Lattice-Guided Human Motion Deformation for Collision Avoidance

Masaki Oshita
Kyushu Institute of Technology
Iizuka, Fukuoka, Japan
e-mail:oshita@ces.kyutech.ac.jp

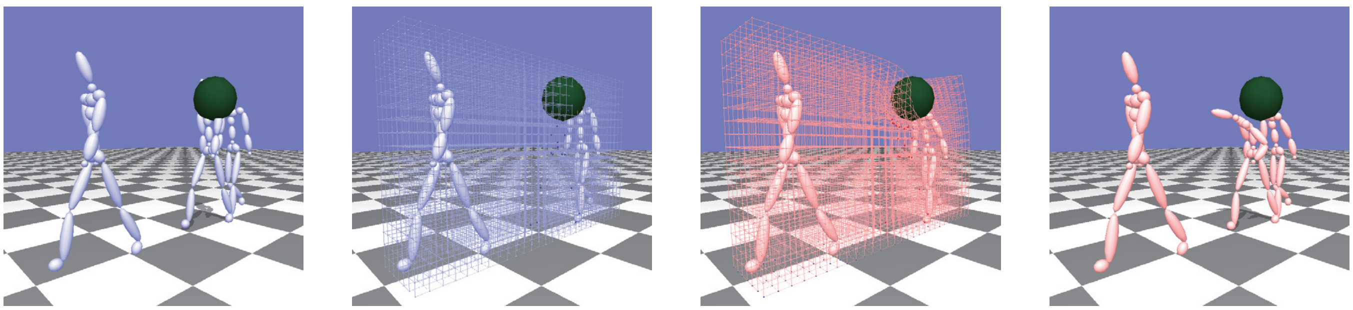


Figure 1: Overview of lattice-guided human motion deformation. (a) Input motion and obstacles. The obstacle (green sphere) overlaps with the input motion. (b) A lattice is constructed so that it covers the input motion. (c) The lattice is deformed to avoid intersection of the body and obstacles. The joint positions (red dots) of each frame pose are altered based on the lattice deformation. (d) An output pose of the deformed motion is computed from the altered joint positions.

ABSTRACT

In this paper, we propose a lattice-guided human motion deformation method. Our key idea is to warp the motion space by applying an existing shape deformation technique to efficiently realize motion deformation for collision avoidance. An input motion is deformed based on the deformation of a lattice that covers the input motion. The lattice is constructed so that it covers an input motion, and it is deformed to avoid still or moving obstacles. The constraints to deform the lattice are determined based on the intersections between the vertices of the lattice that overlaps the character's body during the input motion and the space−time volume of the obstacles. Using these constraints, the lattice is deformed by applying an as-rigid-as-possible shape deformation. The joint positions of each frame pose of the input motion are altered by the lattice deformation and used to deform that frame pose. By introducing a lattice deformation, the constraints to deform a pose can be efficiently obtained while keeping space−time consistency. We evaluated our method using several motions and situations. The results show the effectiveness of our approach.

MiG '17, November 8–10, 2017, Barcelona, Spain

ACM ISBN 978-1-4503-5541-4/17/11…$15.00
https://doi.org/10.1145/3136457.3136475

CCS CONCEPTS

• **Computing methodologies → Animation**; **Motion processing**; *Mesh geometry models*;

KEYWORDS

Human Motion Deformation, Collision Avoidance, Lattice, Shape Deformation

ACM Reference Format:
Masaki Oshita. 2017. Lattice-Guided Human Motion Deformation for Collision Avoidance. In *Proceedings of MiG '17*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3136457.3136475

1 INTRODUCTION

Motion deformation that adopts existing human motion data to a new environment or situation is often required in both off-line animation editing and on-line human motion synthesis. Motion deformation for collision avoidance (i.e. deforming a motion to avoid still or moving obstacles) is an especially difficult problem, because an obstacle is a space−time volume and the constraints needs to avoid them are not clear. Keeping consistency between frames when deforming each frame pose is also an issue.

In this paper, we propose a lattice-guided human motion deformation method. Our key idea is to warp the motion space by applying an existing shape deformation technique to efficiently realize motion deformation for collision avoidance. An input motion is deformed based on the deformation of a lattice that covers the input motion. An overview of our method is shown in Figure 1. The lattice is constructed so that it covers the input motion and is deformed to avoid still or moving obstacles. The constraints to deform the lattice are determined by the intersections between the vertices of lattice that overlap the character's body during the input motion and the space–time volume of the obstacles. Using these constraints, the lattice is deformed by applying an as-rigid-as-possible shape deformation [Alexa et al. 2000; Chao et al. 2010; Sorkine and Alexa 2007]. The joint positions of each frame pose of the input motion are altered according to the lattice deformation and used to deformed the frame pose. By introducing a lattice deformation, the constraints to deform a pose can be efficiently obtained while space–time consistency is maintained. We evaluated our method using several motions and scenarios, and the results show the effectiveness of our approach.

The remainder of this paper is organized as follows. Section 2 reviews related work and Section 3 provide an overview of our method. Section 4 describes lattice construction and deformation. Section 5 explains motion deformation and Section 6 presents the experimental results and discussion. Finally, Section 7 concludes this paper.

2 RELATED WORK

There are various methods for pose deformation. Inverse kinematics [Wei and Chai 2011; Zhao and Badler 1994] is one common technique. There are several approaches to inverse kinematics such as numerical, analytical, and data-driven methods. To apply inverse kinematics, the target position(s) of the end-effector(s) must be specified. Inverse kinematics is often used for generating or deforming a reaching motion when the end-effector and target position are specifically determined. However, this is not suitable for collision avoidance because the specific constraints for the end-effectors cannot be determined. Moreover, if inverse kinematics is applied to each frame pose, the consistency between frame poses is not guaranteed. Some path planning algorithms can be combined with inverse kinematics for generating or deforming collision free motions [Shapiro et al. 2007; Yamane et al. 2004]. However, such a path planning algorithm must repeat local deformation to avoid collisions, and it is difficult to deform the overall pose and motion smoothly.

Space–time optimization is an approach for motion synthesis [Gleicher 1998; Liu and Popovic 2002; Witkin and Kass 1988]. It can be used to deform a motion in order to minimize an objective function while meeting some constraints. Although it is possible to define an objective function for collision avoidance, because the dimensionality of human motion is very high, there is no guarantee that a plausible motion can be generated. Moreover, this approach incurs high computational costs and cannot generate an output motion within a reasonable time.

Motion warping [Witkin and Popović 1995] is a fundamental technique for motion editing and is widely used. However, it is usually applied in joint space. Motion warping in 3-dimensional space is not straightforward. It is possible to deform an end-effector trajectory in 3-dimensional space and use it with inverse kinematics [Lee and Shin 1999]. Warping a full-body pose in 3-dimensional space is not common and is the novelty of our approach.

Combining computational geometry with character animation is an interesting approach. Shape deformation is commonly used to deform a character's geometry given skeleton poses for rendering. However, this type of geometric deformation does not influence character pose or motion. Ho et al. [2010] introduced an interaction mesh to maintain the relationship between two characters or between a character and the environment. Al-ashqar et al. [2013] introduced a relationship descriptor to maintain the relationship between a character and the environment. These methods are suitable for maintaining a distance from the environment including still obstacles. However, they are not suitable for collision avoidance for moving objects. Iwamoto et al. [2015] introduced a multi-layer lattice model for the physical simulation of secondary skin deformation.

Generating avoidance motion for given moving obstacles is a challenging problem and various approaches have been developed [Oshita and Masaoka 2011; Wampler et al. 2010; Zordan et al. 2007]. Most previous methods take a data-driven approach and find an appropriate motion from a collection of motions based on given obstacles. However, these data-driven approaches require a large number of motions to handle arbitrary obstacles in general. By combining a motion deformation method with such a data-driven approach, more appropriate avoidance motions can be generated.

3 OVERVIEW

This section provides an overview of our method. Our method consists of three steps: lattice construction, lattice deformation, and motion deformation.

Given an input motion and still or moving obstacles, our method deforms the input motion to avoid the obstacles and generates an output motion. Our method deforms each frame pose of the input motion. The duration of the motion does not change. In addition, our method maintains the foot-contact constraints of the input motion. This constraint is introduced to avoid simply translating the whole motion to avoid obstacles. Instead, the character is expected to avoid the obstacles by deforming the frame poses during the motion while keeping the foot-contact constraints. Depending on the obstacles, it may not be possible to avoid all of them. In such case, our method generates an output motion where the character tries to avoid the obstacles but eventually collides with them. We consider this to be a reasonable output.

A human character model is represented as an articulated figure. A pose is represented by the position and orientation of the root (the pelvis) and joint rotations. A motion is represented by a series of poses. In our experiments, a character model with 18 joints is used.

Each still or moving obstacle is a space–time volume. They can be represented in any way as long as collision detection and penetration depth can be computed. In our implementation, the obstacles are represented by a set of spheres. The size of each sphere can be either be a constant or time-varying function. A moving obstacle is additionally represented by the trajectory of the sphere.

Basically, our method deforms a single character motion. It can be applied to deform multiple characters' motions that interact with each other. For example, our method can calculate the avoidance motion of one character in response to an attack motion of another character. In this case, only the avoiding character's motion is deformed and the attacking character's motion is treated as the motion of a moving obstacle. Deforming multiple characters'

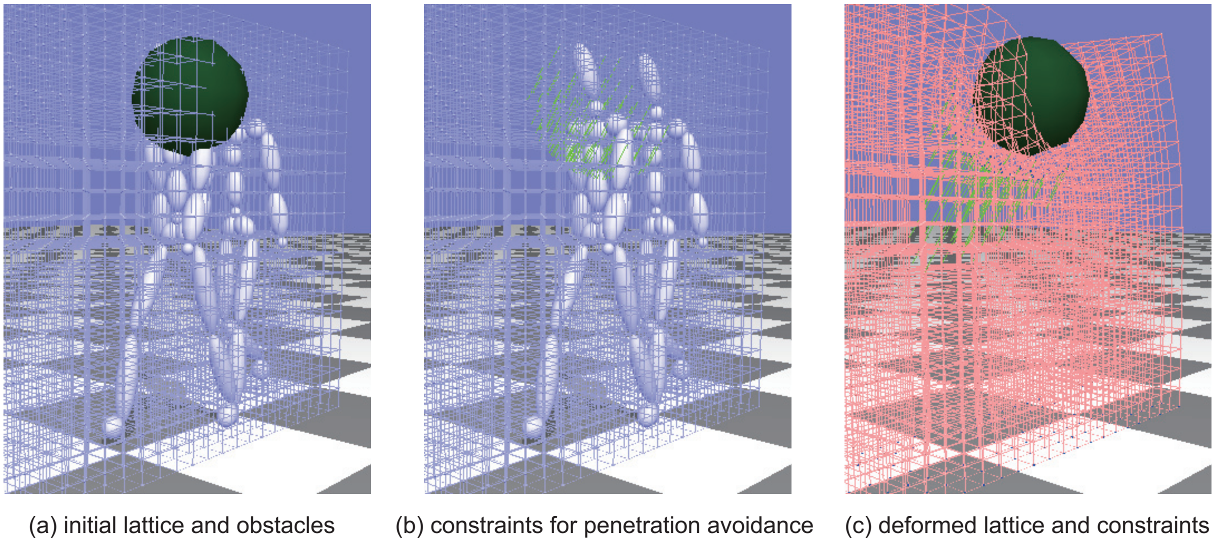


Figure 2: Lattice deformation. Constraints for penetration avoidance are visualized by the green segments.

motions together is more complicated. It can be done by repeating small deformations of each character's motion. However, such an application is outside of the scope of this paper.

4 LATTICE CONSTRUCTION AND DEFORMATION

4.1 Lattice Construction

A lattice is constructed so that it covers the entire input motion. Using the joint positions of all frame poses of the input motion, we determine the axis-aligned minimum bounding box. A lattice is generated by dividing the bounding box at a specific size. In our experiment, the size of each cell is $0.1 \times 0.1 \times 0.1m$.

4.2 Collision Detection and Constraints

This step determines the constraints for each vertex of the lattice based on the intersection between the vertices of the lattice that overlaps the character's body during the input motion and the space–time volume of the obstacles. Not all vertices of the lattice need to avoid the obstacles. Only the vertices at which the character's body and obstacle both occur should be moved to avoid the intersection.

First, a time-varying probability function is computed for each vertex of the lattice. When the vertex overlaps any part of the character's body, the probability becomes 1.0. Otherwise, the probability is 0.0. The time-varying probability functions for all vertices are determined based on the frame poses of the input motion.

Second, the overlap between the vertices and obstacles are checked for all frames. Because we use spheres to represent obstacles, as explained in Section 3, the overlap between the obstacles and vertices is easily checked.

Third, for the vertices that overlap the obstacles, the spatial constraints needed to move to a target position for penetration avoidance are determined. Determining such target positions is not straightforward. If the vertices are simply moved to the nearest point on the surface of the overlapping obstacle, even though these vertices could avoid overlapping with the obstacle, the lattice still overlaps with the obstacle. To avoid this kind of situation, a group of vertices that overlap with the same obstacle should move in the same direction to deform the lattice. For this purpose, we move the vertices toward the center of the input motion. The center of motion $\mathbf{c}$ is computed from all frame poses in advance. Each vertex is moved in direction

$$\mathbf{d}_i = \mathbf{c} - \mathbf{v}_i, \tag{1}$$

where $\mathbf{v}_i$ is the i-th initial vertex position. The distance to the nearest surface in direction $\mathbf{d}_i$ is computed to determine the target positions of the vertices. When a vertex overlaps with obstacles in multiple frames, the target position is computed for each frame and the target potion that is farthest from the initial vertex position is used as the spatial constraint. An example of the constraints for penetration avoidance is shown in Figure 2 (b).

4.3 Fixing Constraints

As explained in Section 3, our method maintains the foot-contact constraints of the input motion. To satisfy this constraint, the bottom of the lattice is fixed to the ground. Without these constraints, the whole lattice would simply be translated to avoid obstacles and the input motion would not be deformed. Therefore, fixing constraints are necessary. The fixing constraints are given to all vertices that contact the ground. Our method can handle uneven ground as well.

4.4 Lattice Shape Deformation

Based on the constraints on the vertices of the lattice, the lattice is deformed so that the shape of original lattice is maintained as

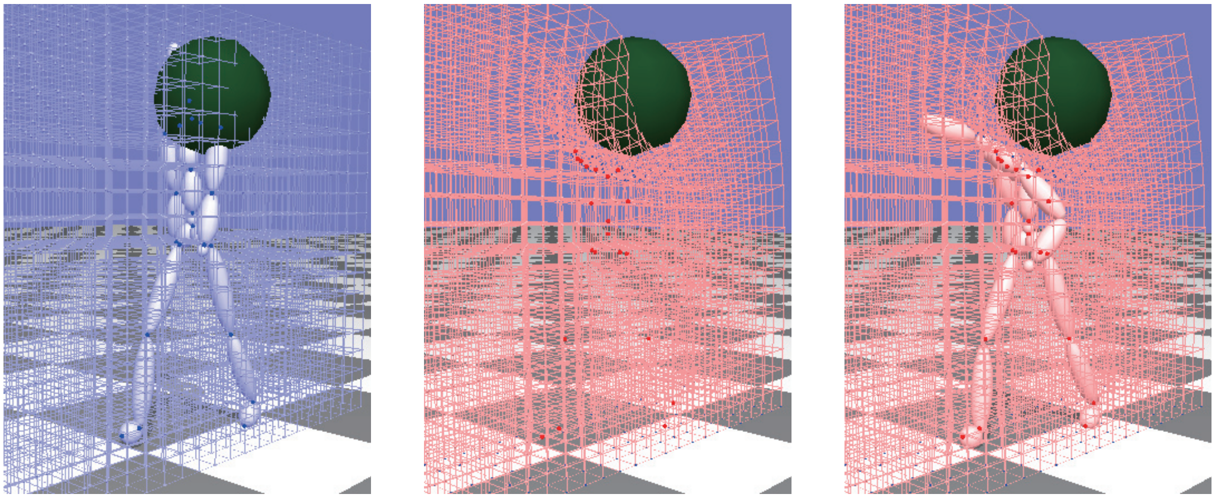


Figure 3: Translation of joint positions. (a) Initial joint positions are represented by blue dots. (b) Translated joint positions are represented by red dots. (c) Deformed pose based on the translated joint positions.

much as possible while satisfying the constraints. To realize such a deformation, we employ an as-rigid-as-possible deformation [Alexa et al. 2000; Chao et al. 2010; Sorkine and Alexa 2007]. Essentially, this deformation preserves the energy for each triangle shape,

$$E(M) = \sum_{t \in T} a_t \sum_{i,j \in t} w_{ij} \left| \left(\mathbf{x}'_i - \mathbf{x}'_j \right) - \left(\mathbf{x}_i - \mathbf{x}_j \right) \right|^2 \tag{2}$$

where T is a set of triangles in mesh M, a_t is the area of triangle t, and $\{i, j\}$ is a pair of vertices. In addition, $\mathbf{x}_i$ and $\mathbf{x}'_i$ are the positions of the i-th vertex before and after deformation, respectively. The energy is measured by the sum of the norm of the edge vector changes. We used the libigl software library [Jacobson et al. 2016] for our implementation.

5 MOTION DEFORMATION

Motion deformation is realized by deforming each frame pose independently. Because the pose constraints are determined based on the lattice deformation, the space–time consistency of the constraints and deformed poses is guaranteed. To deform a pose, the joint positions are translated based on a lattice deformation and the pose is determined based on the altered joint positions.

5.1 Translation of Joint Positions

The positions of the root (pelvis) and all joints are translated based on the initial and deformed lattices so that their local positions within the lattices are preserved. First, joint positions $\mathbf{p}_i$ are computed using forward kinematics from the frame pose. Second, their local positions are determined from the initial lattice. The local position contains the cell that contains position c_i and the barycentric coordinate position within cell $\mathbf{b}_i$. Third, translated positions $\mathbf{p}'_i$ are obtained based on the deformed lattice, c_i, and $\mathbf{b}_i$. An example of translation of joint positions is shown in Figures 3(a) and (b).

5.2 Pose Deformation

The frame pose is deformed to satisfy translated positions $\mathbf{p}'_i$. As each joint point is translated according to the lattice deformation, the body length constraints between connecting joints may be broken. In practice, the body length errors do not become very large because we use as-rigid-as-possible shape deformation. However, the deformation still causes some errors. Our method handles these errors and determines a stable output pose.

The root orientation and rotations of all joints are calculated to satisfy the constraints as much as possible. Each rotation is determined from the root to the end-effector joints independently.

When the joint is connected to one joint (e.g., the arm and leg joints), the joint rotation is computed to satisfy the position of the connecting joints. The rotations of the i-th joint $\mathbf{r}_i$ is computed by

$$\mathbf{r}_i = (\mathbf{p}'_j - \mathbf{p}_i) \times (\mathbf{p}_j - \mathbf{p}_i), \tag{3}$$

where j is the connecting joint, $\mathbf{p}_i$ and $\mathbf{p}_j$ are the current positions, and $\mathbf{p}'_j$ is the target position of the j-th joint. In addition, $\mathbf{r}_i$ is a rotation vector whose direction represents the rotation axis and whose length represents the rotation angle. Joint rotation matrix $\mathbf{R}_i$ is computed from $\mathbf{r}_i$.

When the joint is connected to three joints (e.g., the pelvis and chest joint), the rotation is computed to satisfy the positions of the three connecting joints. The rotations of i-th joint $\mathbf{R}_i$ are computed by

$$\mathbf{v}_j = \mathbf{p}_j - \mathbf{p}_i \tag{4}$$

$$\mathbf{v}'_j = \mathbf{p}'_j - \mathbf{p}_i \tag{5}$$

$$\mathbf{R}_i = \left(\mathbf{v}'_j \mathbf{v}'_k \mathbf{v}'_l \right) \left(\mathbf{v}_j \mathbf{v}_k \mathbf{v}_l \right)^{-1} \tag{6}$$

where j, k, and l are three connecting joints and $\mathbf{R}_i$ is a 3×3 rotation matrix. Because rotation matrix $\mathbf{R}_i$ may become irregular, it must be normalized so that each column vector becomes 1.0 in length and is perpendicular to the others.

Using the above algorithm, all joint rotations are computed deterministically. As mentioned above, the constraints may not be satisfied. Although it is possible to introduce an optimization process and repeat small deformations until the errors of the joint positions are minimized, we did not take such an approach because it makes the deformed poses nondeterministic and may break the consistency between frame poses.

6 EXPERIMENTS AND DISCUSSION

We evaluated our method using motion capture clips and manually specified obstacles. Some results are shown in Figure 4 and the accompanying video. We successfully generated deformed motions that avoid stationary or moving obstacles.

The average execution times were measured on a standard desktop computer with an Intel Core i7-6700 3.4 GHz CPU. The lattice deformation process took 1.025, 0.484, and 0.607 s in the cases of Figure 4 top, middle, and bottom, respectively. The lattice sizes were $7 \times 29 \times 19$, $10 \times 10 \times 18$, and $9 \times 14 \times 18$, respectively. The pose deformation process per frame took 0.00012 s on average and 0.00038 s at most.

There are several limitations of our current method that can be observed in the results. Because the motion space is deformed before the motion deformation and the lattice itself does not animate during motion, when the character's body passes the same point on the lattice more than once, unnecessary avoidance movement can be generated. For example, in our results for twisting and punching motions, when the character moves back to the initial pose after twisting and punching, unnecessary avoidance movement is generated. This problem can be solved by animating the lattice shape during motion. However, this solution could be computationally inefficient and make the deformed motions unstable.

As discussed in Section 5.2, because the joint positions are altered based on a lattice deformation, the constraints of the character's skeleton such as body lengths and possible range of joint rotation are not maintained. In our results, changes in the distances between joints (body lengths) did not become a problem. The deformed poses are successfully computed by our method. However, the possible range of joint rotation is not always satisfied and unfeasible poses sometimes appear. This problem can be solved by introducing limitations to the joint rotation during pose deformation. However, the deformed poses may not satisfy the joint positions and cannot avoid obstacles. Moreover, when a pose is substantially deformed, the balance of the character might not be maintained. Basically, our method is intended to deform original poses a little to avoid obstacles. When a large deformation is required for collision avoidance, motion deformation is not suitable and the motion must be changed to another motion to handle the obstacles.

Our method is limited to 3-dimensional space and does not deform motion in the time domain. Because the duration of motion does not change, when a large pose deformation occurs, the character may move very quickly. When avoiding a moving obstacle, control of the motion timing is important, although it is not considered by our current method.

The cell size could be an important parameter in our method. As mentioned in Section 4, we used $0.1 \times 0.1 \times 0.1$ m in our experiments. When the cell size is too large, the constraints for collision avoidance are not determined properly and the deformed motion penetrates obstacles. When the cell size is too small, the shape deformation still works properly. However, the computational cost increases. In our experiments, as the lattice sizes are relatively small, the shape deformation (as-rigid-as-possible deformation) is completed quickly (in less than one second) and the whole process for motion deformation runs in real time.

7 CONCLUSION

In this paper, we proposed a lattice-guided method for human motion deformation. Our key idea is to warp the motion space by applying an existing shape deformation technique to realize motion deformation for collision avoidance efficiently. By introducing a lattice deformation, the constraints needed to deform a pose can be efficiently obtained while maintaining space–time consistency. Although the current method has several limitations, we believe that our approach is novel and promising. Our future work includes extending our method to solve these limitations.

ACKNOWLEDGMENTS

This work was supported in part by a Grant-in-Aid for Scientific Research (No. 15H02704) from the Japan Society for the Promotion of Science (JSPS).

REFERENCES

Rami Ali Al-ashqar, Taku Komura, and Myung Geol Choi. 2013. Relationship Descriptors for Interactive Motion Adaptation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2013*. 45–53.

Marc Alexa, Daniel Cohen-Or, and David Levin. 2000. As-rigid-as-possible shape interpolation. In *SIGGRAPH '00 Proceedings*. 157–164.

Isaac Chao, Ulrich Pinkall, Patrick Sanan, and Peter Schröder. 2010. A simple geometric model for elastic deformations. *ACM Transactions on Graphics (SIGGRAPH 2010)* 29, 4 (2010), Article No. 38.

Michael Gleicher. 1998. Retargetting motion to new characters. In *SIGGRAPH '98 Proceedings*. 33–42.

Edmond S. L. Ho, Taku Komura, and Chiew-Lan Tai. 2010. Spatial Relationship Preserving Character Motion Adaptation. *ACM Transactions on Graphics (SIGGRAPH 2010)* 29, 4 (2010), Article No. 33.

Naoya Iwamoto, Hubert P. H. Shum, Longzhi Yang, and Shigeo Morishima. 2015. Multi-layer Lattice Model for Real-Time Dynamic Character Deformation. *Computer Graphics Forum (Pacific Graphics 2015)* 34, 7 (2015), 99–109.

Alec Jacobson, Daniele Panozzo, et al. 2016. libigl: A simple C++ geometry processing library. (2016). http://libigl.github.io/libigl/.

Jehee Lee and Sung Young Shin. 1999. A Hierarchical Approach to Interactive Motion Editing for Human-like Figures. In *SIGGRAPH '99 Proceedings*. 39–48.

C. Karen Liu and Zoran Popovic. 2002. Synthesis of Complex Dynamic Character Motion from Simple Animations. *ACM Transactions on Graphics (SIGGRAPH 2002)* 21, 3 (2002), 408–416.

Masaki Oshita and Naoki Masaoka. 2011. Generating Avoidance Motion Using Motion Graph. In *International Conference on Motion in Games 2011 (Lecture Notes in Computer Science 7060)*. 20–131.

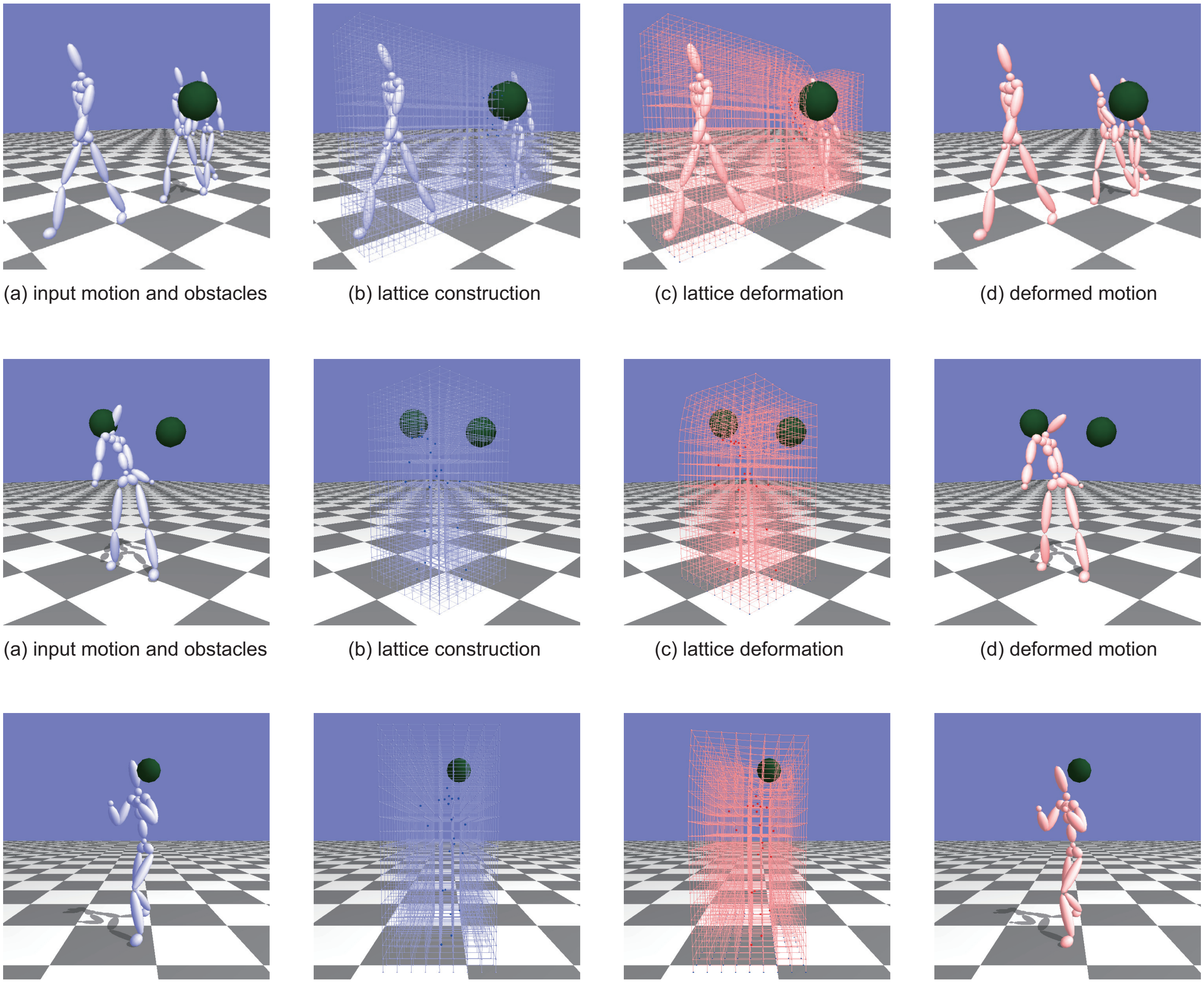
Ari Shapiro, Marcelo Kallmann, and Petros Faloutsos. 2007. Interactive Motion Correction and Object Manipulation. In *Symposium on Interactive 3D graphics and games (I3D) 2007*. 137–144.

Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible surface modeling. In *Eurographics symposium on Geometry processing 2007*. 109–116.

Kevin Wampler, Erik Andersen, Evan Herbst, Yongjoon Lee, and Zoran Popović. 2010. Character Animation in Two-Player Adversarial games. *ACM Transactions on Graphics (SIGGRAPH 2011)* 29, 3 (2010), Article No. 26.

Xiaolin K. Wei and Jinxiang Chai. 2011. Intuitive Interactive Human Character Posing with Millions of Example Poses. *IEEE Computer Graphics and Applications* 31, 4 (2011), 78–88.

Andrew Witkin and Michael Kass. 1988. Spacetime Constraints. *ACM Transactions on Graphics (SIGGRAPH '88 Proceedings)* 27, 3 (1988), 159–168.



(a) input motion and obstacles (b) lattice construction (c) lattice deformation (d) deformed motion

(a) input motion and obstacles (b) lattice construction (c) lattice deformation (d) deformed motion

(a) input motion and obstacles (b) lattice construction (c) lattice deformation (d) deformed motion

Figure 4: Experimental results of motion deformation. (Top) Walking motion and a still obstacle. (Middle) Twisting motion and two moving obstacles. (Bottom) Punching motion and a moving obstacle.

Andrew Witkin and Zoran Popović. 1995. Motion warping. In *SIGGRAPH '95 Proceedings*. 105–108.

Katsu Yamane, James J. Kuffner, and Jessica K. Hodgins. 2004. Synthesizing animations of human manipulation tasks. *ACM Transactions on Graphics (SIGGRAPH 2004)* 23, 3 (2004), 532–539.

Jianmin Zhao and Norman I. Badler. 1994. Inverse Kinematics Positioning Using Nonlinear Programming for Highly Articulated Figures. *ACM Transactions on Graphics* 13, 3 (1994), 313–336.

Victor Zordan, Adriano Macchietto, Jose Medin, Marc Soriano, Chun-Chih Wu, Ronald Metoyer, and Robert Rose. 2007. Anticipation from Example. In *ACM Virtual Reality Software and Technology 2007*. 81–84.