion to be edited on-the-fly interactively when playing back the animation.

Our approach has many advantages compared to existing motion editing techniques that can handle close interactions. Using our method, the user can interactively change the sizes of the characters and also edit the postures during close interactions, which makes the method suitable for animators to edit and produce animations. As we compute the movements by solving linear problems, we can easily impose constraints such as ZMP and physical constraints. As a result, our method can be applied to interactive applications such as motion editing and online control of humanoid robots.

1.1 Contributions

The contributions in this paper are summarized as follows:

- We propose a multiresolution approach which can be applied to the Interaction Mesh framework to synthesize close interactions between characters efficiently while satisfying linear constraints.
- The proposed method is highly parallelizable and can be used on-the-fly when playing back animations. By this, *near-realtime* performance can be achieved.

2 Related Work

In this section we review the related work in synthesizing character animations by spacetime optimization and its extensions.

Spacetime constraints [Witkin and Kass 1988] is a method to synthesize realistic animation by spatiotemporally optimizing the motion subject to constraints based on body positions, kinematics and dynamics. It solves the problem of high frequency movements which can appear when per-frame motion editing methods (e.g. Inverse Kinematics (IK)) are used. Rose et al. [Rose et al. 1996] create transitions between human motion segments by spacetime

*Email: edmond@comp.hkbu.edu.hk

constraints. Gleicher [Gleicher 1997] applies spacetime optimization and motion displacement map [Bruderlin and Williams 1995; Witkin and Popovic 1995] to optimize the motion to follow the target trajectories. Using motion displacement map allows a relatively larger interval between keyframes. As a result, the number of variables to be solved is reduced and the performance can be improved. The approach has also been applied to motion retargeting [Gleicher 1998], which is to use the captured motions for characters with identical structure but in different body sizes.

Liu et al.[Liu et al. 2006] simulate realistic interactions of two characters such as one character avoiding another and a parent pulling the hand of the child by iteratively applying such spacetime optimization to each character. Ho et al. [Ho et al. 2010] propose to use Interaction Mesh to represent the spatial relationships between the body segments of the characters. By maintaining the shape of the Interaction Mesh while editing and retargetting the motions by spacetime constraints, the context of the close interaction can be preserved.

Physically plausible motions can also be generated by using an appropriate objective function and constraints in spacetime optimization. Popović and Witkin [Popović and Witkin 1999] apply spacetime constraints to modify the stepping pattern in captured motions as well as generating various kinds of jumping motions. To reduce the computational cost, the spacetime optimization is performed on a simplified character and the result is finally retargetted to the original character using frame-based IK. However, it is not suitable to apply such an approach for handling close interactions as the collisions of the body parts resolved by IK can result in jerky movements. Liu and Popović [Liu and Popović 2002] propose a method to convert a simple animation into a physically valid one by enforcing dynamic constraints such as linear and angular momentum constraints in spacetime optimization.

Although previous approaches about spacetime optimization show a wide variety of high quality movements can be synthesized by spacetime constraints, the high computational costs make it difficult to be applied to interactive or realtime applications. O'Brien et al. [O'Brien et al. 2011] represents the body by a series of vertices instead of segments to accelerate the optimization process. We also use such a representation to improve the performance.

The concept of multiresolution motion editing is explored in previous works such as [Liu et al. 1994] and [Lee and Shin 1999]. [Liu et al. 1994] propose to use hierarchical wavelet basis functions to represent each DOF to reduce the number of discrete variable when solving spacetime optimization. We share the same interest as in [Liu et al. 1994] to edit the motion using coarse-to-fine motion editing technique. Lee and Shin [Lee and Shin 1999] synthesizes new motions by combining the curves which are representing the joint angles of the character at different resolutions. Since the constraints (such as positional constraints) are solved at each frame individually, handling inter-frame constraints requires additional work. On the other hand, our method is based on the original spacetime constraints. The performance is improved by re-arranging and dividing the original large scale problem into multiple smaller problems. As a result, our approach can easily fit into the existing spacetime constraints framework.

The work most related to ours is the one by Al-ashqar et al. [Al-Asqhar et al. 2013]. They accelerate the motion adaptation process by projecting the configuration to the null space of each constraints iteratively, whose convergence is much faster [Shi et al. 2007] than simultaneously imposing all the constraints. Using their approach, motions with close interactions can be retargeted to different characters at interactive framerate. One issue is that they do not consider physical constraints and therefore the movements do not satisfy physical laws. Also, the iterative approach is rather difficult to be parallelized.

In summary, there is a wide range of previous work in spacetime constraints and various techniques are introduced to improve the performance. However, existing methods except either cannot handle close interactions well, or even they do, their convergences is either slow (such as [Ho et al. 2010]) or have issues handling physical constraints (such as [Al-Asqhar et al. 2013]) which can be important for some applications.

In this paper, we propose a multiresolution approach that is highly parallelizable to accelerate the optimization process.

3 Overview

The overview of the proposed method is shown in Figure 1. Given the reference motion, the proposed method retargets the motion to the characters with new body sizes. Our method is divided into two stages, the *pre-processing* and *run-time* stages. In the pre-processing stage, the spatial relationships of the characters are computed (see Section 4.1). Next, the motion is converted into a multiresolution structure (see Section 5) by adaptively extracting keyframes (see Section 5.1) from the reference motion. During run-time, the postures in the keyframes are edited by spacetime constraints. The postures in the middle frames are first estimated by interpolating the adapted keyframes and finally edited by applying spacetime optimization in a short window (see Section 5.2). Further discussion on applying parallel computing to our method is presented in Section 5.3.

We will first briefly explain the Interaction Mesh motion adaptation framework [Ho et al. 2010] and how we use it to synthesize close interactions in Section 3. The Interaction Mesh framework can be used as a frame-based motion editing process and is being used for editing the selected keyframes in our proposed multiresolution model (see Section 5).

4 Interaction Mesh Motion Adaptation Framework

Interaction Mesh [Ho et al. 2010] is a volumetric mesh designed for encoding the spatial relationships between the body parts of the characters and objects that are closely interacting with each other. In particular, a motion adaptation framework has been proposed in [Ho et al. 2010] for retargetting the closely interacting characters into different body sizes while maintaining the context of the interaction. In this section, we will explain the construction of the Interaction Mesh from motion data (Section 4.1) and how the motion is edited by solving a constrained optimization problem (Section 4.2).

4.1 Computing the Interaction Mesh

As in [Ho et al. 2010], an Interaction Mesh is computed from the reference motion at every frame to extract the spatial relationship of the characters in the preprocessing stage. Interaction Mesh is a volumetric mesh composed of vertices and edges. The vertices are the locations of joints of the characters. The edges are computed by Delaunay tetrahedralization [Si and Gaertner 2005] of the point cloud (i.e. the vertices). Here, we apply the Delaunay tetrahedralization to the point cloud that contains the locations of the joints of the two characters.

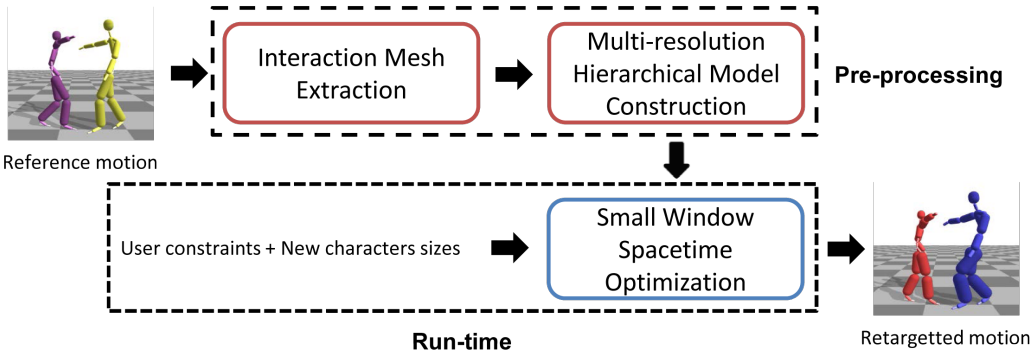


Figure 1: *The overview of our proposed method.*

4.2 Constrained Optimization

In order to preserve the context of the motion, we preserve the geometric details of the Interaction Mesh while editing the postures. More specifically, the distortion of the Interaction Mesh is minimized subject to various constraints during the motion adaptation process. The optimization can be applied per frame to edit each posture or to the entire motion to synthesize a smooth motion. The energy functions and constraints used in the proposed method are presented below.

Notations: Let m be the number of vertices in the Interaction Mesh, $p_j^i (1 \leq j \leq m)$ be the vertices at frame i, V_i be a vector of size $3m$ that includes all p_j^i such that $V_i = (p_1^{i\intercal}, \cdots, p_m^{i\intercal})$, and $\dot{p_j^i}$ and $\dot{V_i}$ be their derivatives.

4.2.1 Energy Functions

As in [Ho et al. 2010], we use the following energy functions in our method, namely *deformation energy* $E_L(V_i')$, *acceleration energy* $E_A(V_i', V_{i-1}')$ and *constraint energy* $E_C(V_i')$. Using the deformation energy can minimize the distortion of the Interaction Mesh. As a result, the spatial relationships of the characters can be preserved. In order to maintain the smoothness and continuity of the motion, the acceleration energy is minimized.

4.2.2 Constraints

Here we explain the constraints enforced in the spactime optimization. In particular, we use the bone-length constraints $C_B(\dot{V_i})$, morphing constraints $C_M(\dot{V_i})$ and collision constraints $C_C(\dot{V_i})$ proposed in [Ho et al. 2010], anti-foot-sliding constraints $C_F(\dot{V_i})$ proposed in [Ho et al. 2013] and balancing constraints $C_{CoP}(\dot{V_i})$ proposed in [Ho and Shum 2013]. The details of each constraint are explained below.

Bone-length, morphing and collision constraints: Bone-length constraints are used for maintaining the distance between adjacent joints of characters as the body segments are rigid. The poses of the characters are gradually morphed to the target sizes using morphing constraints for motion retargetting. In order to prevent penetrations between the bounding volumes of the body segments, collision constraints are enforced and the colliding body segments will be moved apart.

Anti foot-sliding constraints: Visual artifacts such as foot-sliding can be produced if the motion is linearly interpolated. In order to cope with this, we constrain the foot trajectories as follows:

$$C_F(\dot{V^f}_i) = \dot{V^f}_{ref,i} - \dot{V^f}_i \quad (1)$$

where $\dot{V^f}_{ref,i}$ is a vector that contains the positions of the feet joints in the reference motion, and $\dot{V^f}_i$ are the elements of V_i that represent the position of the feet joints.Adapting motions sometimes requires re-planning of the foot plants to preserve the naturalness of the motions. However, this contact planning problem is another open problem which is out of the scope of this paper. In addition, based on our observations, the requirement of big changes of foot plants is rare when changing the proportions of body parts. Therefore, we assume that no significant changes are needed for adapting the motions as in [Ho et al. 2010].

Balancing Constraints: Now we explain how physical constraints can be enforced in the optimization as in [Ho and Shum 2013]. One of the advantages of using spacetime optimization over frame-based motion editing approaches is that physical constraints can be enforced to produce physically feasible motions. In our proposed method, the balance of the character is maintained by constraining the Center of Gravity (CoG) to be lying within the supporting surface on the ground. Specifically, the vertical projection of the CoG on the ground is defined as CoG_{ground} and the supporting surface is evaluated as the area bounded by the planted foot/feet. The Center of Pressure CoP is the closest point from the supporting surface to CoG_{ground}. To maintain the balance of the character, CoP is set as the target location of CoG_{ground}:

$$C_{CoP}(\dot{V_i}) = (CoG_{ground,i} + J_{CoGground,i}\dot{V_i}) - CoP_i \quad (2)$$

where $J_{CoGground,i}$ is the Jacobian of the position derivative of CoG_{ground} with respect to the joint position derivatives, CoP_i is CoP at frame i. We enforced the balancing constraints in one of the experiments and the details are discussed in Section 6.4.

Soft and hard constraints: In the proposed method, anti foot-sliding constraints are set as hard constraints, and the bone-length, collision, balancing and the rest of the position constraints (including the morphing constraints) are set as soft constraints. While maintaining the correct bone-lengths and avoiding penetrations between body parts are important, bone-length and collision constraints are set as soft constraints to stabilize the motion when there is little open space. The balancing constraint is also defined as soft constraint because the character may not be able to keep the balance without re-planning the stepping pattern while satisfying other constraints.

4.3 Iterative Morphing

The adapted motion is computed by solving

$$\arg\min_{\dot{V}_i, \lambda_i (1 \le i \le n)} E_L + w_\Delta E_A + w E_C + \lambda_i^\intercal (H_i \dot{V^f}_i - h_i) \quad (3)$$

where λ_i are the Lagrange multipliers, $H_i \dot{V^f}_i - h_i$ are the hard constraints, w_Δ is a constant weight (we use 0.2), and w is a vector of weights for bone-length, collision, balancing and position constraints, which are set to 4.0, 5.0, 5.0 and 0.4, respectively. Finally, the updated motion can be obtained by $V_i + \dot{V}_i$.

The weights for different energy terms can be adjusted according to the desired effect. For example, in the current setting, the weight of positional soft constraints are smaller than the bone-length, collision and balancing constraints because we want to retarget body size precisely and avoid penetration of body parts when adapting the motions. Since satisfying positional constraints is less important in our experiments, we used a smaller weight for the positional soft constraints to stabilize the motion. On the other hand, if positional constraints are more important (e.g. in reaching motion), the weight of the positional soft constraints term can be larger.

5 Multiresolution Motion Adaptation

The main idea of our multiresolution motion adaptation approach is to reduce the computation of spacetime optimization by first solving the problem using a sparse number of keyframes, and then fine-tuning the results by solving a smaller scale problem locally using a shorter window. Figure 2 illustrates how the method works.

Our multiresolution motion adaptation technique is described in the rest of this section. We first describe how to compute the keyframes in Section 5.1. We next describe how to adapt the motion in Section 5.2.

5.1 Adaptive Keyframing

Now we explain how we adaptively sample keyframes from the reference motion. The quality of the resulting motion is highly affected by the keyframe selection as the initial estimation of the in-between motion is interpolated by the keyframes. We compare two commonly used keyframe selection methods, Curve Simplification [Lim and Thalmann 2001] and Frame Decimation [Li et al. 2005]. To compare the two methods, we implement them and extract keyframes from a highly dynamic motion (judo, see Figure 4(a)) and a less dynamic motion (hugging, see Figure 5(a)). In order to fit the Curve Simplification method into our framework, the curves are computed based on the 3D joint positions instead of the joint angles as done in the original framework [Lim and Thalmann 2001]. In addition, all DOF are considered in our comparison instead of a subset of DOF. After extracting the keyframes, we linearly interpolate them and compute the reconstruction error by

$$Reconstruct_{err} = \sum_{i=1}^{m} \sum_{j=1}^{k} (p^j_{rec,i} - p^j_{ori,i})^2 \quad (4)$$

where $Reconstruct_{err}$ is the motion reconstruction error of the characters, m is the total number of frames, k is the total number of joints, i and j are the indices, and $p^j_{rec,i}$ and $p^j_{ori,i}$ are the position of the j-th joint in Cartesian coordinates at i-th frame in the reconstructed and original motions, respectively.

The $Reconstruct_{err}$ of the two methods are compared in Figure 3, where *CS* and *FD* refer to Curve Simplification and Frame Decimation, respectively. It can be observed Frame Decimation outperforms Curve Simplification method in terms of motion quality, especially when the number of keyframes is small. The brute-force search in Frame Decimation extracts a near-optimal keyframe set which produced smaller reconstruction error in the experiments. One disadvantage of using Frame Decimation is that the computational cost is significantly higher when compared with Curve Simplification. However, the keyframes can be extracted in the pre-processing stage. Therefore we use Frame Decimation approach in our proposed framework.

In case that high performance is required for keyframe extraction, for example, when retargetting live captured motion from the user/player in interactive games, the Curve Simplification approach can be used with appropriate settings.

5.2 Motion Adaptation

In this section, we explain the process of adapting motion at runtime. In the original Interaction Mesh framework [Ho et al. 2010], the whole motion is edited at once by spacetime optimization. The large number of variables and constraints lead to slow convergence. Here we reduce the computational cost by first performing the spacetime optimization at the coarser level Lv_0. In other words, we are editing the keyframes at Lv_0 instead of all the frames of the motion. Since the number of frames to be solved is reduced, the number of variables and constraints are also reduced accordingly. As a result, the computational cost is greatly reduced. The edited keyframes will then be used for updating the motion in the finer level.

The first step of the motion adaptation process is to compute coarser level L_{v_0} by adaptive keyframing (Section 5.1) from the reference motion. Then, after solving the spacetime optimization on L_{v_0}, we compute the motion of the frames on the finer level (denoted here by Lv_1) by linearly interpolating the in-between frames between every two consecutive keyframes. As these interpolated postures may not satisfy the constraints (e.g. bone length constraints), we use the interpolated postures as the initial guess for the spacetime optimization in the finer level.

Small window spacetime optimization At Lv_1, the keyframe postures computed in Lv_0 will be fixed (i.e. will not be changed during the later optimization) and the poses between them are adapted by spacetime optimization. We use the same optimization method as done in Lv_0. Since the number of frames between two consecutive keyframes is small, this spacetime optimization problem can be solved quickly. The motion between each pair of keyframes is repeatedly computed to obtain the final animation. The foot trajectories of the original motion are used as the target position of the anti foot-sliding constraints to avoid foot sliding.

5.3 Parallel Computing

In this section, we explain the use of parallel computing to improve the performance of the proposed method. We will explain how to perform parallel computing in two different levels - global and local. On the global level, the proposed multiresolution hierarchical model enables motion editing and play-back in parallel. More specifically, only a small amount of time is required for the spacetime optimization at the coarser level, and the small window optimization can be done on-the-fly when playing back the animation in the previous short window. As a result, the animators can interactively edit the character poses and their sizes and observe how such edits affect the animation sequence.

The spacetime optimization in the short window is highly parallelizable. When preparing the constraints matrices for the optimization, there is no dependency between the computations on different frames. As a result, we can utilize multiple cores/threads on

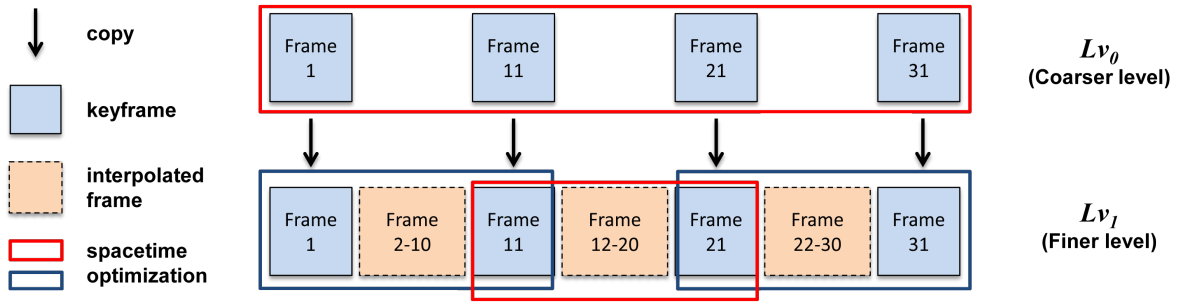


Figure 2: *An example of a multiresolution hierarchical model. The red and blue rectangles indicate different smaller scale optimizations.*

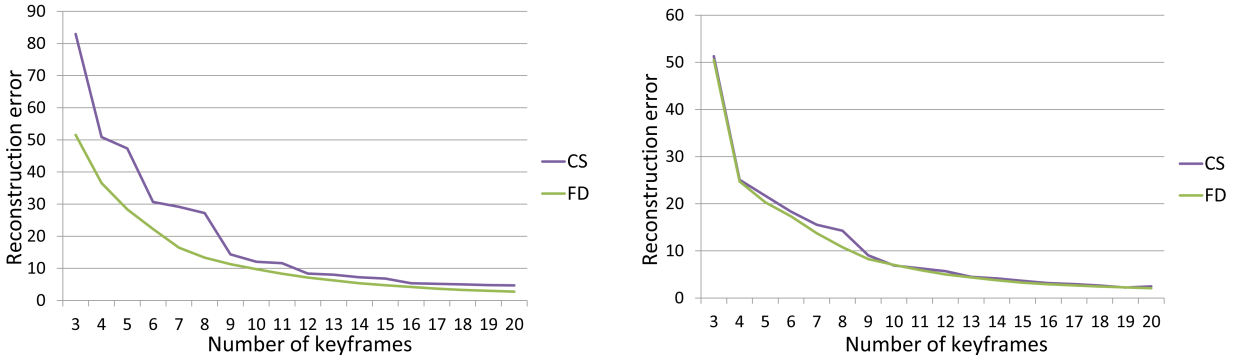


Figure 3: *The reconstruction error obtained from different keyframe extraction approaches on (a) Ohgoshi motion and (b) Hugging motion.*

CPU/GPU to compute the constraint matrix for each frame in parallel to reduce the time required. We implement our method both on CPU and GPU; the implementation details are explained in Section 6.

6 Experimental Results

To evaluate the effectiveness of the proposed method, we evaluate the computed movements in terms of motion quality and computational cost. In terms of the quality, we first check whether the proposed method can accurately retarget the reference motions to the target character sizes by evaluating the bone length error. We also examine the smoothness of the motion by comparing the movements with those computed by the original Interaction Mesh framework. Finally, we compare the computational cost of our approach and the original Interaction Mesh framework. The experiments are done using judo, hugging and fighting motions between two characters. The original Interaction Mesh approach [Ho et al. 2010] is run with 10, 20 and 30 iterations in the experiments. Finally, we show an additional experiment which enforces body balance constraints to ensure physical correctness of the retargeted motions.

Notations: For simplicity of the explanation, we refer the original Interaction Mesh spacetime optimization approach as SP, the proposed multiresolution approach with Frame Decimation keyframe extraction as MR.

6.1 Average Bone Length Error

The average bone length error is computed by

$$Bone_{err} = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{n}\sum_{j=1}^{n}\frac{|len^j_{tar,i} - len^j_{act,i}|}{len^j_{tar,i}} \qquad (5)$$

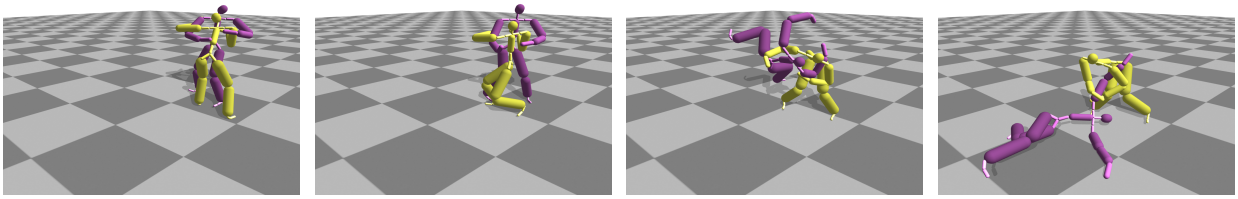
where $Bone_{err}$ is the average bone length error of the characters, m is the total number of frames, n is the total number of body segments, i and j are the indices, and $len^j_{tar,i}$ and $len^j_{act,i}$ are the target and actual bone lengths of the j-th segment at i-th frame, respectively.

Judo Motion, Ohgoshi Throw: In this experiment, we retarget the highly dynamic judo motion to characters with new morphologies. The screenshots of the original and retargeted motions are shown in Figure 4. The bone lengths and body segments of the thrower and defender are uniformly scaled up and down by 10%, respectively. The results are listed in Table 1.

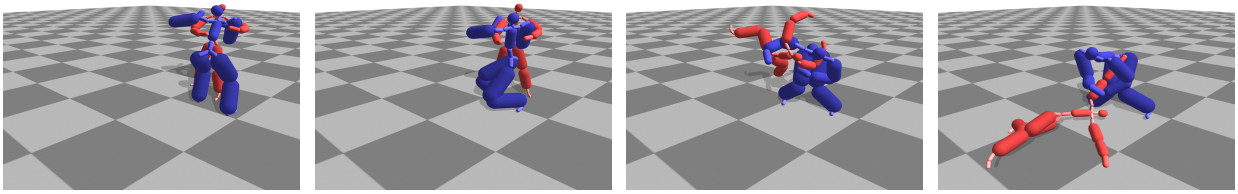
From the results produced by SP, the bone length error rate is high (4.72%) when the motion is computed by 10 iterations. The bone length error rate dropped significantly to 0.41% by using 20 iterations and coverage slowly to 0.39% with 30 iterations.

For the proposed method, we extract two sets of keyframes, 11 and 19 keyframes. From Table 1, the results show that the average bone length error decreases in the order of 65-75% when the number of iterations at Lv_1 is doubled (from 5 to 10). When comparing MR to SP, the results show that using MR results in lower bone length error rate compared to SP for a similar amount of computation time. For example, when solving SP with 10 iterations, $Bone_{err}$=4.72% and the time required is 2.55 seconds. Using MR with 11 keyframes (10 iterations at Lv_0 and 5 iterations at Lv_1), we get $Bone_{err}$=0.53% when a similar amount of computation (2.46 seconds).

Hugging - 1 In this experiment, we retarget the less dynamic hugging motion to characters with new morphologies. The screenshots of the original and retargetted motions are shown in Figure 5 (a) and (b). Again, the bone lengths and body segments of one char-

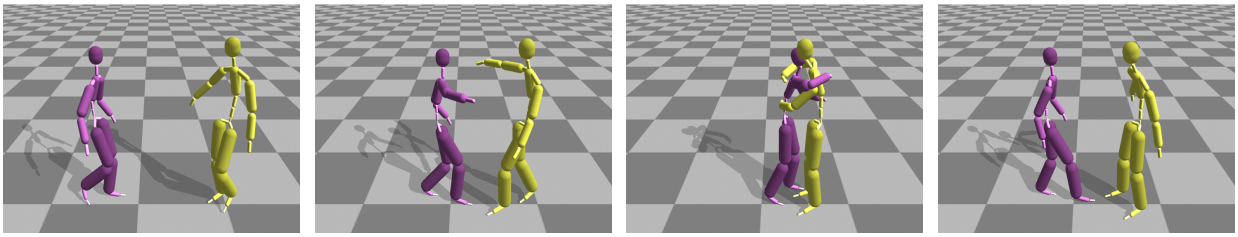


(a)

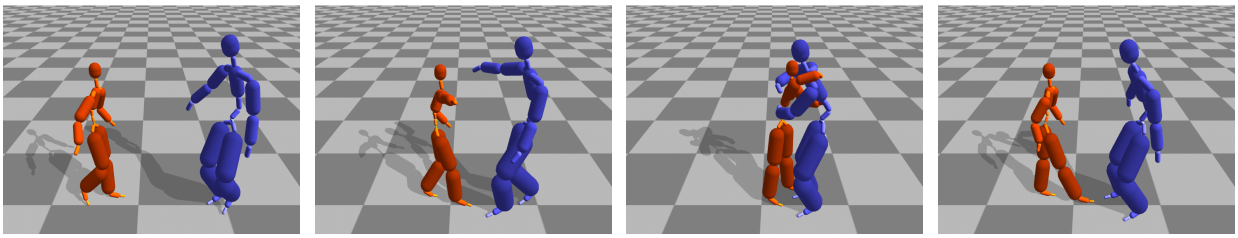


(b)

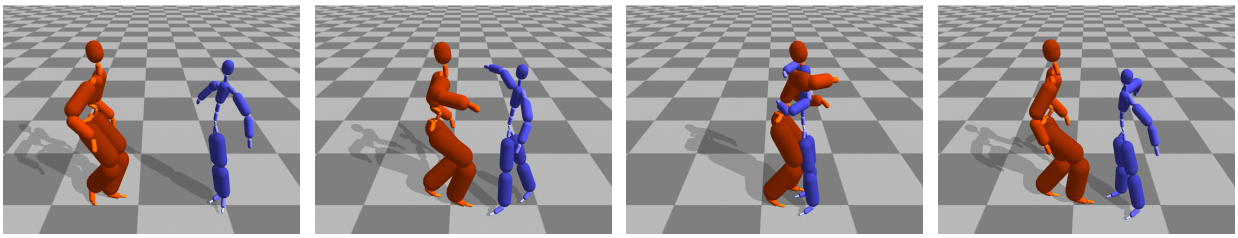
Figure 4: *The screenshots of the Ohgoshi motion - (a) Original motion. (b) Retargetted motion.*



(a)



(b) Hugging - 1



(c) Hugging - 2

Figure 5: *The screenshots of the hugging motion - (a) Original motion. (b) and (c) Retargetted motions.*

Table 1: *Results of the retargetted Ohgoshi motion*

Method	Number of Keyframes	Number of Iterations		Bone Length Error (%)	Joint Velocity Difference	Optimization at Lv_0 (s)	Total time (s)
		Lv_0	Lv_1				
Spacetime	-	-	10	4.72	0.00	-	2.55
			20	0.41	0.00		4.99
			30	0.39	0.00		7.43
Frame Decimation	11	10	5	0.53	0.01	0.54	2.46
		15	5	0.28	0.01	0.74	2.65
		10	10	0.12	0.02	0.51	4.27
	19	10	5	0.27	0.00	0.69	3.47
		15	5	0.20	0.00	1.01	3.76
		10	10	0.10	0.00	0.71	6.27

acter were uniformly scaled up by 10% (Figure 5 (b) blue) and the other character were uniformly scaled down by 10% (Figure 5 (b) red). The results are listed in Table 2.

From the results produced by SP, $Bone_{err}$=1.28% when 10 iterations were used. The bone length error rate dropped to 0.76% by solving 20 iterations and coverage slowly to 0.72% at 30 iterations. The results are similar to those obtained using SP on the judo motion explained previously.

When comparing our method MR with SP, again, MR consistently outperforms SP when a similar amount of computation is spent. For example, when solving SP in 10 iterations, $Bone_{err}$=1.28% and the time required is 5.70 seconds. In MR with 15 keyframes (15 iterations at Lv_0 and 5 iterations at Lv_1), we obtained $Bone_{err}$=0.62% when a similar amount of computation (5.20 seconds) is spent. This pattern also applies to MR with 27 keyframes (10 iterations at Lv_0 and 10 iterations at Lv_1), in which $Bone_{err}$=0.56% and 11.58 seconds is spent. On the other hand, $Bone_{err}$=0.76% is obtained when spending 11.21 seconds on solving 20 iterations in SP.

Hugging - 2 In the third experiment, we retarget hugging motion to characters with another set of morphologies which are having larger scaling factors than those in the second experiment. The screenshots of the retargetted motions are shown in Figure 5 (c). The bone lengths and body segments of one character was uniformly scaled up by 20% (Figure 5 (c) red) and the other character were uniformly scaled down by 20% (Figure 5 (c) blue). The results are listed in Table 3.

Since the characters are scaled more than the second experiments, the bone length error is expected to be larger when the same number of iterations is used when solving the optimization problem. From the results produced by SP, $Bone_{err}$=3.62% when 10 iterations were used. The bone length error rate dropped to 1.98% by solving 20 iterations and coverage slowly to 1.30% at 30 iterations.

When comparing our method MR with SP, again, MR consistently outperforms SP when a similar amount of computation is spent. For example, when solving SP in 10 iterations, $Bone_{err}$=3.62% and the time required is 6.57 seconds. In MR with 15 keyframes (15 iterations at Lv_0 and 5 iterations at Lv_1), we obtained $Bone_{err}$=0.78% when a similar amount of computation (5.12 seconds) is spent. Using MR with 27 keyframes (10 iterations at Lv_0 and 15 iterations at Lv_1) also outperforms SP, in which $Bone_{err}$=0.22% and 13.13 seconds is spent. On the other hand, $Bone_{err}$=1.98% is obtained when spending 12.28 seconds on solving 20 iterations in SP.

Back Breaking Attack: In this experiment, the bone lengths and body segments of the attacker and defender are uniformly scaled up and down by 15%, respectively. The screenshots of the original and retargeted motions are shown in Figure 6. The results are listed in Table 4.

From the results produced by SP, the bone length error rate is 3.49% when the motion is computed by 10 iterations. The bone length error rate dropped significantly to 2.20% by using 20 iterations and coverage to 1.42% with 30 iterations.

For the proposed method, we extract two sets of keyframes, 20 and 30 keyframes. When comparing MR to SP, the results again show that using MR results in lower bone length error rate compared to SP for a similar amount of computation time. For example, when solving SP with 10 iterations, $Bone_{err}$=3.49% and the time required is 6.32 seconds. Using MR with 20 keyframes (15 iterations at Lv_0 and 5 iterations at Lv_1), we get $Bone_{err}$=1.60% when a similar amount of computation (5.27 seconds).

6.2 Average Joint Velocity Difference

The average joint velocity difference is computed by

$$Joint_{vel} = \frac{1}{m}\sum_{i=2}^{m}\frac{1}{k}\sum_{j=1}^{k}(\frac{p_i^j - p_{i-1}^j}{\Delta t} - \frac{p_{ori,i}^j - p_{ori,i-1}^j}{\Delta t})^2 \quad (6)$$

where $Joint_{vel}$ is the average joint velocity difference of the characters, m is the total number of frames, k is the total number of joints, i and j are the indices, Δt is the interval between two consecutive frames, and p_i^j and p_{i-1}^j are the position of the j-th joint in Cartesian coordinates at i-th and $i-1$-th frame of the edited motion, and $p_{ori,i}^j$ and $p_{ori,i-1}^j$ are the position of the j-th joint in Cartesian coordinates at i-th and $i-1$-th frame of the original motion. The $Joint_{vel}$ computed from the judo, hugging (1 and 2) and back breaking attack motions are listed in Table 1, 2 and 3, and 4, respectively. The results indicate that the joint velocities differences are small and the smoothness of the motion in the original motion is preserved in the adapted motions.

In summary, by evaluating the average bone length error and average joint velocity difference, we show that our proposed method outperforms the original Interaction Mesh framework in retargetting motions as our method can synthesize the motion when a similar amount of computation is used.

6.3 Computational Cost

In this section, we compare the performance of the proposed method with SP in terms of computational cost. Table 1 to 4 show that the computational costs of the motions produced by MR is significantly lower than those computed by SP when the $Bone_{err}$ is at similar level. For example, when retargetting the judo motion, the total time required by the MR with 19 keyframes (10 iterations at Lv_0 and 5 iterations at Lv_1) is 3.47 seconds, which is only 46.70% of the time required by the SP with 30 iterations.

In addition, the proposed multiresolution approach can retarget the motions on-the-fly when playing back the animation. As explained in Section 5.3, only Lv_0 optimization has to be solved before displaying the animation in the initial stage. Using the judo example again, while the total time required by MR is 3.47 seconds, the Lv_0 low-resolution optimization took 0.69 seconds only as shown in Table 1. For Lv_1 optimization, which is the full-resolution small window optimization, the average time required for solving one frame in one iteration is 6.12 ms. Using a standard 30Hz frame rate (i.e. 33.33 ms per frame) for playing back animations and movies, our proposed method can solve 5 iterations for two characters on average in the Lv_1 optimization on-the-fly.

The experiments are run on a computer with Intel Core-i7 Processor 3.40 GHz and nVidia GeForce GTX 285 graphics card. The system is implemented on Windows with Visual C++. The constraint matrices are calculated in parallel on 4 threads on CPU using OpenMP [OpenMP Architecture Review Board 2008] and ViennaCL 1.1.2 [Rupp 2011] is used as the linear solver on GPU.

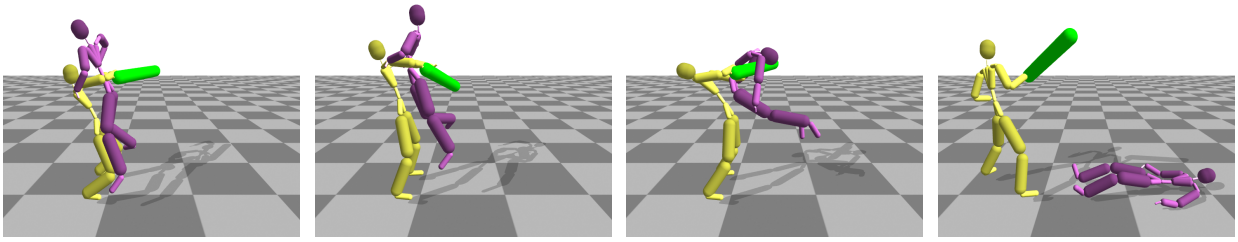
In summary, experimental results show that our proposed method is more computationally efficient than the original Interaction Mesh framework. In addition, the multiresolution approach enables motion retargetting and playing back the animation in parallel. By this, we can achieve a *near-realtime* performance with a small amount of time required for Lv_0 optimization as initialization.

Table 2: *Results of the retargetted Hugging - 1 motion*

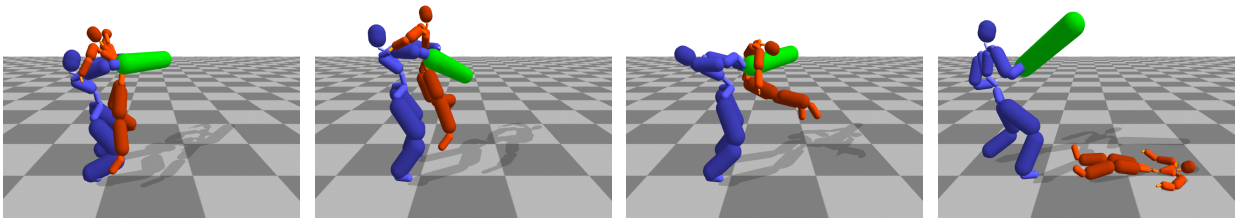
Method	Number of Keyframes	Number of Iterations		Bone Length Error (%)	Joint Velocity Difference	Optimization at Lv_0 (s)	Total time (s)
		Lv_0	Lv_1				
Spacetime	-	-	10	1.28	0.00	-	5.70
			20	0.76	0.00		11.21
			30	0.72	0.00		16.85
Frame Decimation	15	10	5	0.81	0.00	0.88	4.74
		15	5	0.62	0.00	1.31	5.20
		10	10	0.52	0.00	0.89	8.72
	27	10	5	1.10	0.00	1.42	6.60
		15	5	0.91	0.00	2.09	7.14
		10	10	0.56	0.00	1.34	11.58

Table 3: *Results of the retargetted Hugging - 2 motion*

Method	Number of Keyframes	Number of Iterations		Bone Length Error (%)	Joint Velocity Difference	Optimization at Lv_0 (s)	Total time (s)
		Lv_0	Lv_1				
Spacetime	-	-	10	3.62	0.00	-	6.57
			20	1.98	0.00		12.28
			30	1.30	0.00		18.09
Frame Decimation	15	15	5	0.78	0.01	1.63	5.12
		10	15	0.33	0.00	1.24	11.75
		15	15	0.23	0.00	1.60	12.03
	27	15	5	0.55	0.00	2.31	6.18
		10	15	0.22	0.00	1.71	13.13
		15	15	0.22	0.00	2.33	13.81



(a)



(b)

Figure 6: *The screenshots of the back breaking attack motion - (a) Original motion. (b) Retargetted motions.*

Table 4: *Results of the retargetted Back breaking attack motion*

Method	Number of Keyframes	Number of Iterations		Bone Length Error (%)	Joint Velocity Difference	Optimization at Lv_0 (s)	Total time (s)
		Lv_0	Lv_1				
Spacetime	-	-	10	3.49	0.00	-	6.32
			20	2.20	0.00		11.70
			30	1.42	0.00		17.24
Frame Decimation	20	10	5	2.31	0.02	1.46	4.78
		15	5	1.60	0.01	1.82	5.27
		10	15	1.25	0.02	1.50	11.95
	30	10	5	2.53	0.00	1.89	5.83
		15	5	2.22	0.00	2.35	6.07
		10	15	1.51	0.00	1.80	13.34

6.4 Motion Adaptation with Body Balance Constraints

In this experiment, we present an example of retargeting a carrying object motion while enforcing body balance constraints to ensure the physical correctness of the motion. Specifically, we added a soft constraint to guide the Center of Pressure (COP) of the character to be lying over the support polygon to maintain the balance while retargeting the motion to a character with larger body sizes (i.e. 10% larger). The results are shown in Figure 7. The result generated by our method (Figure 7 (c)) shows the body parts are coordinated such as the knees) to keep the COP to be lying over the support polygon. The results highlight the advantage of our proposed method over another real-time motion adaptation approach [Al-Asqhar et al. 2013] in which physical constraints can be enforced in our method to produce physically feasible motions. The readers are referred to the attached video demo to evaluate the quality of the retargeted motions.

7 Conclusion and Discussion

In this paper, we propose a multiresolution approach for solving spacetime optimization problems efficiently. We apply our method to retarget motions of closely interacting characters based on the design of the Interaction Mesh framework [Ho et al. 2010]. Experimental results show that our method consistently outperforms the original Interaction Mesh framework in terms of motion quality and computational efficiency. We further show that the highly parallelizable nature of the multiresolution model enables motion retargeting on-the-fly when playing back the animations and *near-realtime* performance can be achieved.

While the experimental results show the proposed method outperforms the motion adaptation framework presented in [Ho et al. 2010], the selection of the parameters such as window size in the high-resolution (i.e. fine level) optimization can affect the quality of the resultant motion as well as the speed of the optimization. When the size of the small window is large, the low-resolution (i.e. coarse level) optimization step will contain sparse keyframes only. Since the in-between frames in the small windows will be interpolated from the adapted keyframes, sparse keyframes will result in over-smoothing the motion. As a result, some of the detailed movements in the motion will be lost. In our proposed method, the keyframes are extracted automatically using Frame Decimation. In all of our experiments, every small window contains less than 10 frames and the detailed movements are preserved in the adapted motions. Another advantage to keep the small windows with fewer frames is to facilitate the optimization problem to be solved more easily. With fewer variables and constraints in smaller windows, the computation cost required is usually lower than large optimization problems. As a result, higher performance in terms of speed can be achieved.

Another related issue is the arrangement of the small windows. In the proposed method, the keyframes will not be edited and used for bounding the small windows. By minimizing the acceleration energy across frames in the small window spacetime optimization, we do not encounter any results with non-smooth motion in all of our experiments. Although no unsmoothness is observed, we expect non-smooth motion can be produced when there is a large change in the acceleration of the joints across the small windows. One possible solution is to have overlapping with the previous window (e.g. last few frames in previous windows) when minimizing the acceleration energy while keeping the frames in the previous window and the border frames non-editable.

One limitation of our proposed method is the assumption in the smoothness of the movements between keyframes. When solving the optimization on the small windows at fine level, the in-between frames will be interpolated from the keyframes which bound the small window. While smooth motion can be created, some high frequency movements such as those in the back breaking attack experiment presented in Section 6 can be lost (see the attach video). One of the possible solutions is to extract enough keyframes to capture such kind of detailed movements. Another possible solution is to add the motion details from the original motion when editing the motion at fine levels similar to [Lee and Shin 1999], which is one of our future directions.

Acknowledgements

This work was supported in part by Hong Kong Baptist University Science Faculty Research Grants (FRG1/12-13/055 and FRG2/12-13/078) and the Hong Kong Research Grant Council (Project No. GRF210813).

References

AL-ASQHAR, R. A., KOMURA, T., AND CHOI, M. G. 2013. Relationship descriptors for interactive motion adaptation. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, New York, NY, USA, SCA '13, 45–53.

BRUDERLIN, A., AND WILLIAMS, L. 1995. Motion signal processing. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '95, 97–104.

GLEICHER, M. 1997. Motion editing with spacetime constraints. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, ACM Press, New York, NY, USA, 139–ff.

GLEICHER, M. 1998. Retargetting motion to new characters. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 33–42.

HO, E. S. L., AND SHUM, H. P. H. 2013. Motion adaptation for humanoid robots in constrained environments. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 1–6.

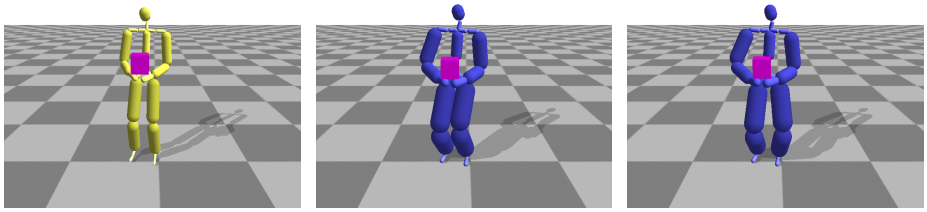
HO, E. S., KOMURA, T., AND TAI, C.-L. 2010. Spatial relationship preserving character motion adaptation. *ACM Transactions on Graphics 29*, 4, 1–8.

HO, E. S. L., CHAN, J. C. P., KOMURA, T., AND LEUNG, H. 2013. Interactive partner control in close interactions for real-time applications. *ACM Trans. Multimedia Comput. Commun. Appl. 9*, 3 (July), 21:1–21:19.

LEE, J., AND SHIN, S. Y. 1999. A hierarchical approach to interactive motion editing for human-like figures. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 39–48.

LI, S., OKUDA, M., AND TAKAHASHI, S. 2005. Embedded keyframe extraction for cg animation by frame decimation. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, 1404 –1407.

LIM, I. S., AND THALMANN, D. 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 vol.2.



(a) (b) (c)

Figure 7: *The screenshots of the carrying object motion - (a) Original motion. (b) Retargetted motions obtained using [Ho et al. 2010]. (c) Retargetted motions obtained using our method with body balance constraints.*

LIU, C. K., AND POPOVIĆ, Z. 2002. Synthesis of complex dynamic character motion from simple animations. *ACM Trans. Graph. 21*, 3 (July), 408–416.

LIU, Z., GORTLER, S. J., AND COHEN, M. F. 1994. Hierarchical spacetime control. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '94, 35–42.

LIU, C. K., HERTZMANN, A., AND POPOVIĆ, Z. 2006. Composition of complex optimal multi-character motions. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, 215–222.

O'BRIEN, C., DINGLIANA, J., AND COLLINS, S. 2011. Spacetime vertex constraints for dynamically-based adaptation of motion-captured animation. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, New York, NY, USA, SCA '11, 277–286.

OPENMP ARCHITECTURE REVIEW BOARD, 2008. OpenMP application program interface version 3.0, May.

POPOVIĆ, Z., AND WITKIN, A. 1999. Physically based motion transformation. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 11–20.

ROSE, C., GUENTER, B., BODENHEIMER, B., AND COHEN, M. F. 1996. Efficient generation of motion transitions using spacetime constraints. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '96, 147–154.

RUPP, K. 2011. *ViennaCL 1.1.2 User Manual*.

SHI, X., ZHOU, K., TONG, Y., DESBRUN, M., BAO, H., AND GUO, B. 2007. Mesh puppetry: Cascading optimization of mesh deformation with inverse kinematics. *ACM Trans. Graph. 26*, 3 (July).

SI, H., AND GAERTNER, K. 2005. Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In *Proc of the 14th International Meshing Roundtable*, 147–163.

WITKIN, A., AND KASS, M. 1988. Spacetime constraints. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 159–168.

WITKIN, A., AND POPOVIC, Z. 1995. Motion warping. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '95, 105–108.