Relationship Descriptors for Interactive Motion Adaptation

Rami Ali Al-Asqhar* Taku Komura† Myung Geol Choi‡

University of Edinburgh

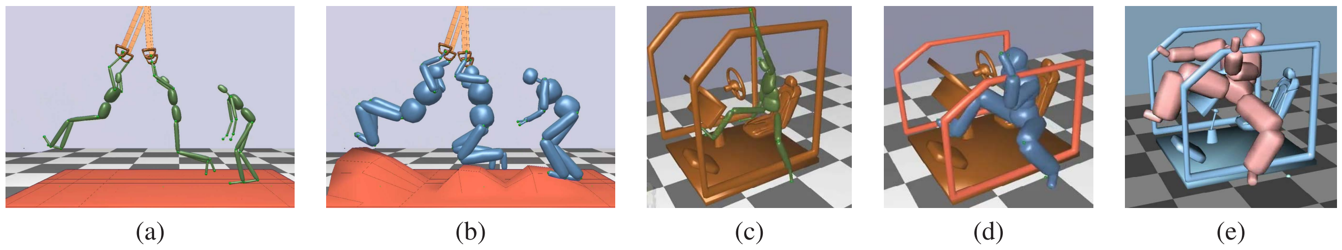


Figure 1: *(a) An original swinging motion (b) adapted to a deformed terrain and enlarged body. (c) A motion to ride on a car (d) adapted to a larger character and a smaller entrance. (e) Failure case when using a sparse set of positional constraints and repulsion in the direction of normal vectors.*

Abstract

This paper presents an interactive motion adaptation scheme for close interactions between skeletal characters and mesh structures, such as moving through restricted environments, and manipulating objects. This is achieved through a new spatial relationship-based representation, which describes the kinematics of the body parts by the weighted sum of translation vectors relative to points selectively sampled over the surfaces of the mesh structures. In contrast to previous discrete representations that either only handle static spatial relationships, or require offline, costly optimization processes, our continuous framework smoothly adapts the motion of a character to large updates of the mesh structures and character morphologies on-the-fly, while preserving the original context of the scene. The experimental results show that our method can be used for a wide range of applications, including motion retargeting, interactive character control and deformation transfer for scenes that involve close interactions. Our framework is useful for artists who need to design animated scenes interactively, and modern computer games that allow users to design their own characters, objects and environments.

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

Keywords: motion adaptation, motion retargeting, motion editing

1 Introduction

There is a strong demand in the gaming industry for an online motion adaptation scheme that can handle close interactions between multiple characters, a character and an object or a character and its environment. Modern computer games may include a wide range of environments and character designs in the package. They even allow players to design their own characters, and the environments may be procedurally generated. The characters may need to move through restricted environments, manipulate tools, and interact with other characters never encountered before. Preparing the motion data for all combinations of character morphologies and surface geometries can significantly increase the amount of data in the game package. Thus, recycling motion of close interactions for different characters and objects is convenient for the game developers.

Existing techniques of motion retargeting and adaptation either can only handle a limited variation of close interactions or require high computational cost. Some methods adapt motions by imposing a sparse set of positional [Gleicher 1998; Lee and Shin 1999; Choi and Ko 2000] and distance constraints [Liu et al. 2006]. As we present later in this paper, they can suffer from collisions and discontinuities when applied to scenes of close interactions. In order to handle close interactions, denser representations based on minimal spanning trees [Zhou et al. 2010] and volumetric meshes [Ho et al. 2010] are proposed. These methods require costly iteration processes. Also due to their discrete representations, they require using spacetime optimization when applied to scenes where the spatial relationships dynamically change overtime.

In this paper, we overcome the problems of existing techniques by proposing a new descriptor for representing the spatial relationship between a character and the mesh structures that it interacts with during the animation. With this representation, the pose of each body part is described by the weighted sum of relative translation vectors with respect to points sampled on the surface of the object in close proximity. The descriptors are sampled such that the movement with the same context can be reproduced by the summation of a small number of relative vectors, with little artefacts.

Our interactive framework keeps the movements of the characters to be continuous between frames even when there are significant deformations of the environment or large updates in the character morphology. In order to achieve this, first, we use the same set of descriptors in all the frames. Instead of using a temporal window as done in [Ho et al. 2010], we apply a spatial window that defines the volume that the descriptors are used to recompute the target joint position. Next, during runtime, the posture of the character is computed using a fast motion warping scheme, which pulls the body joints toward their target locations defined by the relationship-

*s0814395@sms.ed.ac.uk
†tkomura@ed.ac.uk
‡myunggeol@gmail.com

based descriptors while satisfying constraints. This framework can quickly reproduce postures that follow the original context.

We show examples in which characters adapt their movements to the geometric changes of objects and the environment, such as a character walking over a deformed terrain and a character adapting its posture to a re-shaped object that it is interacting with. We also show examples of retargeting movements of close interactions to characters of different morphologies and a cloth reconfigured to a shape that it is covering. The method is especially useful for interactive applications such as computer games as the motion reconstruction can be done quickly and robustly. The method is simple and easy to implement, and also produces plausible human motion, making it highly practical for synthesizing and editing close interactions.

Contributions

- A relationship-based representation for character poses using descriptors sampled in the environment.
- An online framework to adapt the movements of characters to large updates in the environment and character morphologies while preserving the context of the original scene and continuity of movements.
- A scheme to adaptively sample the relationship descriptors according to the original animation such that the motion can be reconstructed from a limited number of samples.

2 Related Work

For synthesizing movements where a character interacts with other objects or characters, most existing methods use a sparse set of positional and distance constraints to enforce a spatial relationship between characters and the environment. A few more recent works on character animation and shape modeling consider implicit spatial relationships.

Constraint-based motion synthesis Combination of optimization and, a sparse set of positional and distance constraints is a scheme that has been adopted for physically-based animation [Popović and Witkin 1999; Liu and Popović' 2002; Fang and Pollard 2003; Bai et al. 2012; Liu 2009], motion editing [Callennec and Boulic 2004], motion retargeting [Gleicher 1998; Lee and Shin 1999; Choi and Ko 2000], and adaptation to the environment [Choi et al. 2011]. For online character animation, the optimization must be done per-frame [Choi and Ko 2000] or using a short temporal window [Liu 2009]. In order to cope with the discontinuities caused by discretely turning on and off the constraints (such as the support foot constrained on to the ground), Shin et al. [2001] propose to adjust the weights of the constraints according to the Euclidean distance to the target location.

However, methods based on a sparse set of positional and distance constraints can result in collisions and penetrations of the body parts with the environment, objects, and even with the body itself. Especially when the interaction involves concavity, such collisions cannot be resolved by simply adjusting the trajectories in the direction of normal vectors (as done in [Lyard and Magnenat-Thalmann 2008]), as they may split the trajectories in the temporal domain. The trajectories need to be replanned using global path planners (as done in [Shapiro et al. 2007]), which is not suitable for online character animation.

In fact, during close interactions, the body is implicitly constrained by many factors including the context of the movements and obstacles that the body does not contact in the original motion. In order to adapt and retarget movements that involve close interactions of the body and mesh structures, it is important to encode such implicit spatial relationships in the representation.

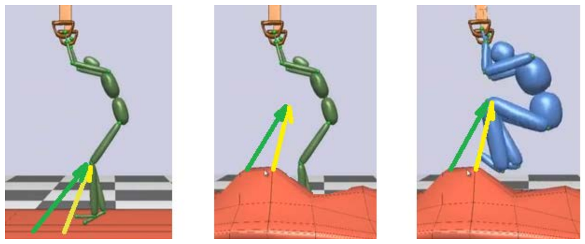


Figure 2: *The overview of the method: The position of the joints are represented by the weighted sum of relative vectors from descriptors computed in the original motion (left). The target location of the joints are computed from the updated geometry of the environment (middle). The motion adapts to the new geometry of the environment and morphology of the character (right).*

Character animation by spatial relationships There have been a few recent works which take into account the implicit spatial relationships of interacting characters and objects when synthesizing new animation. Kwon et al. [2008] handle the spatial relationships between characters in group motions by encoding the neighborhood formations and individual trajectories as Laplacian coordinates. When editing the trajectories, the relative spatial arrangements of characters are preserved by applying Laplacian mesh editing techniques [Alexa 2003; Sorkine et al. 2004]. Ho et al. [2010] apply the same idea to each body parts of articulated characters and preserve the context of the interactions during the animation. Zhou et al. [2010] represent the spatial relationships between multiple objects by a minimum spanning tree of Euclidean distance. As these are discrete descriptors whose topology change per configuration, they require offline spacetime optimization to adapt movements whose spatial relationship change over time, which make them unsuitable for real-time applications such as computer games and interactive motion editing.

Mordatch et al. [2012] introduces an auxiliary function to bridge the gap between contact and non-contact states for synthesizing novel movements. Their approach can produce a novel context from scratch using an objective function by an offline optimization process. How their descriptors can be applied for adapting movements to different geometry has not yet been explored, which is the main topic our research.

Relationship-aware shape modeling Some work of shape modeling takes into account the relationships between different objects for the modeling purpose. Harmon et al. [2011] defines an energy of overlapping and amend the shapes such that this energy is minimized. Similar to positional constraints, the spatial relationships are not encoded unless a contact occurs. Brouet et al. [2012] propose a scheme for adapting clothes to characters of different sizes. They also use relative vectors with respect to the bones of the body to define the location of the cloth particles. Their approach is for obtaining a static garment that fits the size of a new character rather than transferring an animation that involves dynamic movements of deformable meshes to characters of different geometry and morphology. Our representation is similar to Wrapdeformers [2013], which is a function in Maya that let users associate the deformer object with an influence object manually. In contrast, we automatically associate the character motion to the mesh structure

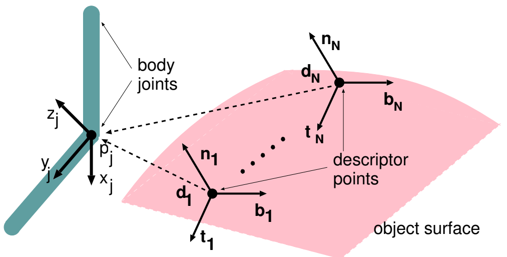


Figure 3: *The local coordinates defined at the body joints and at the sample points on the object.*

through the analysis of the original animation and selective sampling of the descriptors.

3 Overview

The overview of the scheme is shown in Fig. 2. Our method is composed of the preprocessing and run-time stages.

In the preprocessing stage, the original motion and geometry data of the characters and objects are loaded. The system scans through the animation to convert the scene into our representation, in which the poses of the body parts are described by the weighted sum of relative translations from points called descriptor points sampled on the mesh structures that compose the environment or the objects that the character interacts with (see Fig. 2 left, Section 4). The candidate of the descriptor points are computed by projecting the joint and end effector positions onto the surface of the objects in all the frames (see Section 5), but are only adopted according to our novel scheme specialized for close interactions (see Section 6).

During run-time, the geometries of the objects or environment and the morphology of the environment are updated (see Fig. 2 middle) and then the posture of the character is recomputed per frame using a motion warping scheme which moves the body joints toward the target locations defined by the descriptors while taking into account the bone-length and additional constraints (see Fig. 2 right, Section 7).

4 Relationship Descriptors

In this section, we introduce a novel approach to represent the movements of the character relative to the geometry of the object that it is interacting with. This representation is especially useful for reproducing animation with the same context even when the geometry of the object is changed. In our representation, the joint positions are computed by the relative translations from a static set of points called descriptor points. The descriptor points are sampled on the surface of the mesh structure by analyzing the interaction in all the frames (see Section 5 and Section 6 for the sampling scheme).

Now, we explain how to compute the joint positions from the descriptor points. Let us define the position of joint j by $\mathbf{p_j}$, and the descriptor points by $(\mathbf{d}_1, ..., \mathbf{d}_N)$ (see Fig. 3). We also obtain the normal, tangent and binormal vectors from the geometry of the surface, which are defined by $(\mathbf{n}_1, ..., \mathbf{n}_N)$, $(\mathbf{t}_1, ..., \mathbf{t}_N)$, and $(\mathbf{b}_1, ..., \mathbf{b}_N)$, respectively. The tangent vectors are computed by simply picking one of the edges connected to the vertex and projecting it to the tangent plane, and the binormal vectors are computed by the cross product of the normal and tangent vectors. Given a motion, we represent the joint positions $\mathbf{p_j}$ relative to $\mathbf{d_i}$ using these three vectors:

$$\mathbf{p_j} \quad = \quad \mathbf{d_i} + \alpha_{i,j}\mathbf{n_i} + \beta_{i,j}\mathbf{t_i} + \gamma_{i,j}\mathbf{b_i}. \tag{1}$$

As we want $\mathbf{p_j}$ to be influenced by not only one but all the descriptor points in proximity, we represent it as the weighted sum of Eq. (1) of all the descriptor points instead:

$$\mathbf{p_j} = \quad \sum_i^N w_{i,j}(\mathbf{d_i} + \alpha_{i,j}\mathbf{n_i} + \beta_{i,j}\mathbf{t_i} + \gamma_{i,j}\mathbf{b_i}) \tag{2}$$

where $w_{i,j}$ is the normalized weight between joint j and descriptor point $\mathbf{d_i}$ whose value is dependent on the distance between the two points and how much the normal vector $\mathbf{n_i}$ is facing towards $\mathbf{p_j}$. For computing the weights, we first calculate the following term for all the descriptor points:

$$w'_{i,j} = \frac{\mathbf{n_i} \cdot (\mathbf{p_j} - \mathbf{d_i})}{\|\mathbf{p_j} - \mathbf{d_i}\|}. \tag{3}$$

The weight fades out as the distance between $\mathbf{p_j}$ and $\mathbf{d_i}$ increases:

$$w''_{i,j} = w'_{i,j} f(\|\mathbf{p_j} - \mathbf{d_i}\|), \tag{4}$$

where

$$f(d) = \begin{cases} 0 \ (r_2^j \leq d) \\ 1 - \frac{d - r_1^j}{r_2^j - r_1^j} \ (r_1^j < d < r_2^j) \\ 1 \ (d \leq r_1^j), \end{cases} \tag{5}$$

r_1^j is the distance to the closest descriptor point and r_2^j is set to $r_1^j + \frac{1}{4} \times bodyheight$. Finally, we normalize the weights:

$$w_{i,j} = \frac{w''_{i,j}}{\sum_i w''_{i,j}}. \tag{6}$$

Using Eq. (2), the position of each body joint can be reconstructed from the geometry of the object surface. When the geometry of the object is changed, the updated poses of the body joints can be computed by feeding the mapped descriptor points and the axes of their local coordinates into these equations. More about the reconstruction process is explained in Section 7.

5 Sampling Proximity Points on Surfaces

In this section, we explain how we sample proximity points on the object surface, which are going to be candidates of the descriptor points $(\mathbf{d_1}, ..., \mathbf{d_N})$.

Proximity points are sampled on the object surface where the character closely interacts with. This is done by examining if a body joint is within the coverage of each triangle composing the object mesh, and then projecting the position of the body joint onto the surface of the triangle and computing its barycentric coordinates.

The definition of the triangle coverage and the way to check if a body joint lies within the coverage is as follows. First, we take each triangle of the object mesh and define a volume which is composed of the triangle and the planes connected by the outgoing normal vectors at the triangle vertices. We define the coverage of the triangle as the space which is contained within this volume (see Fig. 4). We test if the joint lies within the coverage of a triangle by a ray-casting test (casting a ray in arbitrary directions and checking the number of intersections with the boundary), which is often used in graphics for evaluating if a point is within a shadow volume. Note that a joint can be contained in multiple coverage, especially when a joint is in proximity of a concave object or multiple objects. This

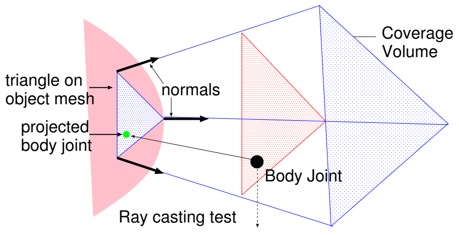


Figure 4: *The coverage of a triangle is defined by its normal vectors at the vertices. Coverage of the body joints are examined by a ray-casting test. Those inside are projected on the triangle and their barycentric coordinates are computed.*

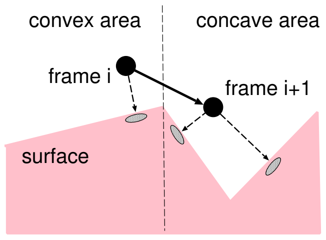


Figure 5: *The disadvantage of using a different set of descriptor points per frame: the number of descriptor points changes when the joint moves into a concave area.*

is important as we want the body joint to be associated to multiple surface segments of the object, such that the body posture is affected by all the descriptor points in proximity.

Once we know a joint is within the coverage of a triangle, we project it on the triangle by finding a point within the triangle whose normal vector defined by the barycentric coordinates penetrates the joint position in 3D space (the green point in Fig. 4). This is a difficult problem to solve as it brings rise to a system of non-linear equations with the barycentric coordinates of the projected point as the unknowns. We solve this using Newton's method with an initial guess at one of the triangle points. The candidate point on the surface triangle is projected back on a plane that is parallel to the surface triangle and passes the body joint (the red triangle in Fig. 4). The barycentric coordinates that make the back-projected point overlap with the joint is computed by iteratively decreasing the distance between them.

6 Computing Descriptor Points

In this section, we explain about the method to sample the descriptor points among the proximity points computed in Section 5. There can be two methods for describing the movements by the relationship descriptors; (1) using the proximity points computed per frame or (2) using a static set of descriptor points for the entire animation. We first describe the problem of the first approach and then describe our adaptive sampling scheme based on the second.

Problems of Per-Frame Set of Descriptor Points A naive approach is to change the set of descriptor points per frame by adopting the proximity points computed in each frame by the method explained in Section 4. In such a case, we can assume a body joint is reconstructed from proximity points sliding over the surface. A motion computed by this method, however, suffers from jerkiness when the geometry of the object is changed. This is due to the sudden increment or decrement of the proximity points when the joint moves into or comes out from a concave area of the surface. This is illustrated in Fig. 5: a joint that is in a convex area in frame i with only one proximity point enters a concave area in frame $i+1$ where the number of proximity points increases to two. When the geometry of the surface changes, the reconstructed position of the joint will be very different in the two frames as an additional descriptor point is involved in frame $i+1$. In fact, this is a problem common with applying discrete representations such as minimal spanning trees [Zhou et al. 2010] or Delaunay tetrahedralization [Ho et al. 2010]; the topological change of the trees / graphs between frames results in discontinuity and jerkiness of the motion. Mordatch et al. [2012] also evaluates the interaction between the fingers and the object by such a one-to-one projection. This limits their method to be applicable only to convex objects.

Another problem is that there is a lot of redundancy in the data because different sets of proximity points need to be saved for all the frames although many points are cluttered in close proximity. This often happens in actions such as sitting on a chair, in which the character does not move its body much. A lot of samples will be duplicated on the chair between frames.

Adaptively Sampling a Fixed Set of Descriptor Points To solve these problems, we use a fixed set of descriptor points throughout the animation, which are sparsely sampled on the surface of the mesh. We apply an incremental sampling scheme, passing as an argument a geodesic distance radius parameter, whose value dynamically changes according to the distance between the joint and the sample point on the surface:

$$r = \begin{cases} 0.5 \times d \text{ (if } d > d_0 \text{)} \\ 0.5 \times d_0 \text{ (otherwise)} \end{cases} \tag{7}$$

where d_0 is the threshold for a minimum radius, which is set to $0.05 \times bodyheight$. This value is tunable according to the task and context. When a new proximity point is sampled on the surface of the mesh, the previous samples in the geodesic radius are examined (see Fig. 6). If there are no previous samples in the radius, this sample is adopted as the new descriptor. For fine interactions such as the hand manipulating objects, the number of sample points will be larger, and for those such as avoiding obstacles when walking, it will be smaller.

The efficiency of this sampling scheme comes from its biased and incremental nature. First, it produces more samples where the character closely interacts, which is needed for achieving precise control. It also helps to avoid collisions while preserving close distances where the body is in close contact with the environment. Second, as samples are produced incrementally, we can easily add new movements into the data while making use of the previously

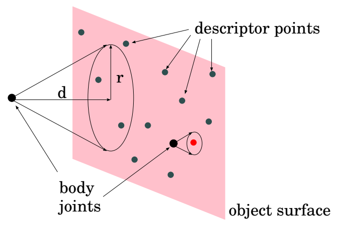


Figure 6: *The adaptive radius scheme to sample descriptor points: The radius scales with the distance. When there is no previous descriptor points sampled in the radius, a new descriptor point is placed. Only a single descriptor point will be sampled in the right case.*

produced samples. This is useful for a commonly used environment such as a room. For offline schemes such as K-means clustering, it is difficult to handle a long animation sequence due to its poor scalability. The clusters also need to be recomputed when a new set of data are added to the original set.

The idea behind using geodesic distance for the radius is that we prefer to cluster the points based on the distance that the character needs to travel along the surface between the sample points. We do not want the motion to be affected by, for example, edits on the other side of the wall that the character is in close proximity, which can happen if we use a radius based on Euclidean distance.

7 Adapting Postures to New Geometries

Now we explain about our motion warping scheme that recomputes the posture of the character when the geometry of the mesh structures is updated. We first explain about recomputing the joint positions by the relationship descriptors according to the geometric deformation of the objects or environment, and then about the motion warping scheme that finalizes the character posture.

7.1 Motion Retargeting by the Relationship Descriptors

Once the position of the descriptor points and their normal, tangent and binormal vectors are re-mapped, the positions of the joints can be computed using Eq. (2). These are going to be the desired locations of the joints.

For object shape deformation, in addition to affine transformations such as translation, rotation and scaling, we also allow the users to insert skeletons and edit the mesh by linear blending. The descriptor points are mapped to the deformed geometry.

We also show examples of switching the geometry of the objects. The corresponding descriptor points and their local coordinates between different geometries can be obtained by computing the dense correspondence between the original and new object. Computing the dense correspondence is itself a research topic and a universal solution does not exist; it is even more difficult for objects such as chairs whose topologies can be different. In our experiments, we use a rather simple approach based on finding the shortest distance point on two objects after a gross manual alignment. This method works fine as the descriptors are only sparsely distributed over the object.

7.2 Spatial Relationship-aware Motion Warping

Here we propose a scheme to warp the movements of the characters such that they reach the desired joint positions computed by the relationship descriptors as much as possible, while satisfying other constraints such as the bone length and positional constraints.

The key to our relationship-aware motion warping scheme is the introduction of the affinity concept: this is a measure of the translational flexibility of a joint. We prefer the joints that are in close proximity of the surface to move less, while those which are farther away from it to move more freely. This is due to the fact that joints near the surface tend to make contacts with the surface, or need to be carefully controlled to avoid collisions. The affinity values are computed by summing the weights of the associated descriptors on the surface computed in Eq. (6) and normalizing them:

$$s_j = \frac{\sum_i^N w_{i,j}}{\sum_j^{N_j} \sum_i^N w_{i,j}} \tag{8}$$

where j is index of the joints and N_j is the number of joints.

In the rest of this section, we explain the procedure of our relationship-aware motion warping scheme. In order to satisfy the constraints, we use a non-linear inverse kinematics solver [Jakobsen 2001] that can robustly compute a solution with only a small number of iterations, despite large deformations of the mesh structures or updates in the character morphology. The **force accumulation** and **integration** steps are run first, and then the **constraints** step is iterated N_c times (set to 3 in our experiments). The entire process is repeated N_s times (set to 3 in our experiments) for each time step, which means force accumulation and integration steps are run N_s times, and the constraint step is run $N_s \times N_c$ times in every frame.

Force Accumulation Step: Instead of explicitly manipulating the joints, we control them by applying virtual forces to the particles that correspond to the joints. The forces are computed by multiplying an elastic constant to the difference between their current and target positions:

$$\mathbf{f}_j = k(\mathbf{p_j^{tar}} - \mathbf{p_j^{cur}}) + \mathbf{f^{ext}}. \tag{9}$$

where k is an elasticity constant that is set to 1, $\mathbf{p_j^{tar}}$ is the target position of the joint computed using the relationship descriptors by Eq. (2), $\mathbf{p_j^{cur}}$ is the current joint position, and $\mathbf{f^{ext}}$ is the external force that is added if an extra effect such as the wind blowing the body needs to be applied.

Integration Step: We apply the the virtual forces computed by Eq. (9) to the joints using Verlet integration:

$$\mathbf{p_j^{new}} = \mathbf{p_j^{cur}} + d(\mathbf{p_j^{cur}} - \mathbf{p_j^{prev}}) + \frac{1}{2}\mathbf{f_j}\frac{1}{N_s^2} \tag{10}$$

where $\mathbf{p_j^{prev}}$ is the position of the joint in the previous iteration and d is a coefficient that is added to reduce the wobbling effect whose value is set linear to the joint's affinity value ($d = 0.8$ when $s_j = 0$ and $d = 0.2$ when $s_j = 1$).

Constraints Step: Using the updated particle positions $\mathbf{p_j^{new}}$, we compute the final positions of the joints that satisfy the bone-length, positional and collision constraints by iteratively updating the particle positions until the errors of all the constraints are below a certain threshold. To satisfy the bone-length constraints, the positions of each particle is updated by the following equation:

$$\Delta\mathbf{p_j} = \frac{s_k}{s_k + s_j}\frac{\mathbf{p_j} - \mathbf{p_k}}{\|\mathbf{p_j} - \mathbf{p_k}\|}(l^0 - \|\mathbf{p_j} - \mathbf{p_k}\|) \tag{11}$$

where $\mathbf{p_k}$ is a particle that is connected to joint j by a bone, and l^0 is the length of the bone. This will result in joints with large affinity to move less and small affinity to move more. For positional constraints, we simply move the particle to the target location. For collision constraints, the particles are moved towards the normal direction of the closest polygon for bone-mesh collisions. We did not address bone-bone collisions here in the interest of performance, though we have a good-performing hack solution that is explained in the experiment section.

The bone-length error converges quickly over iterations and most are satisfied in our experimental results within the constant number of iterations that we adopt. There can be cases that the constraints cannot be fully satisfied due to extreme re-scaling of the character sizes. Such error can be monitored and the system can inform the user.

This framework produces continuous movements between frames even under large deformation of the environment and updates in the morphology due to the continuity of target joint positions. When adapting the movements, we first run the above process statically in the first frame. Then, during the animation, we use the joint locations in the previous frame as the initial posture in Eq. (9) and (10).

8 Implementation

We first describe which data are computed on/off-line for minimizing the memory consumption while achieving interactive performance. Next we explain about the character structure that we use in our experiments.

Online Computation of Parameters In our implementation, the only data that is saved in the offline process is the triangle IDs and the barycentric (UV) coordinates of the descriptor points in the triangles (see Section 6). These values are fixed throughout the animation and therefore the memory-overhead per frame is not very high. The rest of the data required for reconstructing the character postures, such as the coefficients of the normal, binormal and tangent vectors in Eq. (2), the weights of the relative vectors computed by Eq. (6) as well as the distance between the joints and the descriptor points in Eq. (5), are all computed on-the-fly by referring to the original motion data. We take this approach as the amount of memory for saving such data is large. The entire computation still fits into a interactive frame-rate for character animation.

Character Structure The character structure depends on the motion data that we use, but are mostly composed of 38 body segments, and each joint has 3DOF rotation. Additional bones are inserted to preserve the rigidity of segments, such as between the left and right hips, as done in [Jakobsen 2001]. Joint angle limits of hinge joints are imposed in a similar way when they reach the minimal and maximal angles.

9 Experimental Results

We present examples of our approach applied to different scenarios, including deforming the geometry of the environment, retargeting movements in which two characters are closely interacting, and finally deformation transfer in which a character surface dynamically changes its spatial relationship with a garment. The computational cost is discussed at the end of this section. For details of the animations, the readers are referred to the supplementary video.

Updating the Geometry of Environments We present three experiments in which we deform environment geometry whilst preserving the context of a scene. In all the examples, we also present results in which character morphologies are changed.

The first example is based on a walking motion from the CMU motion database [Gross and Shi 2001]. In this scene, the input is a motion of a character walking across a bridge which has obstacles above it. The bridge is then deformed dynamically, and duplicated characters walking in different phases adapt their movements to the deformation. (see Fig. 8,(a)).

We next show an example in which a character interacts with a car. The character first enters on the car, grasps various locations inside the car including the the steering wheel, the gear lever, and a mobile phone laying near the feet area. The relationship descriptors are computed in the environment for the entire series of movements by processing them in the sequence described above. Despite the large deformation of the car (scaling the size of the entrance, changing the location of the seat and steering wheel and replacing the seat with a different geometry) and updates of the morphology, plausible movements are computed (see Fig. 1 (c)(d), Fig. 8 (b)).

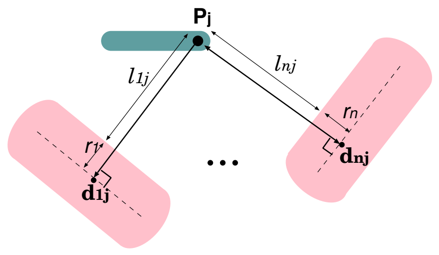


Figure 7: *The bone configuration represented by the weighted sum of relative vectors defined at other bones.*

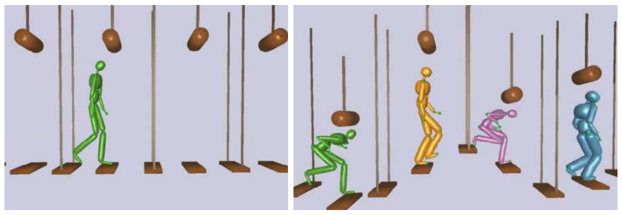
We also show an example in which the motion of riding on the car is retargeted to a larger character using a standard retargeting scheme based on positional constraints and temporal smoothing [Lee and Shin 1999] (see Fig. 1(e)). As the body simply tries to keep the joint angles similar to the original motion for parts that does not involve positional constraints, the body is sometimes repulsed to the opposite side of the obstacle, resulting in discontinuities between frames. Indeed, it can be observed that the trajectories of the body parts need to be replanned such that the body parts do not collide with the obstacles and also stays on the same side of the objects throughout the motion.

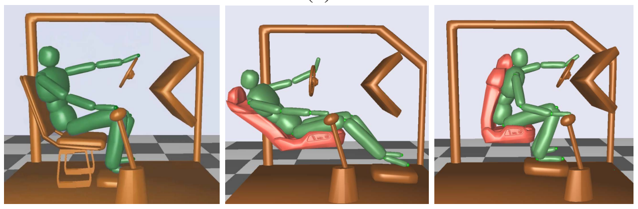
Finally, we show another example from the CMU motion database in which the character jumps onto swinging bars, swings a few times and lands on the ground. The motion is retargeted whilst the ground geometry is deformed and while the user interactively drags the bars (see Fig. 1(a),(b)). This example shows that our approach is even applicable to fast ballistic movements. Although the method does not preserve momentum, the results appear visually plausible thanks to the quasi-physics nature of the motion warping scheme.

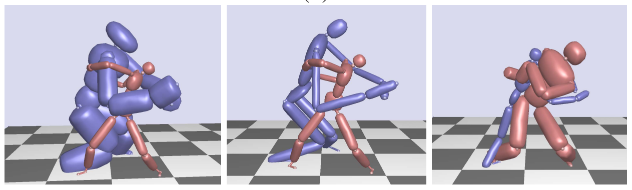
Motion Retargeting of Interactions Between Multiple Characters Multi-character interactions can be achieved in mostly the same way as interactions between characters and mesh structures with slight modifications. Same as Eq. (2), the configuration of a bone j is represented by the weighted sum of relative vectors originating from descriptor points $\mathbf{d_i}$ defined in the rest of the bones $i = 1, ..., n$ (see Fig. 7). The descriptor points $\mathbf{d_i}$ are updated per frame by projecting each joint position to the closest point on all the other bone segments. We do not suffer from the discontinuity problem explained in Section 6 because there are no concavity in the bone segments, and we will always have a descriptor per segment which slides continuously along the surface over time. The recomputation of the descriptor points per frame eases the collision avoidance between bones represented by capsules. The norm of the relative vector $\mathbf{p_j} - \mathbf{d_i}$ can be decomposed into the distance between $\mathbf{p_j}$ and the capsule surface of bone i, and its radius:

$$\sqrt{\alpha_{i,j}^2 + \beta_{i,j}^2 + \gamma_{i,j}^2} = r_i + l_{ij}, \tag{12}$$

where $(\alpha_{i,j}, \beta_{i,j}, \gamma_{i,j})$ is the relative position of joint j with respect to bone i under the same context in Eq. (2), r_i is the capsule radius of bone i, and l_{ij} is the distance between $\mathbf{p_j}$ and the capsule surface of bone i. When the character sizes are updated and the radii of the bones are changed, $(\alpha_{i,j}, \beta_{i,j}, \gamma_{i,j})$ is rescaled such that its norm satisfies Eq. (12). The joints will try to keep a distance of l_{ij} between their positions and the capsules of the other bones. They will avoid penetrating each other without much extra computation in this way.


(a)


(b)


(c)

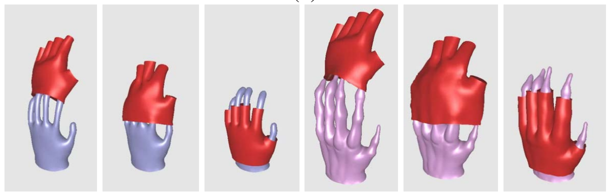

(d)

Figure 8: *(a) Adapting a motion to cross a bridge to ones crossing a dynamically deforming bridge. (b) Adapting movements inside a car to different designs. (c) Retargeting a judo motion to characters of different sizes. (d) An original animation of putting on a glove (left) adapted to a hand with a different geometry (right).*

During the motion warping phase, the postures of both characters are simultaneously updated such that both characters' postures converge to a point in which they each become compliant to the other character's posture. In each iteration, we gradually reduce the size of the spatial window such that all the joint relationships are considered in the beginning and only local spatial relationships are considered in the latter iterations. As we have only three iterations in our implementation, we start from a radius that considers all pair of relationships in the first iteration, with a constant weight of 1 in Eq. (2), and then we half the radius in the second iteration, and finally use only the fade-off window in Eq. (5) in the final iteration. We find this approach works better for close character interactions as both the global and local relations are preserved.

Snapshots of retargeting a judo motion to characters of different sizes are shown in Fig. 8(c). We do not only change the length of the bones but also the radius of the capsules to retarget the motion to fat characters. The body sizes can even be dynamically edited during runtime.

Dynamic Garment Transfer Our method can also be applied for adapting the configurations of deformable objects such as clothes. By applying the same method to the vertices of the cloth as explained for the body joints, the geometry of the cloth can be adapted to the deformation of the underlying object. Thus, we can transfer deformable objects between scenes, in which they dynamically change their spatial relationships with other objects. The deformable object adopts new proportions relative to the new surface to which it's adapting, as we do not preserve the original proportions.

The garment motion of a glove being worn on a human hand is transferred to another hand with completely different size and geometry (see Fig. 8,(d)). Transfer of such deformation by simply re-running a physical simulation under the new condition is difficult due to the passive nature of garments.

Computational Costs The computational time during the offline and online process for each of the above examples are presented in Table 1. We do not impose bone length constraints in the cloth example. The judo examples does not use any polygon models and do not require any offline processing as the samples are all at the bones; they are computed on-the-fly per frame. The online cost is short enough to be applied for interactive applications. The experiments are run on one core of a Core i7 2.67GHz CPU with 2GB of memory. Since most of the costly computation such as sampling points, computing the local coordinates at such points and the weights associated to them are highly parallelizable, speed enhancement can be expected with GPU implementations.

10 Conclusions and Discussions

In this paper, we have presented a practical method for online editing and adaptation of motions that involves close interactions between characters and objects. One major advantage of our approach is that we localize the adaptation scheme to each frame level, such that the postures of the characters can be computed on-the-fly subject to the updated geometries of the meshes and the morphologies of the characters. This is the main difference from previous approaches that use spacetime optimization [Gleicher 1998; Ho et al. 2010], or multiresolution methods [Lee and Shin 1999]. This is achieved by setting up the relationship-based descriptors such that they result in continuous joint target positions between frames.

Although our approach can produce plausible results for computer games and animations, the edited movements do not fully satisfy the laws of physics. For example, in the swinging demo, if the motion of the bars are abruptly edited while the character jumps onto the bars, the body will move in the air and the motion will not preserve angular and linear momenta. Despite the fact our framework can only produce quasi-physical effects, the idea of our relation descriptors is general enough to be applied for physically-based animation and robotics. For example, an interesting possibility is to produce a PD servo such as [Yin et al. 2007; Liu et al. 2010] based on our relation descriptors. In such a case, the character will exert torque such that the body parts reach the target positions rep-

Table 1: *The time required for the computation: #p : number of polygons, #s number of descriptors, fps_{off} / fps_{on} : offline/online frames per second, error : average error rate of the bone length constraint.* [*] *The descriptor for character interactions are recomputed per frame.*

Scene	#p	#s	fps_{off}	fps_{on}	error
bridge	468	399	25	34	0.14%
car	6005	389	13	33	0.36%
swing	1364	400	19	32	0.57%
judo	n/a	2070	n/a[*]	29	0.12 %
cloth	11241	1300	3	4	n/a

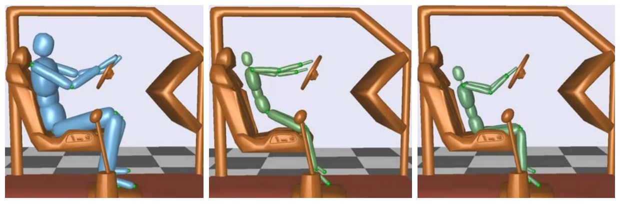


Figure 9: *(left) The original driving, (middle) retargeted to a smaller character, and (right) fixed motion by reducing the affinity values due to the seat-back.*

resented by the relation descriptors. This is the same for methods based on spacetime constraints, such as [Liu and Popović' 2002]; the postures represented by the relation descriptors can be used as constraints and a motion that spatiotemporally satisfies such constraints can be computed by optimization. A simple but possibly effective approach to produce movements that satisfy laws of physics is to start from the movements computed by our scheme and further apply physical filters such as [Yamane et al. 2010; Shin et al. 2003] to convert them into physically plausible ones.

Limitations: Our method fails in cases where the character is scaled down too much when it is surrounded by many surface descriptors in close proximity, such as during a car driving motion. This will cause certain parts of the body to float in the air due to the almost equally large affinity values of all the joints. In the car driving demo, when the character is scaled down, the limbs out-stretch as we attempt to preserve the spatial relations whilst constraining bone-length, and the character's bottom begins to float unnaturally losing contact with the seat. The user has a number of options to solve this. One option is to apply positional constraints to enforce certain contacts. Another solution is to manually adjust the influence of certain descriptors. In the driving example, the floating effect can be removed by reducing the influence of the seat-back (see Fig. 9). Finally, the user may manually adjust joint affinity values. In the car-driving case the user may apply a maximum affinity value to the bottom of the character such that all IK occurs around the hip whilst the hip spatial relations are most strongly enforced.

Our analysis of the scene is simple and may not be suitable in some situations; for example, passing near by an obstacle does not mean one always wants to pass close by the obstacle. In such a case, the walking motion can be undesirably edited when the obstacle is moved away from the body. Adaptively changing the weights of the descriptors according to the edits can be a solution to this problem.

We currently do not make use of the target orientation; the orientation of the joints are computed by simply applying SLERP at the joints such that they reach their target locations. Therefore, we cannot produce effects such as twisting the hand. Such effects can be produced by introducing a rotation parameter around the bone axis and adding virtual torques to the bones by PD control as done in Eq. (9).

Future Works We are interested in using these descriptors for synthesizing novel animation such as achieved in [Mordatch et al. 2012]. We are also interested in learning movements in our representation. This can be an interesting direction because interactions learned by observation can be applied in a broader range due to the adaptability of the scheme.

Acknowledgement

We thank the anonymous reviewers for their constructive comments. This work is partially supported by grants from EPSRC (EP/H012338/1) and EU FP7 TOMSY.

References

ALEXA, M. 2003. Differential coordinates for local mesh morphing and deformation. *The Visual Computer 19*, 2-3, 105–114.

BAI, Y., SIU, K., AND LIU, C. K. 2012. Synthesis of concurrent object manipulation tasks. *ACM Transactions on Graphics 31*, 6.

BROUET, R., SHEFFER, A., BOISSIEUX, L., AND CANI, M.-P. 2012. Design preserving garment transfer. *ACM Transactions on Graphics 31*, 4, 36:1–36:11.

CALLENNEC, B. L., AND BOULIC, R. 2004. Interactive motion deformation with prioritized constraints. *Proceedings of ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 163–171.

CHOI, K.-J., AND KO, H.-S. 2000. Online motion retargeting. *Journal of Visualization and Computer Animation 11*, 5.

CHOI, M. G., KIM, M., HYUN, K., AND LEE, J. 2011. Deformable motion: Squeezing into cluttered environments. *Computer Graphics Forum 30*, 2, 445–453.

FANG, A. C., AND POLLARD, N. S. 2003. Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics 22*, 3, 417–426.

GLEICHER, M. 1998. Retargetting motion to new characters. *Proceedings of SIGGRAPH*, 33–42.

GROSS, R., AND SHI, J. 2001. The CMU Motion of Body (MOBO) Database. *Robotics Institute, Carnegie Mellon University*, CMU-RI-TR-01-18.

HARMON, D., PANOZZO, D., SORKINE, O., AND ZORIN, D. 2011. Interference aware geometric modeling. *ACM Transactions on Graphics 30*, 6.

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

JAKOBSEN, T. 2001. Advanced character physics. *In Game Developers Conference Proceedings*, 383–401.

KWON, T., LEE, K. H., LEE, J., AND TAKAHASHI, S. 2008. Group motion editing. *ACM Transactions on Graphics 27*, 3.

LEE, J., AND SHIN, S. Y. 1999. A hierarchical approach to interactive motion editing for human-like figures. *Proceedings of SIGGRAPH '99*, 39–48.

LIU, C. K., AND POPOVIĆ', Z. 2002. Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics 21*, 3, 408–416.

LIU, C. K., HERTZMANN, A., AND POPOVIĆ, Z. 2006. Composition of complex optimal multi-character motions. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

LIU, L., YIN, K., VAN DE PANNE, M., SHAO, T., AND XU, W. 2010. Sampling-based contact-rich motion control. *ACM Transactions on Graphics 29*, 4.

LIU, C. K. 2009. Dextrous manipulation from a grasping pose. *ACM Transactions on Graphics 28*, 3.

LYARD, E., AND MAGNENAT-THALMANN, N. 2008. Motion adaptation based on character shape. *Computer Animation and Virtual Worlds 19*, 3-4, 189–198.

MORDATCH, I., POPOVIC, Z., AND TODOROV, E. 2012. Contact-invariant optimization for hand manipulation. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

POPOVIĆ, Z., AND WITKIN, A. 1999. Physically based motion transformation. *Proceedings of SIGGRAPH '99*, 11–20.

SHAPIRO, A., KALLMANN, M., AND FALOUTSOS, P. 2007. Interactive motion correction and object manipulation. *Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*.

SHIN, H. J., LEE, J., SHIN, S. Y., AND GLEICHER, M. 2001. Computer puppetry: An importance-based approach. *ACM Transactions on Graphics 20*, 2, 67–94.

SHIN, H., KOVAR, L., AND GLEICHER, M. 2003. Physical touch-up of human motions. *Proceedings of Pacific Graphics*.

SORKINE, O., LIPMAN, Y., COHEN-OR, D., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Proceedings of Symposium on Geometry Processing*, 179–188.

WRAPDEFORMERS. 2013. *Autodesk Maya 2013 User Guide*.

YAMANE, K., ARIKI, Y., AND HODGINS, J. K. 2010. Animating non-humanoid characters with human motion data. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 169–178.

YIN, K., LOKEN, K., AND VAN DE PANNE, M. 2007. Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics 26*, 3.

ZHOU, K., XU, W., TONG, Y., AND DESBRUN, M. 2010. Deformation transfer to multi-component objects. *Computer Graphics Forum 29*, 2.