THE UNIVERSITY
of EDINBURGH

Relationship Descriptors for Interactive Motion Adaptation

Rami Al-ashqar

Doctor of Philosophy

Institute of Perception, Action and Behaviour

School of Informatics

University of Edinburgh

2017

ABSTRACT

In this thesis we present an interactive motion adaptation scheme for close interactions between skeletal characters and mesh structures, such as navigating restricted environments and manipulating tools.

We propose a new spatial-relationship based representation to encode character-object interactions describing the kinematics of the body parts by the weighted sum of vectors relative to descriptor points selectively sampled over the scene. 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 deformations in the objects and character morphologies in real-time whilst preserving the original context and style of the scene.

We demonstrate the strength of working in our relationship-descriptor space in tackling the issue of motion editing under large environment deformations by integrating procedural animation techniques such as repositioning contacts in an interaction whilst preserving the context and style of the original animation.

Furthermore we propose a method that can be used to adapt animations from template objects to novel ones by solving for mappings between the two in our relationship-descriptor space effectively transferring an entire motion from one object to a new one of different geometry whilst ensuring continuity across all frames of the animation, as opposed to mapping static poses only as is traditionally achieved.

The experimental results show that our method can be used for a wide range of applications, including motion retargeting for dynamically changing scenes, multi-character interactions, and interactive character control and deformation transfer for scenes that involve close interactions. We further demonstrate a key use case in retargeting locomotion to uneven terrains and curving paths convincingly for bipeds and quadrupeds.

Our framework is useful for artists who need to design animated scenes interactively, and modern computer games that allow users to design their own virtual characters, objects and environments, such that they can recycle existing motion data for a large variety of different configurations without the need to manually reconfigure motion from scratch or store expensive combinations of animation in memory. Most importantly it’s achieved in real-time.

Lay Summary

Video games and films involve a lot of 3d animations between virtual characters and environments which involve close interactions such as two characters engaged in judo or a character manipulating a tool or navigating a constrained environment. These animations are often recorded through motion capture which is static and specific to the one scene configuration. What happens however if the artist decides to change the size of the character, change the proportions of the scene, or to switch the object the character is interacting with entirely, i.e. to a new tool to manipulate? The static animation will no longer conform to the new object or scene proportions and there will be collisions, penetrations etc. We need a way in which to add flexibility to the animations and make them dynamic such that they can adapt automatically to the new geometry, proportions and configurations of the deformed scenes or characters involved such that the context of the interaction is preserved and bypassing the need for any manual fix-up work. Furthermore we wish to achieve this adaptation in real time such that we can dynamically deform the scene whilst the character adapts. Our thesis proposes a representation to capture such interactions and then to interactively adapt animations to deformed scenes and character morphologies in real-time.

Acknowledgements

I’d like express my thanks and great appreciation to my supervisor Taku Komura for all his help, guidance and patience in seeing me through this phD and providing many opportunities for me along the journey as well as his help in the writing phase of the papers.

I’d like to also express my thanks and appreciation to my phD colleagues for their help, discussions and insight, in particular Myung Geol Choi who has helped me in my research by collecting motion capture data for my work and helping to compare my method against previous work.

I acknowledge the funding of this study provided by the TOMSY grant.

Finally I’d like to thank my family, particularly my father, for their help, patience and support, sometimes financial, in the course of my PhD journey.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

This thesis is presented in the ‘collection of papers’ format. I will be attaching four papers. Three of these papers are published. One is not published. Some of the papers involve collaboration, and for each I will clearly specify the extent of my contribution wherever collaboration is involved. For two of the papers I am the sole first author, for one I am a co-first author, and for another a key contributor, and I shall specify my contribution whilst acknowledging the contribution of collaborators for each paper in clear terms.

Signature _____ Rami Al-ashqar

Date: 26/07/2016

Table of Contents

Chapter 1

Introduction

1.1 Background

In this modern day and age, 3D digital characters are used in all kinds of applications from movies to video games, virtual reality, augmented reality, robotics and more. These characters are not static, they are animated and often interact with their environments. A lot these animations involve close interactions. Examples of this are multi-character interactions such as two characters engaged in judo, a character manipulating a tool i.e. handling a weapon or a character navigating a constrained environment such as a car. These interactions are often complex, tough to generate accurately and are sensitive to changes in the scene configurations and proportions. But there is a very high demand for them.

There is hence 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.

1.2 Problem Statement

One of the key problems with animations involving complex interactions is that they're difficult to put together. Acquisition of animations is costly. It often involves gathering lots of motion capture data for which sessions are costly to set up and take time. When it comes to interactions every animation must be precisely captured such that it conforms exactly to the object or other character for the interaction in question. If not, then manual animation fix-up work must be applied to the interacting characters and objects to get them to conform to one another. Hence there is a strong demand for motion adaptation schemes for close interactions between characters and environments, such that when the proportions of one character or object in the scene is changed, then the animation is updated at the joint angle and position level in order to preserve the context of the original interaction automatically bypassing the need for manual fix-up work.

An example of this is two characters engaged in Judo. The original motion data used to capture this interaction may have been set up precisely such that the character animations conform to one another in the interaction given the human sizes and proportions used to capture the data. For instance, in the original motion data the hand of the one character is correctly placed relative to the shoulder of another character in a certain frame of the interaction. But what happens when the sizes and proportions of the characters are changed, such that we enlarge one character while decreasing the size of the other due to the needs of the game or system. Whilst they will both run the same animations at their different sizes correctly at the joint angle level, on the other hand they will no longer conform to one another because the interaction context will be violated. For instance the one character's hand will no longer be placed upon the other character's shoulder in the aforementioned frame in the interaction, and may instead be far above or below when it should be in contact, and there will be penetrations etc. Another example is a character entering a car. If the character is enlarged, then again the context of the interaction will be violated, such that the character's head may penetrate the car ceiling as he enters, the contacts are lost, etc. To avoid manual fix up work, such problems motivate automatic schemes to adapt the motion to the deformed scenes.

This problem is confounded further in cases such as video game applications. Modern computer games may include a wide range of environments and character designs in the package. Often they even allow players to design their own virtual characters with unique sizes and proportions. Those same characters would then need to navigate through restricted environments and handle tools or interact with one another. Preparing the interaction motion data manually for all combinations of character morphologies and surface geometries is costly and laborious and furthermore can significantly increase the amount of data in the game package adding overhead in memory. Thus, adapting recycled motion data of close interactions for different characters and objects is convenient for game developers.

To solve such problems, the first thing one would need to do is to encode the context of the interaction. It is therefore proposed that interactions are encoded based on spatial relations of interest. Some examples of spatial relationships are for example the location of a character's head relative to a ceiling in the case of the car interaction, or the position of the arm of one character in relation to the body of the other in the judo example. In order to preserve the context of an interaction, we would then need to adapt the animations, or the postures of the characters over a window of time, in such a way that we preserve the spatial relationships of the original interaction.

There have been previous techniques that attempt to adapt and retarget motions by attempting to preserve spatial relationships which are encoded in different ways. But 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 animations by imposing a sparse set of positional [Gleicher 1998; Lee and Shin 1999; Choi and Ko 2000] and distance constraints [Liu et al. 2006] which represent the spatial relations. As we present later in this thesis such techniques can suffer from collisions and discontinuities when applied to scenes of close interactions. On the other hand, 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.

[Ho et al 2010] propose a structure known as the interaction mesh (see Fig. 1) for instance, where a Delaunay triangulation is applied to all the joints/limbs in question for the two interacting characters to generate a volumetric mesh known as the Interaction Mesh which encodes the spatial relations. Then when a positional constraint is enforced upon one of the limbs or joints or the size and proportions of one character's limbs is changed introducing new bone length constraints, the rest of the joints are adapted by attempting to preserve the volume of the interaction mesh using Laplacian deformation techniques to preserve the spatial relations subject to the new constraints.

The problem is that methods such as [Zhou et al. 2010] and [Ho et al. 2010] require costly iteration processes. Also due to their discrete representations, they require the use of spacetime optimizations over windows of time when applied to scenes where the spatial relationships dynamically change over time. This means that they can't realistically be applied for real time applications because they can't be solved continuously per frame and must be solved across the entire timeline of the interaction.

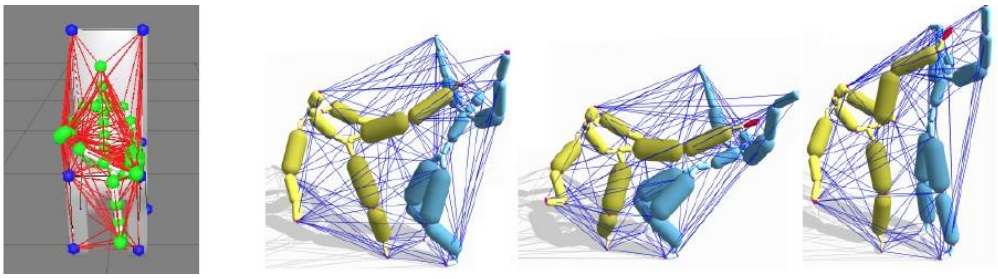


Figure 1: Interaction mesh [Ho et al, 2010] used to preserve interaction context when character morphology is deformed. Expensive as it requires space-time optimization, unsuitable for real-time applications.

Often applications would need motion adaptation to take place in real-time, particularly where you have dynamically changing scenes, or where a player can

select characters of different sizes and proportions in a real-time video game environment where we wish to use the same source animations to save data and memory hence we need to retarget on the fly. This means that it's important to introduce a motion adaptation scheme which is real-time and can be applied continuously per frame.

1.3 Goal & Contributions

In this thesis, we aim to overcome the problems described above as well as the problems of existing techniques by proposing a new descriptor that can be used to represent spatial relationships between character limbs and objects or environments in order to encode interactions. With this representation, the pose of each body part is described by the weighted sum of relative vectors with respect to other limbs or points selectively sampled upon triangulated meshes in the scene based on proximity (see Fig. 2). In the case of a character-object interaction, when the mesh surface is deformed we can retrieve the target locations of the joints of the character by re-projecting their positions based on the relative vectors but this time originating from the newly deformed descriptors which have updated positions and local 3-spaces at that point. Those target locations are where we want the joints of the character to be relative to the object surfaces in order to preserve the spatial relations.

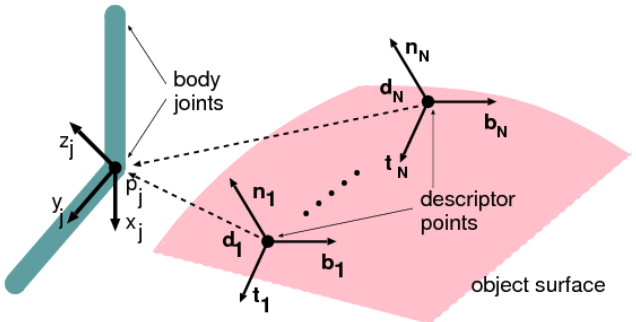


Figure 2: New efficient spatial-relation descriptor proposed based on weighted sum of relative vectors between a character joint and sampled points upon a surface involved in an interaction, suitable for real-time applications.

We further propose a motion warping framework that ensures that the postures of the characters are adapted continuously between frames even though we solve it per frame and not over a temporal window and even when there are significant deformations in the environment or large updates in the character morphologies. With our warping scheme we apply virtual forces to the joints of the character to

continuously drive them towards their target locations computed by the spatial relationships projected from the deformed descriptors whilst applying constraints in parallel to preserve the rigidity of the character.

Our proposed representation can be used for several purposes and combined with other procedural animation techniques to achieve tasks such as contact repositioning for extreme scene deformations. Mappings can be performed in this space to also adapt motions entirely rather than only single postures to novel geometry with completely different topologies and no known correspondence.

As such we introduce a framework in this thesis for encoding spatial relations and adapting motion to deformed objects, environments and character morphologies which has these properties:

1. Simple
2. Fast & Dynamic
3. Solved Per-frame
4. Smooth & Continuous

1.4 Thesis Outline

This thesis follows the 'Collection of Papers' structure. In this case, it shall consist of four papers, 3 published and one not published yet. All attached publications follow the same theme and involve the use of the proposed 'relationship descriptors' representation as outlined in the contribution of this thesis to solve various problems in a novel way.

Each publication has its own self-contained structure including intro, related work, overview of method, implementation, results, discussion & limitations and conclusion sections plus bibliography of references.

Each publication will serve as its own chapter, preceded by a separate introduction where I shall emphasize the extent of my contribution particularly in cases where the publication involves collaboration with others. All publications use the representation I propose however in a novel way to solve problems in the realm of online motion adaptation.

Chapter 2

Publication 1: [SCA 2013]

Relationship Descriptors for Interactive Motion Adaptation

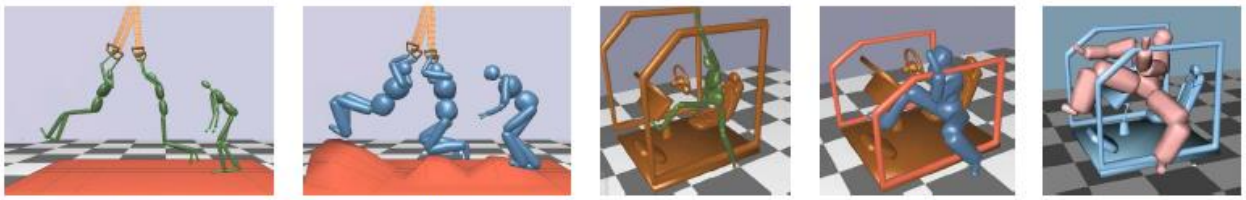


Figure 1: *(a) An original swinging motion (b) adapted to a deformed terrain and enlarged body. (c) A motion to ride on a car (d) adapted to a larger character and a smaller entrance. (e) Failure case when using a sparse set of positional constraints and repulsion in the direction of normal vectors.*

The next attached paper introduces the core contributions of the thesis. It introduces the spatial-relation based descriptors based on relative vectors which are used to encode interactions as well as the fast motion warping scheme used to achieve continuous interactive adaptation. I am the sole first author of this paper, responsible for the entire work from ideation, to the method, implementation, and all the results. Special thanks goes to my supervisor Taku Komura for his guidance as well as Myung Geol Choi for assisting me in acquiring motion capture data used in the results in this paper as well as running previous techniques with the motion data to prove our results are better than previous less suitable techniques for the problem domain.

Relationship Descriptors for Interactive Motion Adaptation

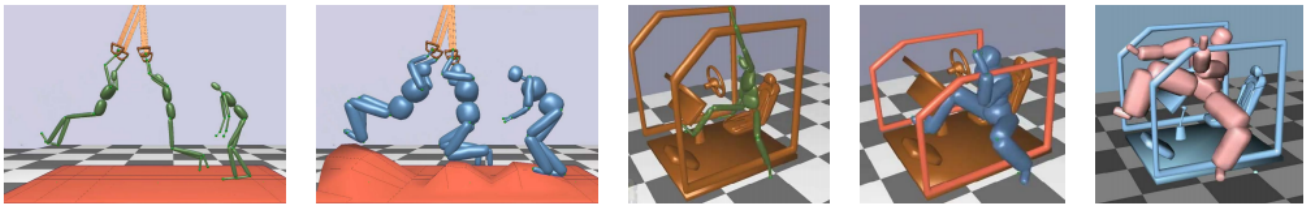


Figure 1: *(a) An original swinging motion (b) adapted to a deformed terrain and enlarged body. (c) A car entering motion (d) adapted to a larger character and a smaller entrance. (e) Failure case when using a sparse set of positional constraints and repulsion in the direction of normal vectors*

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 and orientation 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:

1 Introduction

Synthesizing scenes where a character closely interacts with mesh structures, such as moving in a restricted environment and manipulating objects, from scratch, is a very painstaking process. Capturing such movements may require tracking the objects, scanning or modeling their geometry, and coordinating the object movements to the human body or vice versa.

These processes are very labor intensive and require a long phase of refinement. Recycling existing motion data to mesh structures of different scales and geometries, and characters of different morphology can greatly shorten the process of creating such an animation.

There is also a strong demand in the gaming industry for an online motion adaptation scheme that can handle close interactions between characters and mesh structures. Modern computer games allow players to design their own characters and environments. The player-designed characters may need to move through environments, manipulate objects, and fight with characters designed by the other players. Preparing the motion data for all combinations of character morphologies and object geometries can significantly increase the amount of data in the game package. Thus, recycling a single motion for different characters and objects is essential for the game publishers.

Existing techniques of motion retargeting and adaptation either require high computational cost or can only handle a limited variation of movements. Some methods handle such problems by imposing positional constraints [Gleicher 1998; Lee and Shin 1999; Choi and Ko 2000]. However, positional constraints are not appropriate especially when the motion involves close interactions but no contacts. In order to cope with this problem, methods to represent the open space between character by minimal spanning trees [Zhou et al. 2010] or volumetric meshes [Ho et al. 2010] are proposed. Minimal spanning trees and volumetric meshes cannot be applied for online motion adaptation when the spatial relationship between the objects are dynamically changing. This is because they are discrete descriptors whose topology may change between frames, which causes the adapted motion to become discontinuous when the geometry of the objects involved in the scene changes. As a result, they require costly spacetime optimization processes using a temporal window, which limit them to offline animation synthesis.

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 and orientation vectors with respect to points sampled on the surface of the object in close proximity. The descriptors are smartly 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-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 kinematic constraints such as positional 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 positional 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], motion editing [Gleicher 1997; Callennec and Boulic 2004; Shi et al. 2007] and motion retargeting [Gleicher 1998; Lee and Shin 1999; Choi and Ko 2000]. However, positional constraints and contacts are not necessarily good descriptors for many close interactions. In

many types of movements that entails close contacts, such as rolling, crawling and lying on chairs, the way contacts happen can be irregular and instantaneous [Liu et al. 2010]. In addition, they do not explain about the spatial relationships between the body and the object that do not make contacts. In fact, 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.

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.

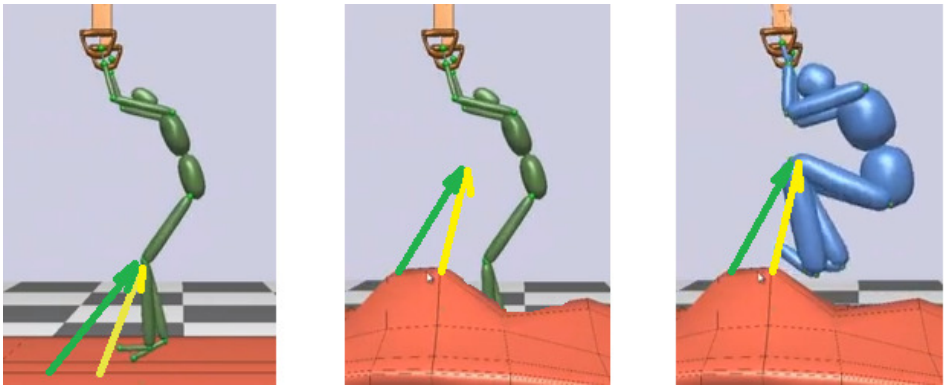


Figure 2: *The overview of the method: The position of the joints are represented by the weighted sum of relative vectors from descriptors computed in the original motion (left). The target location of the joints are computed from the updated geometry of the environment (middle). The motion adapts to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

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 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 and orientations from points called descriptor points sampled on the mesh structures that compose the environment or the objects that the character interacts with (see 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 Fig. 2(b), Section 5), but are only adopted according to our novel scheme specialized for close interactions (see Fig. 2(c), Section 6).

During run-time, the geometries of the objects or environment and the morphology of the environment are updated, 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(d), Section 7).

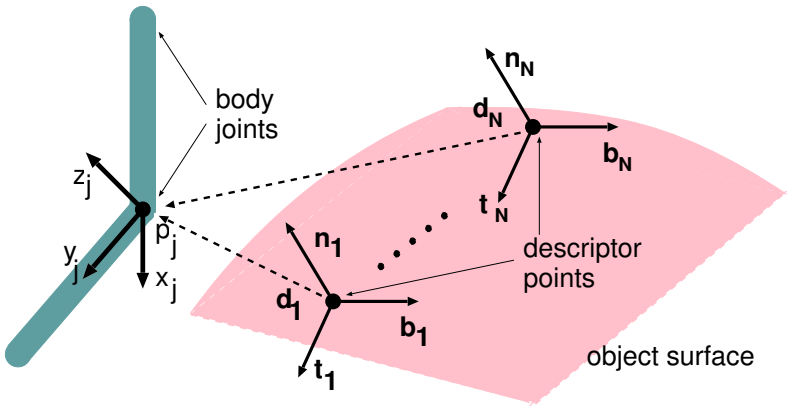


Figure 3: *The local coordinates defined at the body joints and at the sample points on the object.*

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

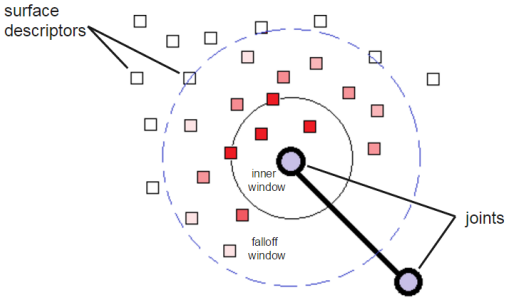


Figure 4: *Spatial window of a joint as it establishes weighted sum of relative vectors with descriptors. Descriptors smoothly slide into the falloff, where weight multiplier begins at 0 and ramps up to 1 as they enter inner window such that we smoothly fade in and out spatial relation weights for continuity.*

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 $\mathbf{d_j}$ using these three vectors:

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

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:

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

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:

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

The weight fades out as the distance between $\mathbf{p_j}$ and $\mathbf{d_i}$ increases (see Fig. 4):

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

where

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

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:

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

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.

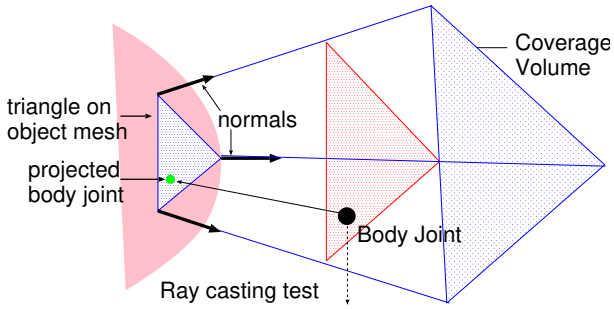


Figure 5: *The coverage of a triangle is defined by its normal vectors at the vertices. Coverage of the body joints are examined by a ray-casting test. Those inside are projected on the triangle and their barycentric coordinates are computed.*

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. 5). We test if the joint lies within the coverage of a triangle by a ray-casting test, 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 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. 5). 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 a convergence algorithm 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

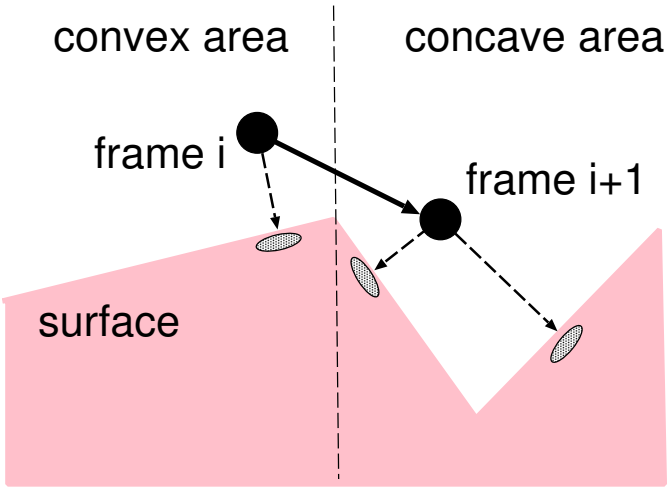


Figure 6: *The disadvantage of using a different set of descriptor points per frame: the number of descriptor points changes when the joint moves into a concave area.*

the body joint (the red triangle in Fig. 5). 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. 6: 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 evalutes 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. Every joint in the character is used when sampling descriptor points upon the scene, and all points sampled by the joints are accumulated into a single final set. We apply an incremental sampling scheme, passing as an argument a geodesic distance radius parameter, whose value dynamically changes according to the distance between a 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 0.03. When a new proximity point is sampled on the surface of the mesh, the previous samples in the geodesic radius are examined (see Fig. 7). 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,

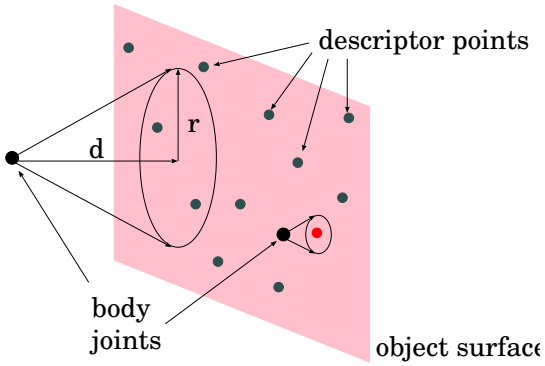


Figure 7: *The adaptive radius scheme to sample descriptor points: The radius scales with the distance. When there is no previous descriptor points sampled in the radius, a new descriptor point is placed. Only a single descriptor point will be sampled in the right case.*

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 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 re-computes 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 different from the affinity of joints in robotics sense, but 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 non-normalized 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 for each timestep, which means force accumulation and integration steps are run N_s times (set to 3 in our experiments), 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 relatioship 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:

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

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 rescaling 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 what data is 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 fi- nally 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

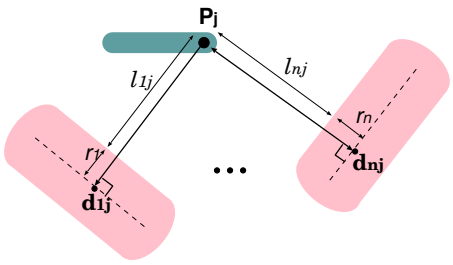


Figure 8: *The bone configuration represented by the weighted sum of relative vectors defined at other bones.*

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)).

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 This can be done in the same way as interactions between characters and mesh structures with slight modification. Same as Eq. (2), the configuration of 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. 8). The descriptor points $\mathbf{d_i}$ are updated per frame by projecting each joint position to the closest point on all the other bones. We do not suffer from the discontinuity problem explained in Section 6 because there is no concavity in bones. 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 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 its position and the capsules of the other bones. They will avoid penetrating each other without much extra computation in this way.

During the motion warping phase, the postures of both characters are simultaneously updated such that both characters get 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 half the radius in the second iteration, and finally use only the fade-off window in Eq. (5) in the last 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. 9(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.

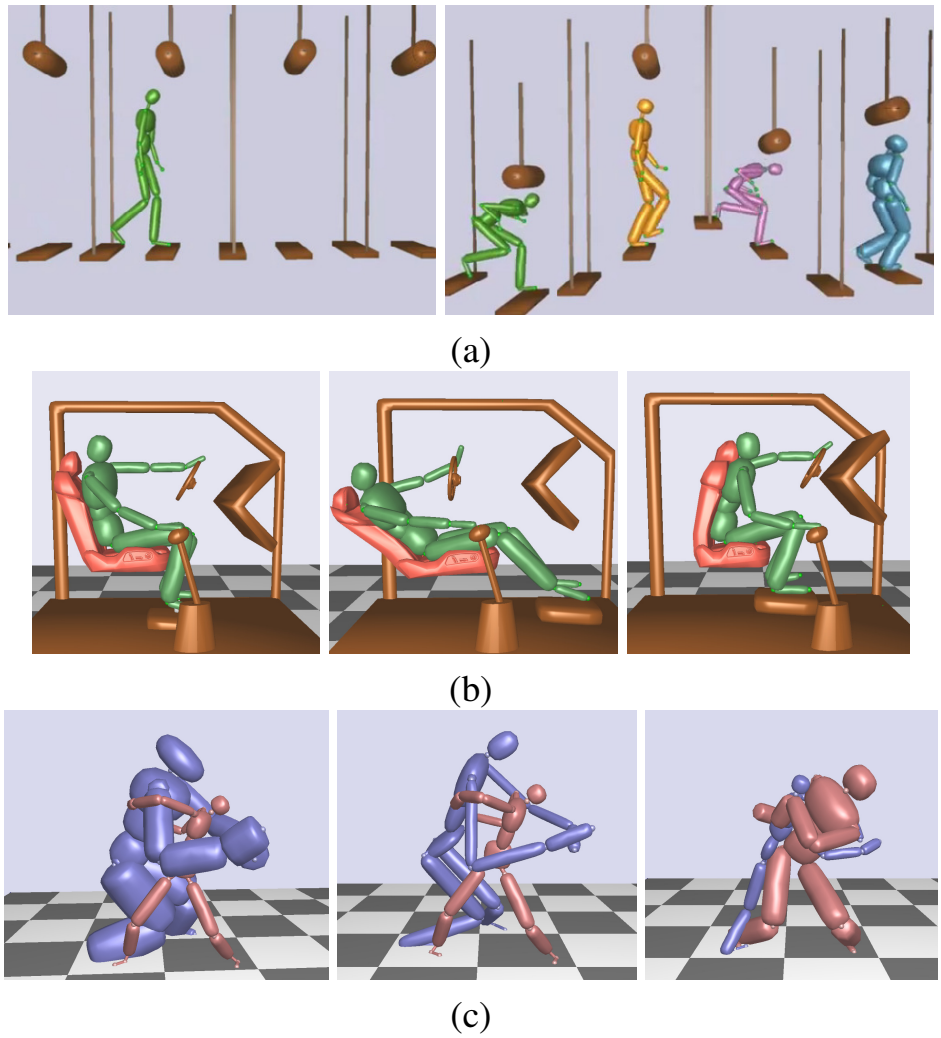


Figure 9: *(a) Adapting a motion to cross a bridge to ones crossing a dynamically deforming bridge. (b) Adapting movements inside a car to different designs. (c) Retargeting a judo motion to characters of different sizes.*

The motion of a glove to be put on a human hand is converted to be put on another hand with completely different geometry and size (see Fig. 10). 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 1GB of memory.

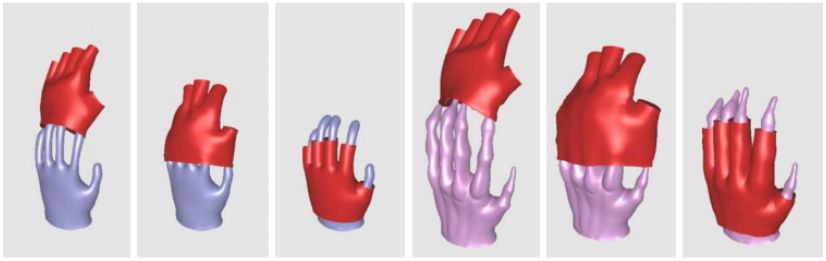


Figure 10: *An original animation of putting on a glove (left) adapted to a hand with a different geometry (right).*

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 angu-

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*

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	46	n/a	29	0.8 %
cloth	11241	1300	3	4	n/a

lar 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 represented by the relation descriptors. This is the same for methods based on space-time constraints, such as [Liu and Popovic 2002]; the postures represented by the relation descriptors can be used as constraints and a motion that spatiotemporalily 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 characters 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).

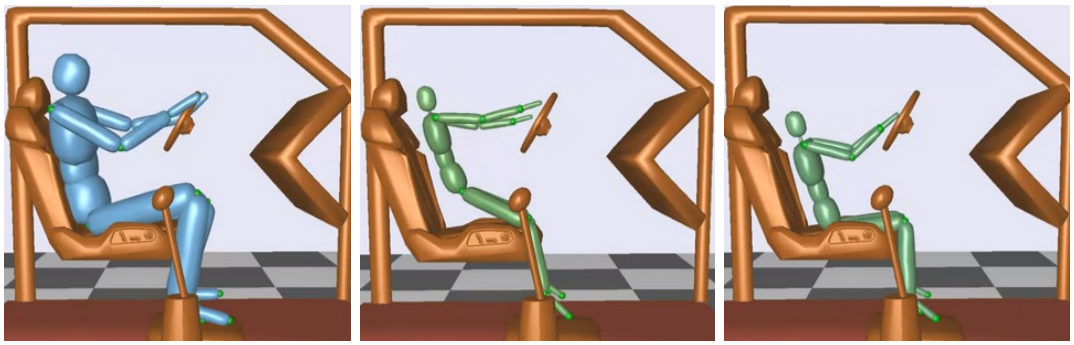


Figure 11: *(left) The original driving, (middle) retargeted to a smaller character, and (right) fixed motion by reducing the affinity values due to the seat-back.*

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.

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.

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. 1997. Motion editing with spacetime constraints. *Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics*.

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

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, 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.

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.

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 Transactions on Graphics 26*, 3, 81:1–81:10.

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*.

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

Chapter 3

Publication 2: [Motion In Games 2015]

3.1 Carpet Unrolling for Character Control on Uneven Terrain

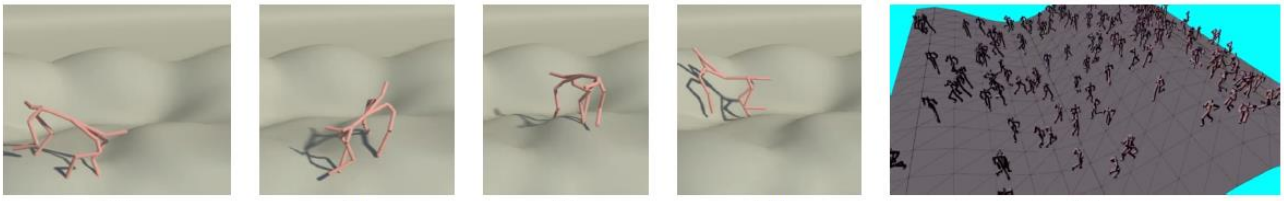


Figure 1: *(a)-(d) Snapshots of a quadruped character walking over an uneven terrain and (e) a snapshot of a large number of biped characters walking in an environment.*

So far we have demonstrated an effective scheme which can be used to adapt animations to deformed scene objects or character morphologies in such a way that we preserve the original context of the interaction in a real-time fashion. The scenarios demonstrated so far include examples such as multi-character interactions i.e. two characters engaged in judo, or character-object interactions such as a character entering a car and assuming a driving position etc. These motions are adapted in a highly generic way, which proves the strength of the system. However some scenarios could be improved with additional tailoring of the method.

An example of a particular use case is the adaptation of locomotion to uneven terrain, a very common problem which deserves its own attention. Previous methods attempt to achieve adapting locomotion in non-generalized ways. Assuming the original motion is a walking forward one, previous methods have to first determine the vertical distance of the feet from the ground at any one frame. Ray-casting must then be used to project the feet to the correct point relative to the height map of the modified ground terrain and then inverse kinematics must be used to constrain the feet of the character to the correct distance above the terrain based on the height map. Furthermore additional fix-up to the hips must be applied to ensure that the feet targets are reachable to begin

with. Due to the non-generalized nature of such methods the feet and hips have to be marked up accordingly to begin with, and IK must be set-up, so there's more preparation work that must be done before adapting, and one can't simply plug and play. Furthermore only the selected joints will be manipulated by IK such as the marked feet and hips, while the rest of the body and spatial relations are ignored which is a further limitation. Depending on the height map, the adapted walking animation may then result in wrong spacing between the footsteps and may speed up and slow down irregularly. It's also difficult to apply additional manipulations such as introducing curving and banking to the locomotion path.

Using our proposed relationship descriptors on the other hand, we can solve the problem of adapting locomotion to uneven terrain in a highly generalized way without much parameter tuning and in such a way that all joints are implicitly considered already, including the hips for reachability. Furthermore it's easy to manipulate the path to add effects such as curving and banking since that can be achieved by manipulating the descriptors directly in a spatial sense bypassing the need to solve over a temporal window, since we would have a collection of descriptors sampled over the entire course of the source animation. Finally it's easy to solve the problem of preserving foot-spacing using descriptors with the same rationale.

If we assume we're adapting a character walking in a straight line upon a flat plane geometry, then the descriptors will all be sampled upon the lane of the character's path upon the ground plane. The placement of the descriptors upon the ground plane will roughly represent the shape of a carpet. When the ground is deformed with a height map the 'carpet' of descriptors will be projected vertically and we can adapt the character to the uneven terrain. However in the case of locomotion there are additional problems to consider. For one, locomotion paths are highly generic and we may wish to dissociate the descriptors in this case from the triangulated mesh of the ground plane such that we can deform the descriptors and create new locomotion paths independently from the mesh. For example we may wish to deform the character's path by attaching the carpet of descriptors to a spline representing the path and curving that spline to deform the descriptors procedurally to achieve a banking like effect to the adapted character locomotion.

Furthermore, if we simply curve the descriptors along a deformed spline or project them vertically based on a height map, then we may introduce unwanted stretching or compressing between the descriptors. Unlike previous examples, not only do the spatial relations between the character limbs and the descriptors plotted upon the ground plane encode meaning, but the self-spatial-relationships between the descriptors themselves also do. In this case the spacing between the descriptors represents the spacing of the foot plants of the character and stretching or compressing the descriptors in this case may affect the walk gait of the character and cause for irregular speed up and slow down of the adapted character. Therefore additional attention is required to solve this issue such that we not only adapt the character to the deformed relation descriptors, but we also attempt to preserve the spacing between the descriptors themselves to maintain footstep spacing.

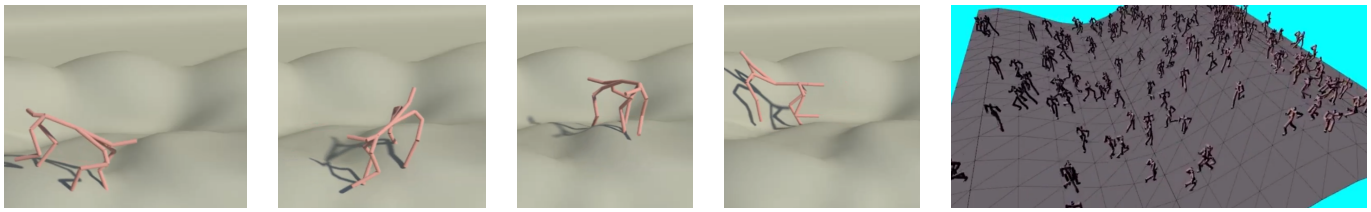
Finally, it's worth noting that the way in which we employ our motion adaptation scheme is designed to be efficient and parallelizable. For one, every joint position can be adapted independently from every other joint before applying the posture warping scheme. And furthermore, every frame of the animation can be independently adapted from every other frame and the results can still be assembled together continuously. This means that our system can benefit from large performance gains if we apply GPU parallelization. One of the most common and practical applications of this is the simulation of crowds of characters all adapting to uneven terrain.

We address all the above concepts and problems in the following chapter. The next attached paper contains the natural next step for the research which is the use of our proposed relationship descriptors for the major use-case of adapting locomotion to uneven terrain as well as GPU parallelization of the work across multiple characters and at the individual joint level for large performance gains. Mark Miller is the first author of this collaborative work though since the parallelization was the main focus. However I am responsible for the core work which this is based on as well as ensuring the concept that my system/method is parallelizable to begin with such that it can enjoy such performance gains. Furthermore I am responsible for the ideation and prototype for the adaptive locomotion and I achieved the first preliminary results for adapting a quadraped to uneven terrain. This involves the use of descriptors and editable curves to which the descriptors are mapped for smooth curving on the fly. I hence came up with the first idea and results for the concept of the 'carpet unrolling' itself such that the locomotion is adapted regardless where the character is initially placed even though the source animation is only a single walking forward one. It's

important to note that all the GPU implementation work though is purely the work of Mark Miller for which this paper is the focus of otherwise.

I then expand upon the work in a follow-up section that tackles the problem of the stretching and compressing of descriptors which can affect the footstep spacing and hence the walk gait. To this effect we also triangulate the descriptors to introduce distance constraints between adjacent descriptors and run particle simulation steps to relax them and preserve the distances between them after deforming them to a height map or spline.

Carpet Unrolling for Character Control on an Uneven Terrain



(a) (b) (c) (d) (e)

Figure 1: *(a)-(d) Snapshots of a quadruped character walking over an uneven terrain and (e) a snapshot of a large number of biped characters walking in an environment.*

Abstract

We propose a carpet unrolling type of relationship descriptor that computes the joint position based on the sum of relative vectors originating from local coordinate systems embedded on the surface of the carpet. Given a terrain that a character is to walk over, the carpet is unrolled over the surface of the terrain. The carpet adapts its surface to the geometry of the terrain and also curves according to the trajectory of the character. As trajectories of the body parts are computed as a weighted sum of the relative vectors, the character can smoothly adapt to the elevation of the terrain and the horizontal curves of the carpet. The carpet relationship descriptors are easy to parallelize and hundreds of characters can be animated in real-time by making use of the GPUs, which makes it applicable to real-time applications such as computer games.

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

Keywords: rig, animation, machine learning, approximation

1 Introduction

In computer animation and computer games, scenes where characters walk over terrains with arbitrary geometry are very common. Characters need to walk over tilted terrains, stairs, and uneven terrains at mountains, while avoiding obstacles, self-collisions and collisions with other characters. Although humans can easily adapt to terrain with different geometry on-the-fly, synthesizing such movements for virtual characters is not straight-forward.

One classical approach to synthesize such movements is to use inverse kinematics. Given a canonical walking cycle over a terrain, the feet trajectories can be adjusted such that the character walks over it. This requires specifying the moments that the feet are in contact with the ground, and producing a trajectory that does not collide with the ground. Also, the the movements of the other parts of the body, including the pelvis, torso and the arms must be adjusted to make the motion appear natural.

Another approach to produce such movements is to apply data driven methods. Movements to walk over various types of terrain can be precaptured and blended to produce movements that satisfies the constraints due to the terrain. This requires capturing various movements and blending them together. Various approaches based on radial basis functions and Gaussian processes have been proposed for such applications. Capturing and managing many types of movements is cumbersome and memory intensive. It will be easier if a canonical motion can be adapted to terrains with different shapes.

In this paper, we solve the problem of terrain locomotion by making use of the relationship descriptors [Al-Asqhar et al. 2013] that is known to be useful for retargeting motions to characters of different sizes or to interact with objects with different geometry. We propose a simple and effective approach that we call "carpet unrolling" to adapt the locomotion to terrains with arbitrary shapes. We present that the method is applicable to characters with different topology, including bipedal and quadruped characters.

The approach is highly parallelizable, and easily runs on the GPU. As a result, the trajectories and movements of hundreds of characters can be adapted during runtime, which makes it applicable to real-time applications such as computer games and virtual reality applications.

2 Related Work

In this section, we discuss about the work related to character control in crowds and motion adaptation methods.

Character Control in Crowd Animation As crowd animation is a very large area, we limit ourselves to methods that involve close character interactions in crowd animation. Very roughly, crowd animation can be divided into global approaches and local agent-based approaches. Global approaches control agents by producing global maps or potential functions [Treuille et al. 2006] and guiding the characters based on such global structures. Despite the fact they produce optimal trajectories to avoid congestion, recently, there is higher interests in local, agent-based approaches, where the control is decentralized to each individual characters.

Various agent-based controllers have been proposed in the area of computer animation and crowd modeling. Based on Renold's flocking model [Reynolds 1987], Helbing et al. [2000] propose an energy-based method that moves the characters towards exists while avoiding each other. Simulations of panicking crowd are produced, and such an approach is useful for evaluating the safety of buildings. Reciprical velocity obstacle [van den Berg et al. 2008] is an approach that is widely adopted in robotics and character animation to control the characters to avoid other characters in the velocity space. Methods based on synthetic vision [Ondrej et al. 2010] can reproduce phenomena such as crowds producing vortices at the intersection and lane-forming when a crowd of people pass through each other.

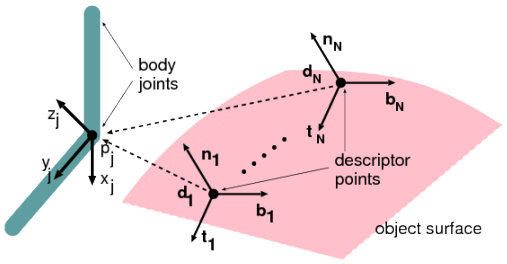


Figure 2: *The local coordinates defined at the body joints and at the sample points on the object.*

In previous crowd animation, each character is mostly treated as a particle or a rigid body and the body movements are simply replayed such that the root of the body follows the trajectory computed from the crowd simulation engine. Our proposed method can be useful for producing an animation of crowds moving over a terrain of arbitrary geometry.

Real-time Motion Adaptation to Different Geometry Recently, there is an increasing interest in adapting character movements and postures to interact with other characters, objects and the environment. A clasical approach to edit the motion of the character to adapt to different geometry, such as the terrain, is to apply inverse kinematics [Lee and Shin 1999; Shin et al. 2001], or spacetime optimization [Gleicher 1997]. Simple control of the end effectors based on inverse kinematics or spacetime optimization do not work well, especially when the motion involves close interactions with the geometry, as unexpected collsions may occur between the body parts.

In order to reproduce more natural movements, data driven approaches are also introduced to follow positional constraints provided by the user. Rose et al. [1998] interpolate movements using radial basis functions (RBF). Kovar et al. [2004] enhance this approach; they search movements of the same classes in the database and produce the user desired movements by interpolating the relevant movements by RBF. Mukai and Kuriyama [2005] apply Gaussing processes to interpolate various movements to reach out, hold objects, and step on stairs of different heights. Mukai [2011] further extends the approach for biped locomotion and propose a data structure called Motion Rings to adapt the locomomotion to different geometry. Although data driven approaches produce excellent results, they require many data samples and the constraints are not precisely satisifed, especially when the training samples are sparse.

Ho et al. [2010] propose a method to adapt existing motion data to different body sizes and environments of different geometry. This approach is further enhanced by Al-Ashqar et al. [2013] to achieve real-time performance. Based on this approach based on relationship descriptors, we present a system that the characters interactively adapt to terrains of different geometry. We also present a parallelisation of the approach such that many characters can be animated in real-time.

3 Relationship Descriptors

In this section, we review the relationship descriptor representation proposed in [Al-Asqhar et al. 2013], 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.

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. 2). 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:

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

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:

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

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:

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

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

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

where

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

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:

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

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 5.

4 Computing the Descriptor Points

The descriptor points are computed by playing the locomotion on a flat terrain. Here we assume we have a locomotion where the character is moving straight in the forward direction. We follow the scheme same as [Al-Asqhar et al. 2013] for computing the descriptor points: Starting from the middle of the gait cycle, the joint positions are projected onto the ground. If there are no descriptor points within in a circle whose radius is the same as the joint height, then we keep that point as the descriptor point. The descriptor point remains for the entire cycle of the gait.

An example of a set of descriptor points for a biped walking character is shown in Fig. 3.

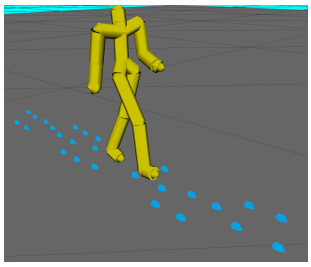


Figure 3: *The descriptor points of a walking biped character embedded inside the carpet.*

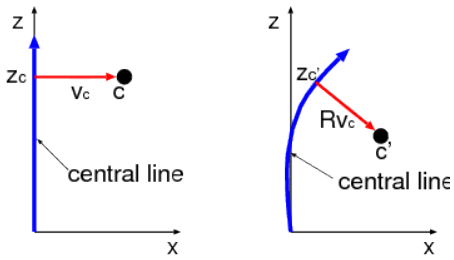


Figure 4: *The descriptor point's offset with respect to the central line is computed in the canonical state (left). When the central line is deformed (right), the descriptor point is computed by adding the rotated offset to the updated projection point on the curve.*

5 Motion Adaptation

Using the positions computed by Eq. (2) from the updated position of the descriptor points as the target, we compute the final posture of the character by an iterative inverse kinematics scheme based on [Jakobsen 2001]. This scheme can be roughly broken down into the following four steps: (1) the computation of the affinity, which determines how strongly the joints must be pulled towards the target positions, (2) the force accumulation and integration step, where the effect of external forces such as pushing, pulling etc. are taken into account, and (3) the bonelength constraint step, where the joint positions are updated such that the distance between the adjacent joints are kept constant.

Affinity Calculation: The affinity values are computed by summing the weights of the associated descriptors on the surface computed in Eq. (6) and normalizing them:

$$s_j = \frac{\sum_i^N w_{i,j}}{\sum_j^{N_j} \sum_i^N w_{i,j}} \tag{7}$$

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

Force Accumulation and Integration: 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:

$$\mathbf{f}_j = k(\mathbf{p}_\mathbf{j}^{\mathbf{tar}} - \mathbf{p}_\mathbf{j}^{\mathbf{cur}}) + \mathbf{f}^{\mathbf{ext}}. \tag{8}$$

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

The target position of the joints are then computed by Verlet integration

$$\mathbf{p}_\mathbf{j}^{\mathbf{new}} = \mathbf{p}_\mathbf{j}^{\mathbf{cur}} + d(\mathbf{p}_\mathbf{j}^{\mathbf{cur}} - \mathbf{p}_\mathbf{j}^{\mathbf{prev}}) + \frac{1}{2}\mathbf{f}_\mathbf{j}\frac{1}{N_s^2} \tag{9}$$

where $\mathbf{p}_\mathbf{j}^{\mathbf{prev}}$ 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 $\mathbf{p}_\mathbf{j}^{\mathbf{new}}$, we compute the final positions of the joints that satisfy the bone-length constraint 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:

$$\Delta\mathbf{p}_\mathbf{j} = \frac{s_k}{s_k + s_j}\frac{\mathbf{p_j} - \mathbf{p_k}}{\|\mathbf{p_j} - \mathbf{p_k}\|}(l^0 - \|\mathbf{p_j} - \mathbf{p_k}\|) \tag{10}$$

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.

6 Carpet Unrolling

We describe about the carpet data structure that we use to adapt the character movements to curves and along the geometry of the terrain. The carpet is a data structure that the relationship descriptors are embedded inside. It is located in front of the character in the direction the which character is heading (see Fig. 3). The carpet can be bent or twisted to make the character turn, or projected vertically onto terrain to allow for the character to walk on some surface.

Given descriptor points $\mathbf{d}$ we find points $\mathbf{c}$, which are local to the character. This is done by multiplying by the character's inverse world matrix $\mathbf{c} = \mathbf{W}^{-1}\mathbf{d}$. The opposite operation is also possible, and the descriptor positions in world space can be given by the forward world matrix $\mathbf{d} = \mathbf{W}\mathbf{c}$.

Turning It is possible to turn the character by bending the carpet along some curve. As the character always follows the location of the carpet the turning motion will adapt naturally.

We use NURBS curves to represent the central line of the gait. Given a straight walking gait, a central line is drawn on the ground that connects the initial and final position of the root. We here assume the central line is along the z axis, where the ground is represented by the xz plane. The projection of the descriptor point $\mathbf{c}$ on the z axis can be obtained by $\mathbf{z_c} = \mathbf{c}(0, 0, 1)^T$, and the offset of the descriptor point from the central line can be obtained by $\mathbf{v_c} = \mathbf{c} - \mathbf{z_c}$ (see Fig. 4, left). When the central line is deformed by moving its control points, the descriptor points are recomputed by adding the rotated offset vector to the corresponding projection point (see Fig. 4, right): $\mathbf{c}' = \mathbf{z}'_\mathbf{c} + \mathbf{R}\mathbf{v_c}$, where $\mathbf{z}'_\mathbf{c}$ is the updated position of the projection point after the curve is deformed, $\mathbf{R}$ is the rotation matrix computed using the tangent direction of the curve at $\mathbf{p}'_\mathbf{c}$.

Terrain Adaption To adapt the animation to some terrain the carpet can be projected onto the surface. To do this the descriptors are converted to world space, projected vertically, and then converted back. $\mathbf{c}' = \mathbf{W}^{-1}\mathbf{\Phi}(\mathbf{W}\mathbf{c})$ Many applications have accelerated data structures for this task such as height-maps, but in the

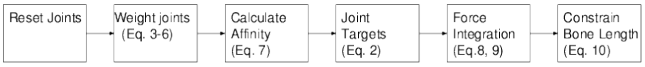


Figure 5: *The flowchart of the motion adaptation scheme.*

general case, where the terrain consists of triangular polygons, the projection operation can use the barycentric coordinates of each triangle t_1, t_2, t_3 in the xz plane α, β, γ. These are first used to test if a descriptor point $\mathbf{d}$ has xz coordinates which lie within the triangle $0 < \alpha < 1, 0 < \beta < 1, 0 < \gamma < 1$. If this is the case the barycentric weights can be used to find the projected position in world space $\mathbf{d}' = \alpha t_1 + \beta t_2 + \gamma t_3$. Many triangles can be discarded from this test by first calculating their axis aligned bounding boxes and not performing any tests if the descriptor point lies outside of it on the xz plane.

7 Parallel Implementation

Our approach is highly parallelizable and can be accelerated using GPU devices where many characters are animated. In this section, we briefly describe how we divide the entire task into subtasks that are executed by individual kernels written in the OpenCL language.

For example, computing the target joint locations by summing the relative vectors (Eq. 2) does not require to know the position of other joints. As such, we can evaluate all joints of all characters in the system simultaneously at many stages. As this current implementation of the system does not concern much interaction between characters, we can evaluate the joints of all characters in the system in parallel. As GPUs can execute hundreds of threads concurrently, this allows us to evaluate hundreds of joints very quickly at the same time. This results in extremely large speed ups compared to the original sequential version.

The breakdown of the motion adaptation task is shown in Figure Fig. 5. In all stages, each character is processed independently from the other and in parallel using an OpenCL workgroup. The description of each stage is as follows:

- **Resetting Joints:** This stage is simply concerned with setting the rest position of all the joints for the current frame. It also resets the state of the system. Each joint is processed independently within the workgroup, using one thread per joint.
- **Weighting Joints:** Calculates the weights of the descriptors and their effects on the joint of the system (Eq. 3-6). As above, one thread per joint is used to performed this stage.
- **Calculate Affinity:** Calculate the affinity value (Eq. 7) that determines how stiff the joint should stay in the space. This stage is performed with one thread per character since the joints cannot be processed independently.
- **Joint targets:** This stage deals with computing the target position of the joints based on the deformed descriptor positions and the previously calculated weight and affinity values (Eq. 2). The joints are processed independently using multiple threads.
- **Force integration:** This stage treats each joint as a particle. Forces are applied to the particles by PD control (Eq. 8) and the positions are integrated by Verlet integration (Eq. 10). Each joint is processed independently by a thread.
- **Constrain Bone Lengths:** We constrain the distance between the joints (Eq. 10). This stage needs to be done with one thread per character as the joints need to be traversed iteratively.

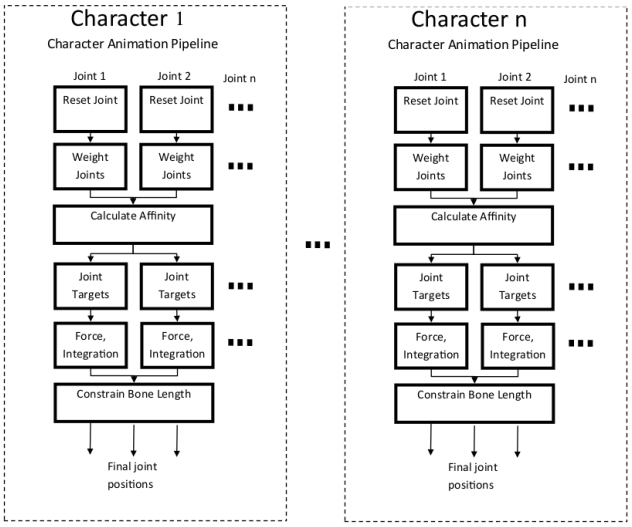


Figure 6: *The parallel pipeline of the system.*

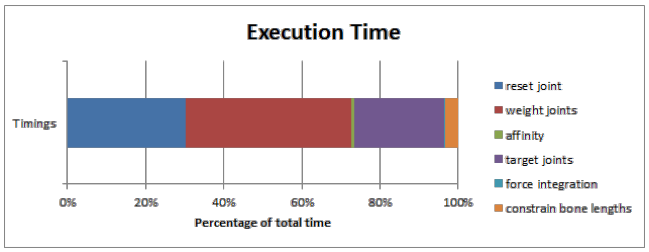


Figure 7: *The breakdown of the execution time.*

The parallel pipeline of the system is shown in Fig. 6, and the breakdown of the execution time is shown in Fig. 7.

8 Results

In this section we present the results of producing the animation of characters walking over uneven terrain using the carpet technique. We also present the performance of the system when running it on different parallel setups. The synthesized animation can be viewed in the supplementary video.

Walking Over an Uneven Terrain We use our approach to generate locomotion for a quadruped character. Given a simple walk loop consisting of just 35 frames of animation we generate a natural looking motion of the character walking over terrain and turning in various directions.

First the looped walking motion is repeated to generate a long motion of the character walking in a straight line. Relationship descriptors are generated under this motion and the weights for the descriptors are calculated. The descriptors are then deformed along a nurbs curve, which represents the trajectory of the character across the terrain. This introduces the turning into the character's motion. Finally, the descriptors are projected vertically onto the terrain and the joint positions integrated using the descriptor weights. This creates the deformed motion of the character walking along the terrain.

Snapshots of the quadruped walking over an uneven terrain are shown in Fig. 1 (a)-(d). Also as can be viewed in the supplementary video, the quaruped character can adapt well to the uneven terrain

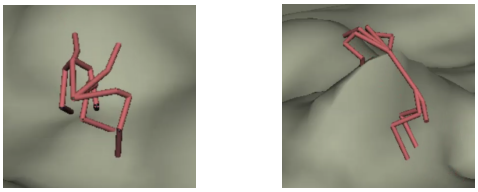


Figure 8: *The method can suffer from very sharp turns (left) and very steep terrains (right).*

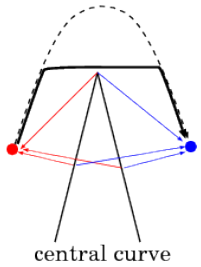


Figure 9: *A sharp turn of the central line will result in a fast translation of the joints when the body passes the acute region of the central line.*

and conduct a natural gait cycle.

Evaluation of Parallelism Four primary setups were tested: CPU single thread, CPU multi-thread, Integrated Intel GPU (20 cores). and dedicated Nvidia GPU (2688 cores). By testing on the different devices available in the system, we gain insight into which may be best for this specific operation. For the evaluation of the performance, 100 frames were measured and an average taken. We tested with a number of characters ranging from 1 to 512. The hardware used is composed of i7-4790k 4.0GHz CPU, 16GB RAM and a Geforce Titan with 6GB GRAM. The system is written in OpenCL 1.2 and compiled in Visual Studio 2013 on Windows 8.1.

The performance of the system in different setups are shown in Table 1; it can be observed that parallelism provides a significant speedup. The integrated GPU system shows better performance up to ten characters, as it shares the host memory with the CPU, while the the dedicated GPU requires transferring the data from the host to the GPU device. However, the dedicated GPU shows better scalability for more characters, thanks to the large number of cores available. Note that the rendering time is not included in these numbers.

In summary, our algorithm is suitable to parallelize and can animate more than 500 characters in real-time on deformable terrains.

9 Discussions and Conclusion

The carpet unrolling approach is highly adaptive and can apply a single type of locomotion to various terrains with different geometry. Compared to classic methods based on inverse kinematics, there is no need to specify when the feet are in contact with the ground. The body movements are automatically computed from the deformed geometry of the terrain, and therefore there is little parameter to tune to produce natural movements. The method is highly parallelizable and can be implemented on the GPU, resulting in hundreds of characters animated in real-time. Currently, we limit the interactions to that between each character and the terrain. In the future, we plan to enhance the method and simulate also the character-character interaction as well as character obstacle interactions.

Limitations Although the characters can adapt well to the terrain, the method may suffer from sharp turns or very steep terrains, as can be observed in Fig. 8. When such extreme cases happen, the linear interpolation of the relative vectors result in bad movements. For example, assume the central curve is turned very sharply as shown in Fig. 9. As the offsets of the descriptor points are made to be perpendicular to the central curve, the offset vector suddenly change its direction when passing the acute region. As a result, the trajectory of the joints farther from the central line will quickly translate when the body passes the acute region, as shown in the bold line of Fig. 9, where the ideal trajectory is a mild curve drawn by the dashed line. A better scheme to locate the descriptor points is needed to handle such a case.

Also, our system does not have any collision avoidance framework between characters. This will involve replanning the carpet trajectory on the fly.

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, 45–53.

GLEICHER, M. 1997. Motion editing with spacetime constraints. *Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics*.

HELBING, D., FARKAS, I., AND VICSEK, T. 2000. Simulating dynamical features of escape panic. *Nature 407*, 487–490.

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.

KOVAR, L., AND GLEICHER, M. 2004. Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics 23*, 3, 559–568.

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

MUKAI, T., AND KURIYAMA, S. 2005. Geostatistical motion interpolation. *ACM Trans. Graph. 24*, 3, 1062–1070.

MUKAI, T. 2011. Motion rings for interactive gait synthesis. In *Symposium on Interactive 3D Graphics and Games*, ACM, 125–132.

ONDREJ, J., PETTR, J., OLIVIER, A.-H., AND DONIKIAN, S. 2010. A synthetic-vision-based steering approach for crowd simulation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010) 29*, 4.

REYNOLDS, C. 1987. Flocks, herds, and schools: A distributed behavioral model. *Proceedings of SIGGRAPH 87 21*, 25–34.

ROSE, C., COHEN, M. F., AND BODENHEIMER, B. 1998. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Comput. Graph. Appl. 18*, 5, 32–40.

Hardware Settings	1	10	20	30	64	128	256	512
CPU single thread	7.84 (0.67)	14.33 (6.57)	18.36 (10.43)	24.04 (15.79)	42.1 (33.27)	83.12 (73.4)	166.95 (155.87)	279.17 (266.3)
CPU multi-thread	9.29 (0.29)	9.71 (0.89)	11 (1.5)	11.8 (1.96)	13.95 (3.99)	18.14 (7.89)	27.35 (15.8)	43.93 (32)
GPU integrated	10.17 (0.54)	10.28 (0.9)	10.24 (1.22)	10.8 (1.83)	13.97 (4.09)	20.51 (8.7)	29.31 (16.48)	42.18 (29.5)
GPU dedicated	10 (1.75)	11.62 (2.33)	12.44 (2.87)	13.2 (3.44)	13.68 (3.57)	15.03 (4.14)	19.17 (6.68)	23.64 (12.26)

Table 1: *The computation time (ms) for one frame in different hardware settings. The numbers in the brackets are execution time spent by the kernels, excluding the tranmission time of the data.*

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.

TREUILLE, A., COOPER, S., AND POPOVIĆ, Z. 2006. Continuum crowds. *ACM Transactions on Graphics (TOG) 25*, 3, 1160–1168.

VAN DEN BERG, J., PATIL, S., SEWALL, J., MANOCHA, D., AND LIN, M. 2008. Interactive navigation of multiple agents in crowded environments. In *I3D '08: Proceedings of the 2008 symposium on Interactive 3D graphics and games*, ACM, New York, NY, USA, 139–147.

3.3 Preserving Footstep Spacing for Carpet Unrolling Method

In this section we now expand upon the work of the ‘Carpet Unrolling’ paper to deal with the problem of the ground plane descriptors stretching, shearing and compressing when deformed to a height map or NURBS curve which can affect the foot-spacing of the adapted character. Unlike previous example scenarios where we only consider spatial relations between character limbs and descriptors, in the case of locomotion the spatial-relationships between the descriptors themselves also encode data which we wish to preserve. The relations between the descriptors represent the footstep spacing of the character and stretching or compressing the descriptors in this case may distort the walk gait of the character speeding up or slowing down the character irregularly. In this section we attempt to tackle this problem.

It is a simple problem to deal with at this point however because it is based on the same principles applied in adapting the character posture to the descriptors. First of all we need to encode the spatial relationships between adjacent descriptors. To do so, we can apply a similar technique to that used in order to generate an interaction mesh as in [Ho et al, 2010]. Except this time we compute simply a 2D Delaunay triangulation upon the descriptor points considering they’re all placed upon a flat XZ plane originally (see Fig. 1). After applying this triangulation, we then delete the extra-long edges. In our example cases any edge with a length greater than half the character’s leg length is deleted, although this parameter can be changed for different effect and depends on the density of the descriptors which is affected by the descriptor sampling window size.

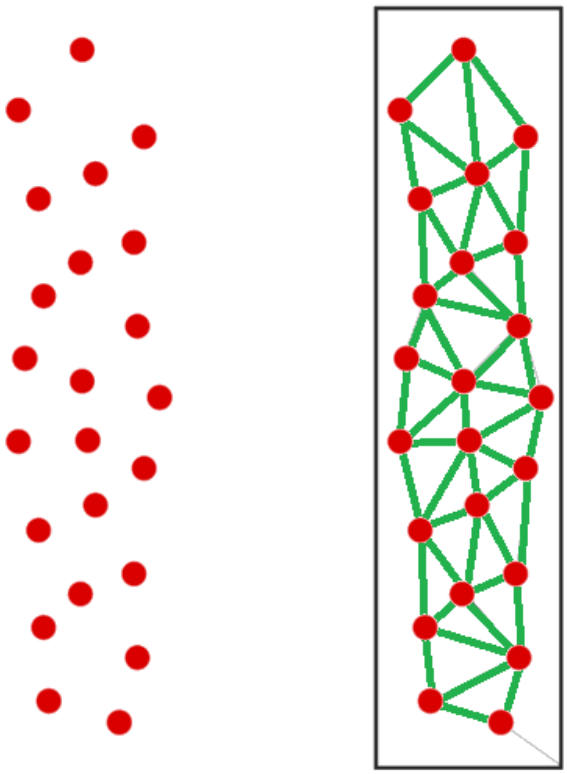


Figure 1: (a) top view of lane of descriptors mapped onto ground plane, sampled from a walking forward animation. (b) 2D Delaunay triangulation with filtering applied to generate a carpet mesh between descriptors where edges represent distance constraints.

When deforming the 'carpet' of descriptors, i.e. when projecting them vertically to the deformed ground height map or curving them along a NURBS curve, we then need to run an iterative fix-up to the descriptors to preserve the spacing between them. We can assume that the carpet of descriptors is now a triangulated mesh essentially which means that we can simulate it as one would simulate cloth. In other words, every descriptor now represents a particle, while every edge between two descriptors on the carpet mesh represents a distance constraint. Whenever the carpet mesh is deformed, we can simply relax it by applying a particle/cloth simulation step to attempt to preserve the spacing between the descriptors. It's unwise to apply the distance constraints as hard constraints because that would make it difficult to deform the carpet in a significant way and deviate from its original shape, but with soft distance constraints however and a limited stretch resistance we can circumvent this problem. Any cloth particle simulation system can be used to achieve this effect. In our case we use simply employ the same particle system which we already outlined in the motion warping phase of our first paper which is used to warp the

joints of the character to the target posture whilst preserving bone length constraints based on the particle system in [JAKOBSEN, 2001]. The joints are simply replaced by the descriptor points whilst the bone length constraints are replaced by the descriptor distance constraints upon the mesh but with added stretch capability. To achieve the stretch capability in the constraints step, we simply project 2 particles related by a distance constraint only when they stretch or compress over or under 15% of the original distance.

To deal with overstretching, we check whether two particles related by a distance constraint stretch to over 15% in distance apart from their original distance, and if that's the case then we project them back to their maximum allowed length using the particle constraints step, where the original distance is enlarged by 15%, and where pj and pk are the two descriptor positions respectively:

$$\Delta \mathrm{p_j} = \frac{\mathrm{p_j} - \mathrm{p_k}}{\|\mathrm{p_j} - \mathrm{p_k}\|}(\|\mathrm{p_j} - \mathrm{p_k}\| - (l^0 * 1.15))$$

On the other hand we also deal with compressions by checking whether two particles are at a distance lower than the 15% allowable range and then apply the constraints step if that is evaluated as the case as such:

$$\Delta \mathrm{p_j} = \frac{\mathrm{p_j} - \mathrm{p_k}}{\|\mathrm{p_j} - \mathrm{p_k}\|}(\|\mathrm{p_j} - \mathrm{p_k}\| - (l^0 * 0.85))$$

If we are using a spline we first deform the descriptors based on the spline once, before deforming to a height map. Then we use a cross-iterative scheme cross-alternating between a descriptor deformation step in the form of deforming the descriptors to the height map, followed by a particle relaxation step, and we repeat this process a number of times till we get a reasonable converged result. We use 3 iterations in our demos for the cross-iterative scheme in our demos.

We run additional demos to demonstrate all the above, first of a human biped adapting a running animation to a terrain height map (see Fig 2), and secondly a quadraped adapting to the carpet deformed with a spline (see Fig. 3).

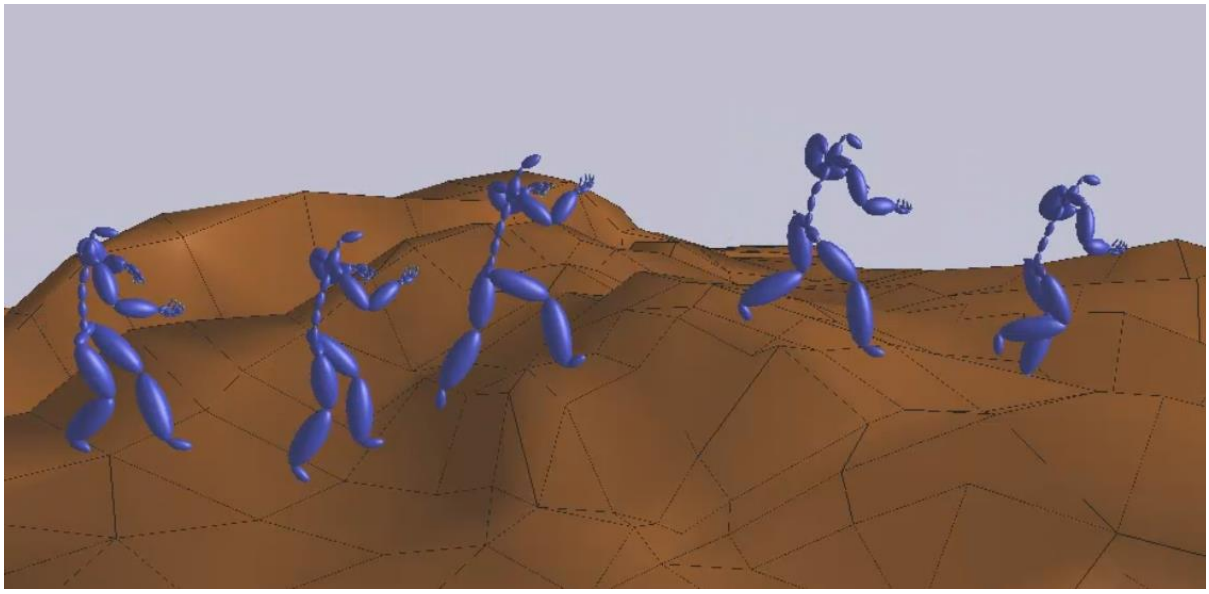


Figure 2: A running animation (biped) adapted to descriptors deformed to ground height map. 5 different frames shown.

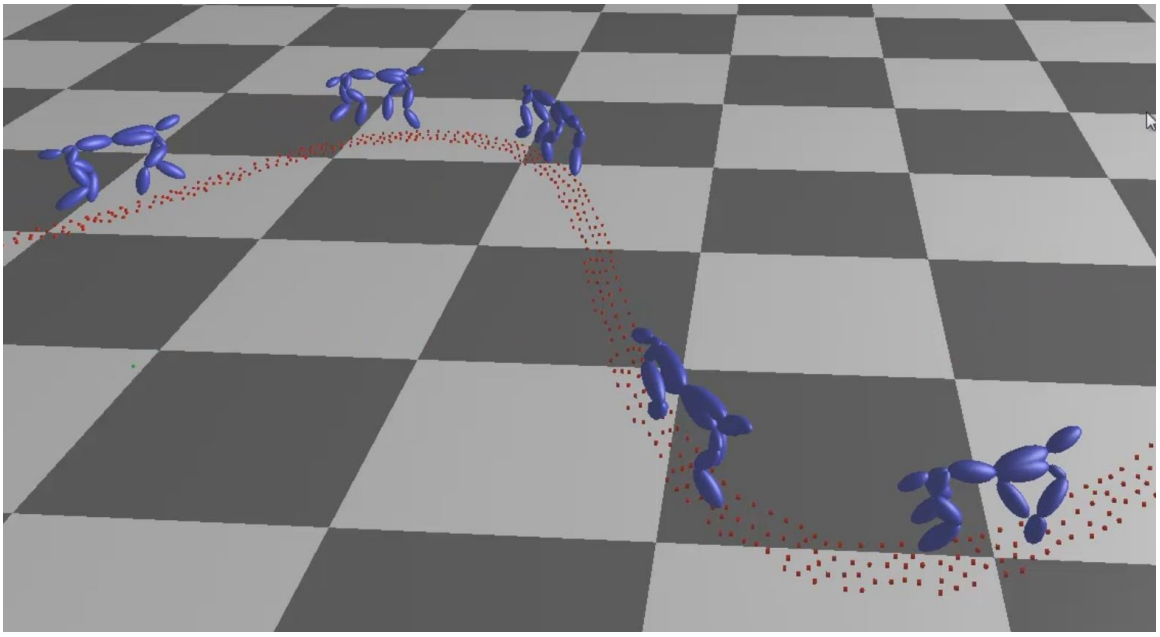
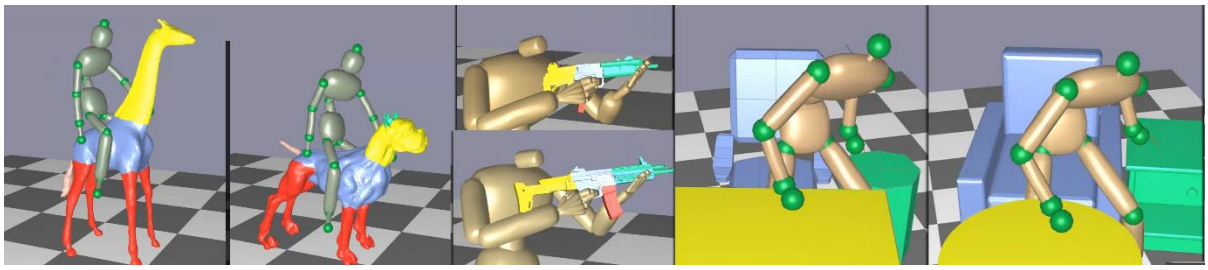


Figure 3: A running animation (quadraped) adapted to descriptors deformed to a curved spline, descriptor adjacency preserved with particle relaxation steps. 5 different frames shown.

Chapter 4

Paper: [ ]

Character Motion Adaptation to Novel Geometry



So far, in all of our demonstrated scenarios which include character-object interactions we have simply used our scheme to adapt animation postures to a single deformed mesh with the same topology. In such cases, we always have a known one to one correspondence between the original and deformed meshes because the mesh topology never changes, but only the vertex positions change, meaning we always know where to place the descriptors after deforming the mesh because the sampled descriptor positions are encoded upon fixed triangle UVs on the object. There is a strong association in this case between the descriptors and the mesh.

What if however we replaced the original 3d object in the interaction entirely with a new object of different topology but of the same class? For example, what if instead of simply deforming the proportions of a car scene or object, we replaced the original car entirely with a novel car of a different mesh topology and structure? In such a case our system would have no idea about how to adapt the motion to the new car, because with the new car topology we would have no association between the descriptors and the novel object and furthermore we would have no one to one correspondence between the original and the novel meshes unless those correspondences were manually authored, meaning we wouldn't know where to plant the descriptors upon the new object such that we can adapt the animation to it.

Despite this fact, being able to adapt animations from a template object of a certain class (i.e. a car) to a novel object of the same class but with different topology automatically would be highly useful and can save lots of time and effort that would otherwise be used manually fixing up motions to conform to the novel objects. In order to solve such a problem we would have to introduce a scheme that can establish some sort of correspondence between the template and novel objects in order to be able to guide the deformation of the descriptors to the new object. This is what we attempt to tackle in this next chapter, except unlike other shape matching techniques that look to explicitly establish a dense one to one correspondence between two objects, we attempt to place the descriptors upon the novel object by fitting a representative posture to the novel object in such a way that the spatial relations between the fitted posture and the novel object are as close as possible to what they were in the original. We then update the descriptor positions to reflect the fitted pose ultimately mapping the entire animation to the novel object over the full timeline of the animation rather than adapting only a single pose.

The next attached paper involves the use of the relationship descriptors space to map entire animations from a template to a novel object/scene based on the co-analysis of the two as well as the analysis of spatial-relation based open-space features surrounding the novel object to determine candidate points for warping a representative posture and hence the descriptors.

I am the sole first author of this paper responsible for ideation, method, implementation, results etc. However I use a feature known as the space coverage feature which is proposed by my colleague Xi Zhao as one of the features used to determine the pose fitting.

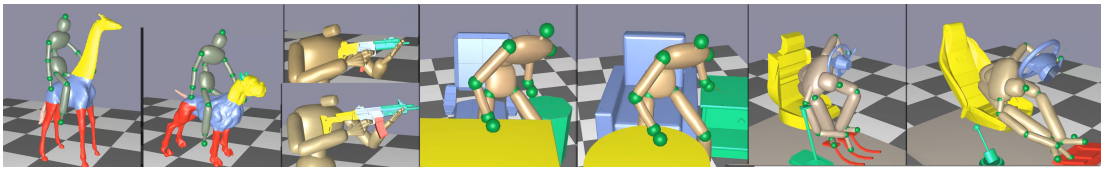
Character Motion Adaptation to Novel Geometry

Rami Al-Ashqar
University of Edinburgh

Xi Zhao
Xian Jiaotong University

Taku Komura
University of Edinburgh

Abstract



We propose a method to adapt motion data of interactions with respect to a template object to novel objects, such as a character manipulating a tool or navigating a restricted environment and avoiding obstacles. Although various methods for hallucinating scenes are proposed, all previous methods only adapt a single pose and not the entire motion. The animation will become discontinuous if such methods are applied to adapt every posture during the motion to novel geometries. In this paper, we cope with this problem by applying the relationship descriptors that describe the motion data by a weighted sum of relative vectors originating from descriptor points embedded in the scene. Our method first assumes a co-analysis between the template and novel objects which are co-segmented such that a rough correspondence between objects with different geometry are established. We then analyse open-space features surrounding the novel object to determine candidate points for warping a representative posture of the character in the animation to the novel geometry and use that to embed the relationship descriptors in the surface of the novel geometry. The relative vectors originating from the new locations of the descriptor points combined with the motion warping scheme will generate a continuous motion of the character interacting with the environment that preserves the original context and style. Our method can be applied for animation production and character control in real-time applications such as computer games given the co-analysis is precomputed.

Index Terms

animation rig, character animation, regression

1 Introduction

HALLUCINATING scenes and virtual environments by virtual characters is a topic that is recently attracting many researchers, thanks to off-the-shelf tools such as Kinect Fusion that can easily scan and digitize indoor environments. Using such methods, virtual characters can be easily added into scenes and motions synthesized such that they appear to interact with their environment. Some methods are probabilistic and can adapt to different sizes and geometries of the objects.

email:s0822954@staffmail.ed.ac.uk
email:xi@xjtu.edu.cn
email:tkomura@ed.ac.uk

Previous methods of character hallucination mainly focus on producing a "snapshot", i.e., a single pose of the character interacting with the objects in the environment, but not a series of movements that include the character approaching and manipulating the objects or avoiding obstacles while interacting with the environment.

Directly applying previous methods for animating the entire motion to interact with objects and environments are difficult due to the discrete representations that are used to describe the interaction between the body and the environment. Most methods specify the contact regions between the body and the environment, and then try to adjust the pose of the character based on such contact information. If such a representation is used, there are going to be discrete switches during the interaction, which can easily result in unsmooth, unnatural movements or collisions with the environment.

In this paper, we tackle this problem of hallucinating the scenes with animated characters, especially where the characters closely interact with the environment. We apply the relationship descriptor representation proposed in [Al-ashqar et al, 2013] for this purpose. In this representation, the motion of the character is described using the weighted sum of relative vectors originating from descriptor points selectively embedded in the environment based on proximity and contacts. The weights are set inverse proportional to the distance between the embedded point and the joint positions, such that they are controlled stronger when the body is closer to the environment. This mechanism guides the character to avoid collisions, and adapt to different geometries while preserving the context of the original scene.

As the motion adaptation method by relationship descriptors [Al-ashqar et al, 2013] assumes the full correspondence between the geometry of the original object and the novel object is known, we propose a novel approach to embed the descriptor points to novel objects. We first apply a co-analysis algorithm to the objects in the same class to compute a rough correspondence between the objects effectively co-segmenting them. We then fit a representative pose of the character during the animation to the novel geometry. Using the fitted pose, we embed the descriptor points into the new geometry such that the fitted pose can be reconstructed using the relative vectors in the original scene. Given the new location of the descriptor points, the entire animation can be reconstructed while preserving smoothness of the original motion and context of the original scene.

We show various examples where the character interacts with the environment, including manipulating and interacting with objects with concavity and

moving through constrained environments such as a car. We compare our method with existing hallucinating methods and show that it outperforms in the sense that it adapts an entire animation over a window of time and not just a single posture. Our method greatly increases the adaptability of the virtual characters to novel geometry. As it runs very fast, it can be applied for interactive applications such as computer games and real-time virtual environments. The online retargetting phase is realt-time and the only real cost lies in the offline pre-computation phase of co-analysis of the template and novel objects. This method can also be applied for applications such as robotics, where the humanoids may need to interact with objects with novel geometries that they may never have faced before.

2 Related Work

Previous work in computer vision involves fitting only individual character postures, or 'snapshots', to novel shapes using the philosophy that analyzing an interaction and its semantics can help with shape recognition and matching as in [Delaitre et al. 2012; Fouhey et al. 2012; Wei et al. 2013].

[Kim et al. 2014] take this further by combining local region classification and global kinematically-constrained search to predict poses for different objects, but again the motivation is shape-analysis related.

In other works this is achieved without even observing the interaction. [Fritz et al. 2006] and [Hermans et al. 2011] predict a list of semantics that represent action types that an object affords. [Sun et al. 2009; Stark et al. 2008] achieve something similar but alongside object classification. [Grabner et al. 2011; Gupta et al. 2011] predict an approximate alignment of a human model to a shape. [Jiang et al. 2013] use a probabilistic method to select a pose from a list of six most common poses and rigidly align it to models in a 3D scene. [Jiang et al. 2012; Jiang and Saxena 2012; Jiang and Saxena 2013] go further by also applying this in object labelling and automatic placement of objects.

[Savva et al, 2014] predict regions in 3D scenes where actions are likely to take place then observe and track people as they interact with the captured environment and train a classifier which allows them to infer the likelihood of actions occurring in regions of new, unobserved scenes.

[Savva et al, 2016] extend this further by using 3D scene and pose data to learn a set of prototypical interaction graphs (PiGraphs) containing priors on the human pose and object geometry to then generate likely poses and arrangements of objects given the action in new regions of a scene.

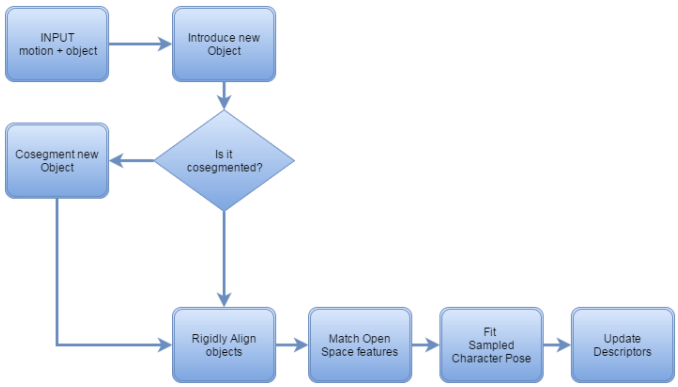


Fig. 1. High level overview

There are a few things that fall in common with most previous techniques involving the analysis of interactions and their semantics as well as the fitting of postures to novel geometry. For one, they are used mostly for the purpose of shape analysis, such as correspondence establishment, detection of salient regions, retrieving functionally similar shapes etc. And for two, only individual postures are ever fit to the novel object rather than entire animations over a window of time.

Our method, while falling in the same realm as the above, has two key distinctions. Most importantly we propose a method that would fit an entire animation across a time-window from a template to a novel object continuously rather than fitting a single posture. The motivations and applications of our method are hence different from previous related work, because our goal is not shape analysis related, but rather it's motion adaptation related as our goal is to adapt entire motions to novel geometry removing the need for animators to recreate or have to manually fix up animations to fit to different scenes. This adds a great degree of flexibility to existing animations and interactions which can be recycled for different scene configurations as a result which is a key motivation. This also means that comparison to previous methods is difficult to evaluate since the goals are different.

3 Overview

Here we discuss the overall methodology of our automated motion adaptation scheme (see Fig. 1).

Our method takes as inputs an interaction sequence between an articulated character and a segmented triangulated object. We assume the triangulated mesh is either a single object which is segmented or a number of labelled

objects comprising a scene. We consider this object in the interaction as the $template$ object.

When we introduce a $novel$ object of a similar class to the template object, the goal is to establish a correspondence between the two objects based on spatial-relation based features. Correspondence is a difficult open problem in research however affected by a lot of varying criteria and semantics and hence to guide the motion transfer, we must first coarsely initialize a correspondence by co-analysis. The main task of co-analysis is co-segmentation, which simultaneously segments all the shapes in the input set in a consistent manner. We use objects of the same class in our demos which are co-segmented by techniques in [Wu et al, 2013], in fact acting upon objects already cosegmented as part of their COSEG database.

We begin by analyzing the interaction by computing relationship descriptors upon the template object and encoding the animation in the space of these descriptors based on spatial relations. To transfer the motion to a novel object the goal then becomes to find a transformation of the relationship descriptor points that maps them well onto the novel object such that the animation will play in the spatial relation space relative to the new object. To find this mapping we must first fit individual postures to the new object, and based on the fitted posture, find a mapping of the descriptor points that reflects the change.

To fit a posture to the novel object, we must first compute the spatial relations of the joints of the character relative to the template object to capture the interaction for that pose. To do so we make use of two features, the distance to object segments and the space coverage feature. We compute these same features upon open space points densely sampled around the novel object and then find strong candidate points towards which each joint can warp towards. After finding ideal candidates for the joints to map to we warp the posture using virtual forces whilst preserving character rigidity. With the posture fitted we run an optimization scheme that finds how the descriptor points on the template should transform to map to the novel object in order for the animation to pass through the new fitted posture.

4 Fitting Characters to Novel Geometry

4.1 Relationship Descriptors for Template Interaction

An interaction representation based on relationship descriptors has been proposed in [Al-ashqar et al, 2013], and is especially useful for smoothly adapting animation in a given context even when the geometry (of the environment or

of the character) is changed. We quickly recall it here for completeness. Rather than representing the configuration of the avatar by a reduced configuration vector $\mathbf{q}$, or by the position of each of its bodies with respect to an arbitrary world frame, the body positions are described by their relative translations from a static set of points called descriptor points (see Fig. 2). The descriptor points are sampled on the surface of the environment (typically by analyzing how closely the body interacts with the surface, i.e., the closer the body interacts with the part, the denser the sampling is going to be).

In this representation, the joint position pj of joint j is represented by the following equation:

$$p_j = \sum_{i}^{N} w_{i,j}(d_i + r_{i,j}) \tag{1}$$

where di is the position of the i-th descriptor point, $r_{i,j}$ is the vector between the i-th descriptor point and the j-th joint, and $w_{i,j}$ is the weight of each vector whose value fades out according to the length of $r_{i,j}$ and its angle with respect to the normal vector at the descriptor point. See the Appendix 1 for the calculation of the weights.

Rather than storing the initial trajectory as a collection of joint positions pj, we store, for each body, a reference to the descriptor points d_i along with the coordinates of $r_{i,j}$ and the weights $w_{i,j}$, because it encodes the spatial relations between the body and the environment, i.e., the context of the scene.

When an object or scene is deformed, the descriptor points move rigidly with respect to the surface, such that the directions of the relative vectors preserves the angle with respect to the normal vector at the position it is attached to the surface. Keeping the descriptor and relative vectors invariant corresponds to maintaining the spatial relationship between the character and its environments. In general, keeping the invariance is not possible, and we rather adapt the character posture to minimize the difference of the relative vectors between the new joint position and the descriptor points, while preserving the rigidity of the character (i.e. satisfying the bone-length constraint)

$$\min_{p_0 \cdots p_n} \quad \sum_{i=1}^{n} \left(p_i - \sum_{j} w_{i,j}(d_j + r_{i,j}) \right)$$
$$\text{s.t.} \quad ||p_i - p_{i-1}|| = l_i \quad \forall i$$

where l_i is the length of the body segment i.

As a result, a new character posture is obtained that integrates the deformations of the environment. In [Al-ashqar et al, 2013] a dedicated inverse-kinematics

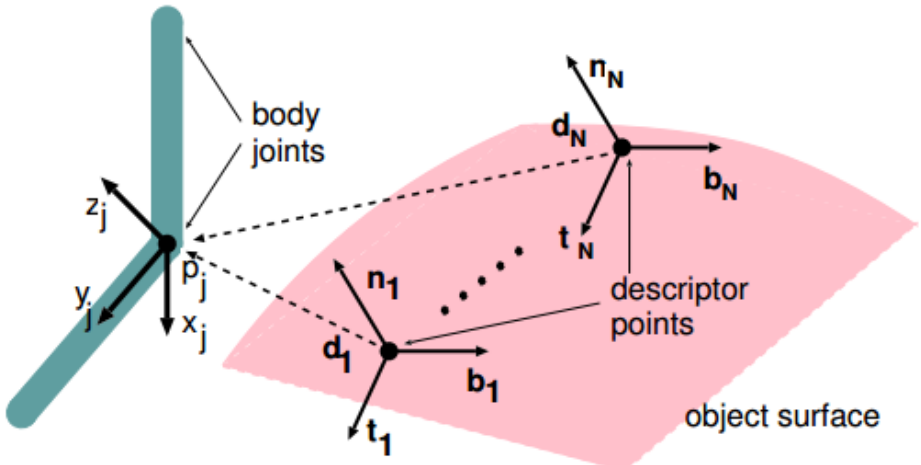


Fig. 2. The joint positions are described as a weighted sum of relative vectors originating from descriptor points embedded in the environment.

scheme, derived from [JAKOBSEN, 2001] is proposed to solve this minimization problem. We directly apply it here and compute relationship descriptors between the character and the template object designed for that specific motion.

To adapt the motion for novel objects or scenes our overall mission is then to find a fitting of these descriptors to the novel geometry in order to transfer the motion whilst preserving the interaction context. First we must co-analyze the template and novel objects.

4.2 Object co-Analysis

In order to adapt a motion from a template object or scene to a novel one, we need to initialize a correspondence between the two. Finding object correspondences is a highly difficult and open problem still under research due to the vast amount of varying semantics and criteria used to determine it. No absolute correspondence solution exists for all cases and in the case of interaction it's necessary to guide the correspondence with some initialization. For this we use co-segmentation.

The strength of our method is that it manages to adapt motions from one object to another even when the topologies of these objects largely differ and it achieves this continuously across all the frames of the animation. Furthermore we do not rely entirely on the object segments. As little as two or three corresponding segments are enough to initialize a correspondence because we

Fig. 3. A series of co-segmented gun objects.

augment the system by using open-space features that encode spatial relations of a point in space relative to the surface.

Given a template and a novel object, we must perform a co-analysis to simultaneously co-segment objects of the same class to initialize a coarse correspondence. For labeled scenes there is no further work to be done since the labels are known beforehand and each object is classified and considered as an entire segment on its own, i.e. a chair is one segment, a table is another etc.

For single object examples we use objects which were co-segmented by techniques in [Wu et al, 2013] as an external module to achieve the co-analysis. The method co-segments objects in an unsupervised manner using affinity aggregation spectral clustering. The method begins with a per-object segmentation of each shape in the input set followed by an extraction of shape descriptors for the initial sets of segments. The segments are then clustered using diffusion maps based on the descriptors. Finally a statistical model is built for each cluster which is used to obtain the final labeling of the shapes in the set. See Fig. 3 and Fig. 4 to see objects which were co-segmented by [Wu et al, 2013].

Once the objects of the same class are co-segmented we then align them. To do so we use simple rigid ICP to find a single transformation matrix that aligns the novel object to the template one such that the distances between the corresponding segment centers are minimized. For static objects, a single rigid

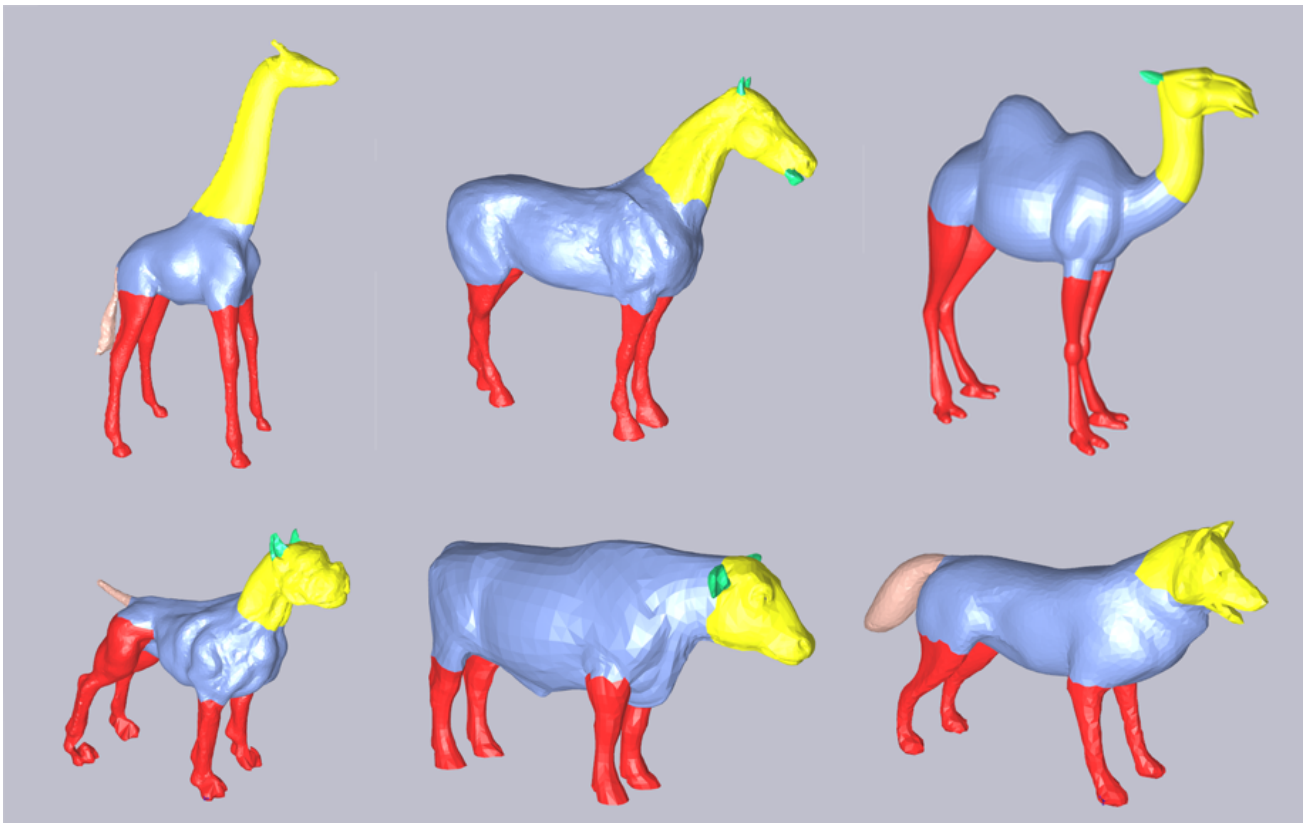


Fig. 4. A series of co-segmented animals, taken from COSEG database [Wu et al, 2013].

alignment is enough. For objects that move over the course of an animation, after aligning the objects in the first frame, we then transfer the transformation matrix which animates the template object to the novel object such that the two are aligned over the course of the animation.

4.3 Computing the Open Space Feature

In order to transfer the motion from the original to the new object, we must first encode the interaction to guide the transfer. Our philosophy is that we should adapt the character's posture around the novel object in such a way that we ensure the spatial relations between the character joints and the new mesh surface are as similar as possible to what they were relative to the template object whilst at the same time minimizing the deviation of the original posture to preserve the self-spatial relations between the character's joints.

We define as SId the vector of segments ids contained in the object. We consider only the intersection of the corresponding segment Ids in both the template and novel object discarding any segment that isn't indexed in both as such:

$$SId = SId_{template} \bigcap SId_{novel} \tag{2}$$

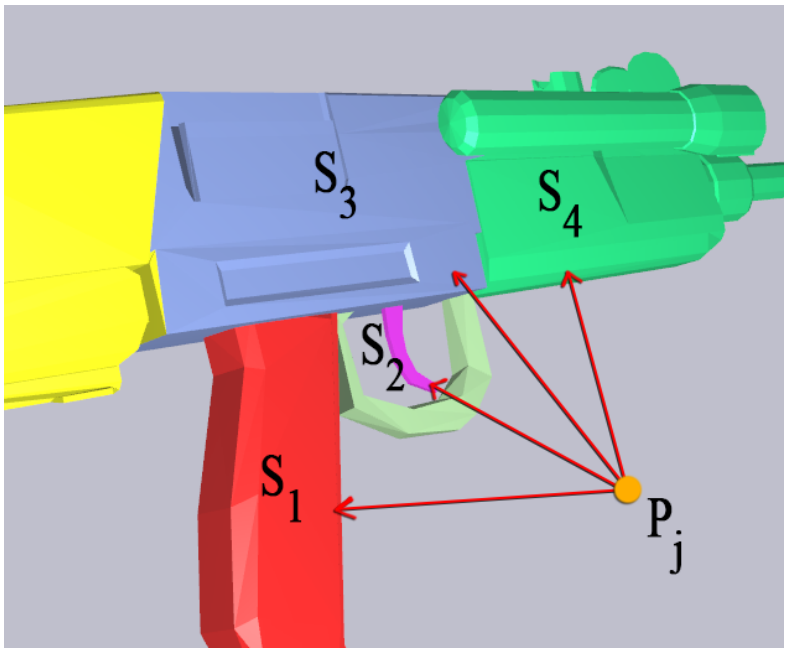


Fig. 5. Distances to segments from a point Pj.

First we define open space features which will guide our posture transfer. To encode the rough spatial relations between the character joints and the mesh-surface in the original interaction, we compute a vector Dj of closest distances d_{js} between each joint j and each segment s in $S_{template}$ of the template object.

$$D_j = \{d_{j1}, d_{j2}...d_{jn}\} \tag{3}$$

For all joints where j is a joint on the character, d is the closest distance from joint j to a segment s, and n is the total number of segments indexed by SId (see Fig. 5).

Although the distance to segments is a strong initial open-space feature, we need an additional feature to more finely capture the spatial relations between the geometry of the mesh and the character joints for further disambiguation purposes in matching. To achieve this, we use what is was proposed by [Zhao, 2014] as the $Space\ Coverage\ Feature$.

4.4 Space Coverage Feature (SCF)

The space coverage feature (SCF), effectively quantifies the relationship between a point in open space and a nearby object. It is proposed by [Zhao, 2014] and we recall it here for completeness.

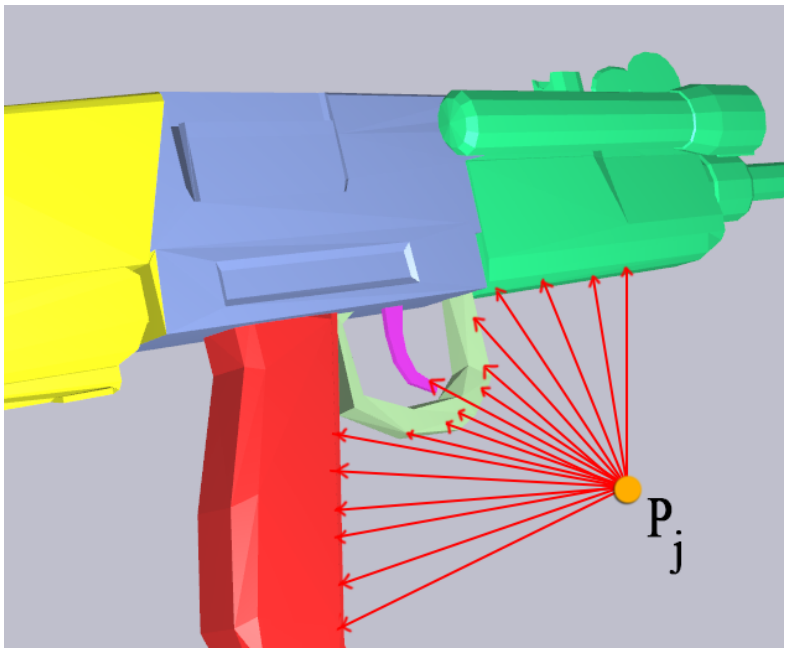


Fig. 6. Rays cast from point P_j to surface to compute space coverage feature.

The key of the SCF feature is to encode the geometry of the open space around objects in the frequency domain using spherical harmonics; a similar feature has been applied for describing object geometry but not for relationships.

Given a point P in open space, a spherical depth map is computed at P. We define a unit sphere centred at P with an orientation determined by the vector from P to the nearest point on the surrounding objects. We sample the distance d from P to all surrounding surfaces. Rays are cast (see Fig. 6) from P in a set of directions obtained by uniformly sampling n points along the latitude and longitude in the local coordinate frame of the sphere, for a total of $n \times n$ rays. We set n to 30 in our experiments. If a ray does not hit an object, d is set to infinity. This approach is inspired by [Shapira et al, 2008] where the rays are cast within the object to produce a feature of the object shape; here we cast from outside to produce a feature of the open space between objects.

Next, we define a discrete spherical function called the *volume diameter function* for P as

$$F_{vdf}(i,j) := \left\{ \frac{d^{min} + e}{d(i,j) + e} \;\middle|\; 0 \leq i,j \leq n-1 \right\}, \tag{4}$$

where i, j are the ray indices along the longitude and latitude directions, respectively, d^{min} is the minimum distance among all rays and used for normalization, and e is an offset whose value is set to the mean of all non-infinite distances. We use e to keep F_{vdf} descriptive even when P is sampled at the surface of the object. Note that the value range of F_{vdf} is $[0, 1]$ due to the normalization, which

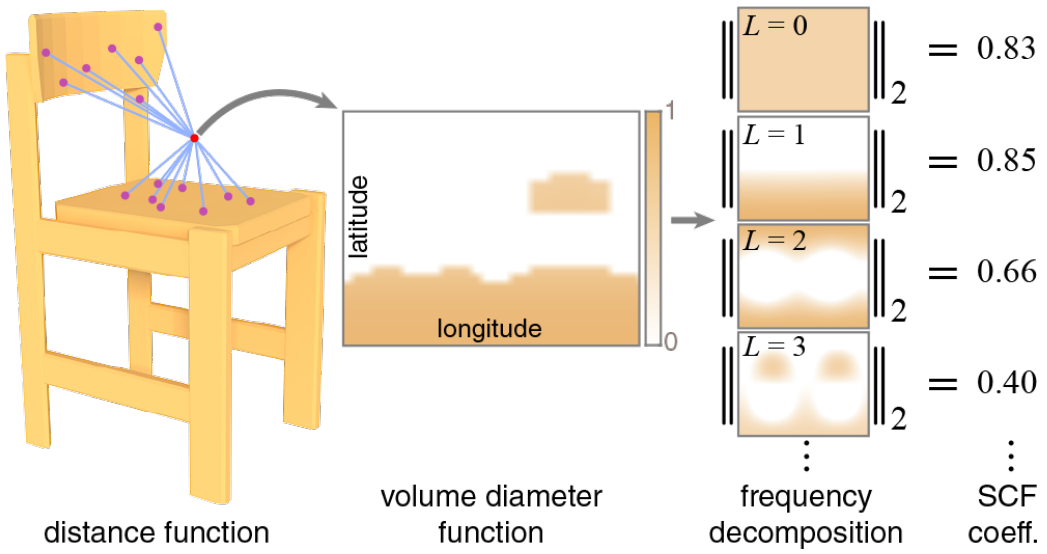


Fig. 7. Space coverage feature explained - diagram from [Zhao, 2014].

makes the feature scale invariant. Infinite ray results in a value of 0, while the ray with minimum distance gets a value close to 1.

The SCF feature vector is the rotation-invariant power spectrum of the volume diameter function. We first compute the spherical harmonics expansion $F_{vdf} = \sum_{l=0}^{\infty} \sum_{|m|\leq l} a_{l,m} Y_l^m$, where $a_{l,m}$ are the spherical harmonics coefficients of F_{vdf} and Y_l^m are the spherical harmonics at frequency l (see Fig. 7). We then define the SCF feature for a joint j relative to the template object geometry as:

$$SCF_j(F_{vdf}^{template}) = \{a_0, a_1, \ldots, a_n\}, \tag{5}$$

where a_l is the power of the function in frequency band l:

$$a_l = \| \sum_{|m|\leq l} a_{l,m} Y_l^m \|_2 = \sqrt{\sum_{|m|\leq l} (a_{l,m})^2}. \tag{6}$$

Note that the last equality holds due to the orthonormality of spherical harmonics. In practice, we use SpharmonicKit [Kostelec spharmonickit, 2008] for computing the spherical harmonics coefficients.

Describing open space in the frequency domain makes our descriptor more robust to small variations in the geometry and topology of the open space.

4.5 Matching the Open Space Feature in New Object

With these features computed for each joint relative to the template object we have enough info to capture the interaction. We wish to now find a similar arrangement around the novel object to which we are to transfer the interaction motion.

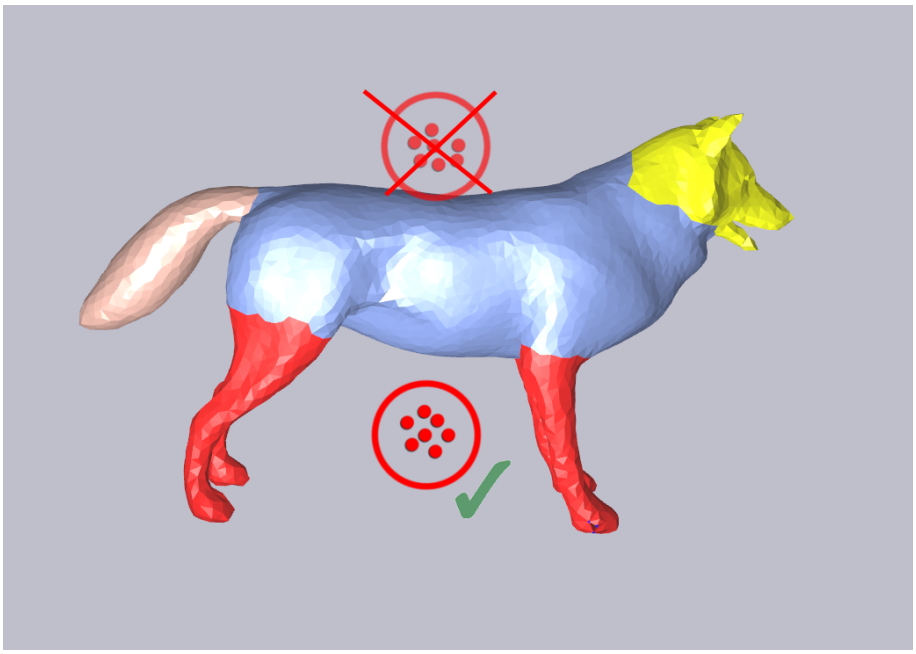


Fig. 8. For right ankle joint, two clusters of points have similar distance to segments features for novel dog object. Can be disambiguated using the space coverage feature for knee joint to eliminate false cluster.

Assuming we have a new object which is co-labelled or co-segmented, we must find candidate open-space points which correspond well to the joint positions around the template object or scene. However there is no analytic way to find these points and there are several possible candidate points which share similar spatial-relation features. In order to find these candidates we must sample the open space.

We uniformly and densely sample, with some randomization, the open-space around the new object with a step distance defined as half the smallest bone-length of the character around the new rigidly-aligned object. We only consider open-space points which are closer than the character's arm length to the surface(s) and discard the rest, especially as we're particularly interested in contacts although we're not limited to interactions involving them. Anything further isn't meaningful in the interaction as it's too far away and we don't attempt to fit joints which are farther than this as they will adapt based on self spatial-relations.

For each open-space point we compute the vector of distances from that point to each segment on the new object or scene. We define as D_o the vector of distances d_{os} between an outer-space point o and segments s in S_{novel} of the new object, indexed by SId.

$$D_o = \{d_{o1}, d_{o2}...d_{on}\} \tag{7}$$

Finally we also compute the space coverage feature SCF_o for the open space point o relative to the novel geometry as we did between the joint and template object.

$$SCF_o(F_{vdf}^{novel}) = \{a_0, a_1, \ldots, a_n\}, \tag{8}$$

At this point we have established corresponding segments between the template and novel objects, rigidly aligned them based on the segment centers, and have computed outer-space features to encode spatial relations. Given this info, we can now warp a posture in order to fit it to the new object based on these spatial-relation features.

For each joint j in a selected pose, we wish to find a list of candidate sample points around the new object to which it corresponds well. We first take the distance-to-segments feature for the joint relative to template object then compare these to the open space point features relative to the novel object.

We compute the difference between the distance-to-segment vector D_j in the original object for a joint j, and D_o for each outer-space point o around the novel object by taking the L2-norm between them. The smaller the distance in the L2-norm the more similar the encoded spatial relation is at the segments level.

$$d_{jo}^{seg} = ||D_j - D_o|| \tag{9}$$

For each joint, we find the difference of its feature with all other open-space point features around the new object and sort it in descending order, the top of the list being closest similarity. We cannot take only the single top candidate as the final corresponding point for the joint to warp towards due to issues of possible anomalies, symmetry around the segments, etc, hence we must consider a cluster of potential candidate points. In order to achieve this we cull the samples and take only the top 5 % of candidates from the list hence a dense sampling initially is somewhat important.

$$A_j = \{d_{j1}^{seg}, d_{j2}^{seg} ... d_{jn}^{seg}\} \tag{10}$$

Each joint will have a small list of candidate points in the new object to which it can correspond to. To further disambiguate the results and remove more anomalies (see Fig. 8) we compute the spherical harmonics of these top candidate points to find spatial relations between them and the entire geometry

they surround of the novel object. For each candidate we take the L2-norm between the SCF coefficients.

$$d_{jo}^{SCF} = ||SCF_j - SCF_o|| \tag{11}$$

We sort the candidates again and take the top 40% of these culling the potential candidates again almost by half.

$$B_j = \{d_{j1}^{SCF}, d_{j2}^{SCF} ... d_{jn}^{SCF}\} \tag{12}$$

Our 5% and 40% thresholds are filtration parameters selected manually. They can be adjusted for different effect. We end up with an even smaller set of meaningful candidate points for which the joint in question has a high probability of corresponding with relative to the new object.

We define the final set of strong candidate points as C_j for joint j. For each o in C_j we define 2 weights.

$$w_{jo}^{SCF} = 1/||SCF_j - SCF_o|| \tag{13}$$

$$w_{jo}^{seg} = 1/||D_j - D_o|| \tag{14}$$

We normalize these weights across all the candidate points C_j for joint j, such that their sums equal to 1. With this normalization, we assume equal importance to the two features and take a final combined weight for each candidate point o in Cj as such.

$$w_{jo} = w_{jo}^{SCF} + w_{jo}^{seg} \tag{15}$$

Although we end up with a small representative set of potential candidates, we may still have an issue due to symmetry since our features are rotation independent. For this we must eliminate irrelevant points based on symmetry. Our candidate points naturally get densely clustered around a few regions of local maxima weights around the object mostly depending on the planes of symmetry regarding distance to segments (see Fig. 9). To identify these regions we simply group the candidates into their natural clusters, where the center of each cluster is approximately the candidate with the greatest local maxima weight in the region.

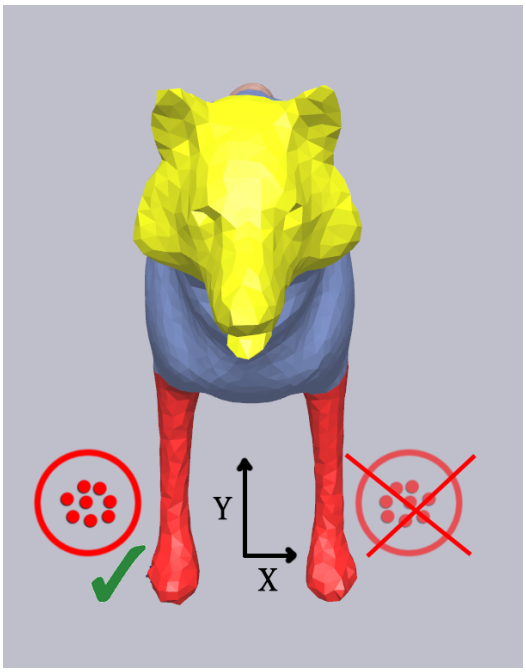


Fig. 9. Two clusters of points have similar features for novel dog object with respect to a joint due to planes of symmetry. Can be disambiguated using sidedness info in local axes for the rigidly aligned objects eliminating the false cluster.

If we have only one significant cluster then there is no further work to be done as there is no symmetry issue for that particular object. If we have significantly distinct clusters, the goal is to then select the most relevant cluster of points as our target. First we determine sidedness information. To do so, we consider the oriented bounded box for both the rigidly aligned template and novel objects. Since they're rigidly aligned we can determine their local axes and then evaluate the position of the center of each cluster relative to the local axes. We then select the cluster which is most similar in position in the local frame of the novel object's oriented bounding box as that of the joint in the local frame relative to the template object.

With a single cluster selected, we identify the corresponding open space target point for the character joint to warp towards as the single candidate point within the cluster which has the strongest weight w_{jo} of all which is effectively a point of local maxima weight.

4.6 Posture Adaptation to Novel Object

At this point we have identified target locations for each joint to warp towards. We now wish to warp the joints of the character towards their targets whilst preserving the rigidity of the bones. We do so as follows.

4.6.0.1 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:

$$f_j = k(\mathbf{p_j^{tar}} - \mathbf{p_j^{cur}}). \tag{16}$$

where k is an elasticity constant that is set to 1, $\mathbf{p_j^{tar}}$ is the target position of the joint determined as the target open space point, $\mathbf{p_j^{cur}}$ is the current joint position.

4.6.0.2 Integration Step:: We apply the the virtual forces computed by Eq. (16) to the joints using Verlet integration:

$$\mathbf{p_j^{new}} = \mathbf{p_j^{cur}} + d(\mathbf{p_j^{cur}} - \mathbf{p_j^{prev}}) + \frac{1}{2} fj \frac{1}{N_s^2} \tag{17}$$

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 $s_j = 1$).

4.6.0.3 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:

$$\Delta pj = \frac{s_k}{s_k + s_j} \frac{pj - pk}{\|pj - pk\|} (l^0 - \|pj - pk\|) \tag{18}$$

where pk 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.

4.7 Embedding Relationship Descriptors to the Novel Geometry

Once a single character posture has been fitted to the novel object, we then wish to update the descriptor positions to reflect this change in the posture and such that the entire motion across multiple frames can be adapted continuously rather than simply the one pose.

We now describe a method to apply rigid transformations to the relationship descriptors such that the motion reconstructed from the transformed relationship descriptors passes the posture computed by the fitting scheme described above, while preserving the context of the original scene. By importing the posture into the space of relationship descriptors, a smooth motion that seamlessly blends the original motion and the edited posture can be produced. By recycling the original relative vectors, the computed motion preserves the spatial relations between the body and the environment in the original motion. Moreover the planned posture will automatically adapt to later deformations of the environment.

It's worth noting that we don't enforce any condition that the descriptors must be planted strictly to the surface of the geometry in the scene, i.e. the descriptors may float slightly outside of the geometry of the new object. Enforcing such a condition would complicate things particularly for objects which have lots of holes and concavities or strange topologies in which case it could even be detrimental to project the descriptors strictly to the surface.

Cost function

Let us assume the position of the joints in the original posture p_j has been updated to $p_j{}'$ due to fitting the posture to the novel object. Our aim here is to modify the original attributes of the descriptors (describing p_j) so that they now correspond to the new position $p_j{}'$. Rather than directly modifying the relative vectors $r_{i,j}$, we impose to keep the length of the vector constant, by only adjusting its origin d_i and orientation R_i, such that the spatial relations between the environment and the body in the original scene is preserved. The main term of the cost function is then:

$$E_J = \sum_{i,j} \|p_j{}' - \sum_{i}^{N} w_{i,j}(d_i + R_i r_{i,j})\|^2 \tag{19}$$

In addition to Eq. (19), we consider the following regularization function that represents the difference in translation and rotation between the descriptor and the surrounding ones:

$$E_S = \sum_{i} \sum_{n_i} (\|d_i - d_{n_i}\|^2 + \|R_i - R_{n_i}\|_F^2) \tag{20}$$

where n_i is an index for the k-nearest neighbors (here we use $k = 3$), and $\|\cdot\|_F$ is the Frobenius norm. This regularization will tend to make the behavior of the collection of descriptors spatially coherent.

Finally, a second regularization term is added to penalize scaling/shearing and large updates from the previous iteration:

$$E_R = \sum_i \|R_i - R'_i\|_F^2 \tag{21}$$

where $'_i$ is the rotation matrix computed in the previous iteration, that is initialized as an identity matrix. This term alone will not manage to guarantee that $_i$ remains a rotation matrix after several iteration; but it is doable by a projection scheme in the minimization scheme, explained below.

We compute the optimal translation and rotation of the relationship descriptors by minimizing the following error function:

$$\min_{\substack{d_1 \ldots d_n \\ R_1 \ldots R_n}} \quad E_J + w_S E_S + w_R E_R, \tag{22}$$

where w_S and w_R are weights described below.

Minimization scheme

Eq. (22) is a non linear minimization problem that we solve by a dedicated iterative approach. At each iteration of the minimization, the problem is reorganized as a linear least-square problem in the form of:

$$E_{\{J,S,R\}} = \|A_{\{J,S,R\}} b - c_{\{J,S,R\}}\|^2$$

where the decision variable b is a vector that includes all the elements of d_i and R_i, i.e., $b = (d_1, R_1^1, R_1^2, R_1^3, ..., d_n, R_n^1, R_n^2, R_n^3)^\intercal$, R_i^m is the m-th row of R_i, and $A_{\{J,S,R\}}, c_{\{J,S,R\}}$ are the coefficient matrices and constant vectors from each error function. As a result, we can compute the b that minimizes an integrated error function $E_J + w_S E_S + w_R E_R$ by

$$b_o = \arg\min_b E_J + w_S E_S + w_R E_R = (A^\intercal A)^{-1} A^\intercal c \tag{23}$$

where $A = (A_J^\intercal, w_S A_S^\intercal, w_R A_R^\intercal)^\intercal$ and $c = (c_J^\intercal, w_S c_S^\intercal, w_R c_R^\intercal)^\intercal$.

As mentioned above, the resulting rotation matrix (computed by solving Eq. (23)) includes scaling/shearing. Therefore, we first apply singular value decomposition (SVD) to the computed matrix: $R_i = U\Sigma V$, and then extract the rotation matrix by $R'_i = UV$. This matrix is fed back into Eq. (21) for the next iteration, until E_J is below the threshold.

Regarding the weights of the terms, w_J and w_R are set to 1, while w_S is initially set to 100 and reduced 1 at each iteration, where the iterations are run 100 times in average. This is to impose more rigidity in the beginning of the optimization and gradually increase the flexibility such that the descriptors can adjust their locations as they converge.

5.1 Animal Riding

In our first demo we transfer an animated sequence of a character riding a quadraped animal (see Fig. 10). This interaction involves a number of movements such as the character waving the straddle, bouncing movement as well as striking the back of the animal to prompt it to move more quickly. Our template interaction involves a character riding a horse. The horse template object is co-analyzed against a number of different animals to co-segment them simultaneously using the unsupervised method of [Wu et al 2013]. In fact these animals are segmented in [Wu et al 2013] experiments and the data is taken directly from there. Using this segment information we sample a key posture of the character sitting on the animal and striking its back, encode the interaction features relative to the template, sample the open space of the novel co-segmented animal object and then warp the posture such that it fits well to candidate target points around the new object. We then fit the descriptors to achieve the results as seen for the pose in question.

To prove the use of the space coverage feature, we emphasize that using the distance-to-segments feature alone can cause ambiguities. Relative to the character's hip or ankle joint for example, there are points underneath the belly of the novel animal that can be equally weighted in the distance-to-segments feature as those points over the top of the back of the animal due to having similar distances to the segments, considering all limbs are a single segment in this demo. To disambiguate this, the space coverage feature will clearly favor the points over the top of the back of the animal since the distribution of geometry relative to that point are very different from those underneath the belly of the animal where there are lots of concavities from multiple directions due to surrounding limbs whereas those limbs can't be seen by the points on top.

Likewise we have symmetry issues. For the knee and ankle joints, strong candidate points cluster naturally around the two legs either side of the animal's body due to symmetry. Therefore using the rigidly aligned local axes we can disambiguate further and select the correct candidate cluster.

5.2 Gun Manipulation

In this demo we transfer an animated sequence of a character handling an assault rifle (see Fig. 11, Fig. 12), one of a series of co-segmented rifles which

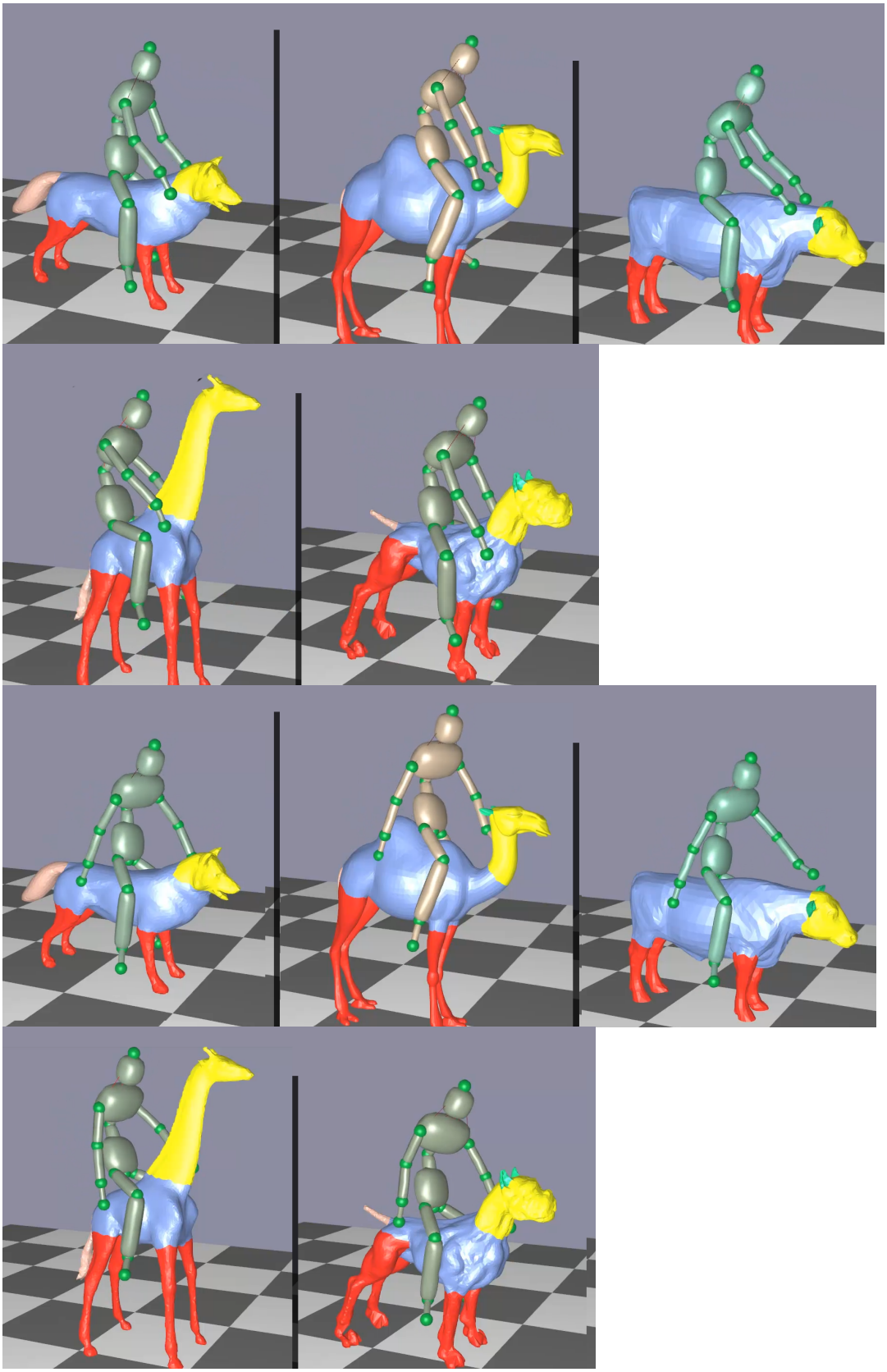


Fig. 10. Demo 1: Riding motion adapted to different co-segmented animals at different frames

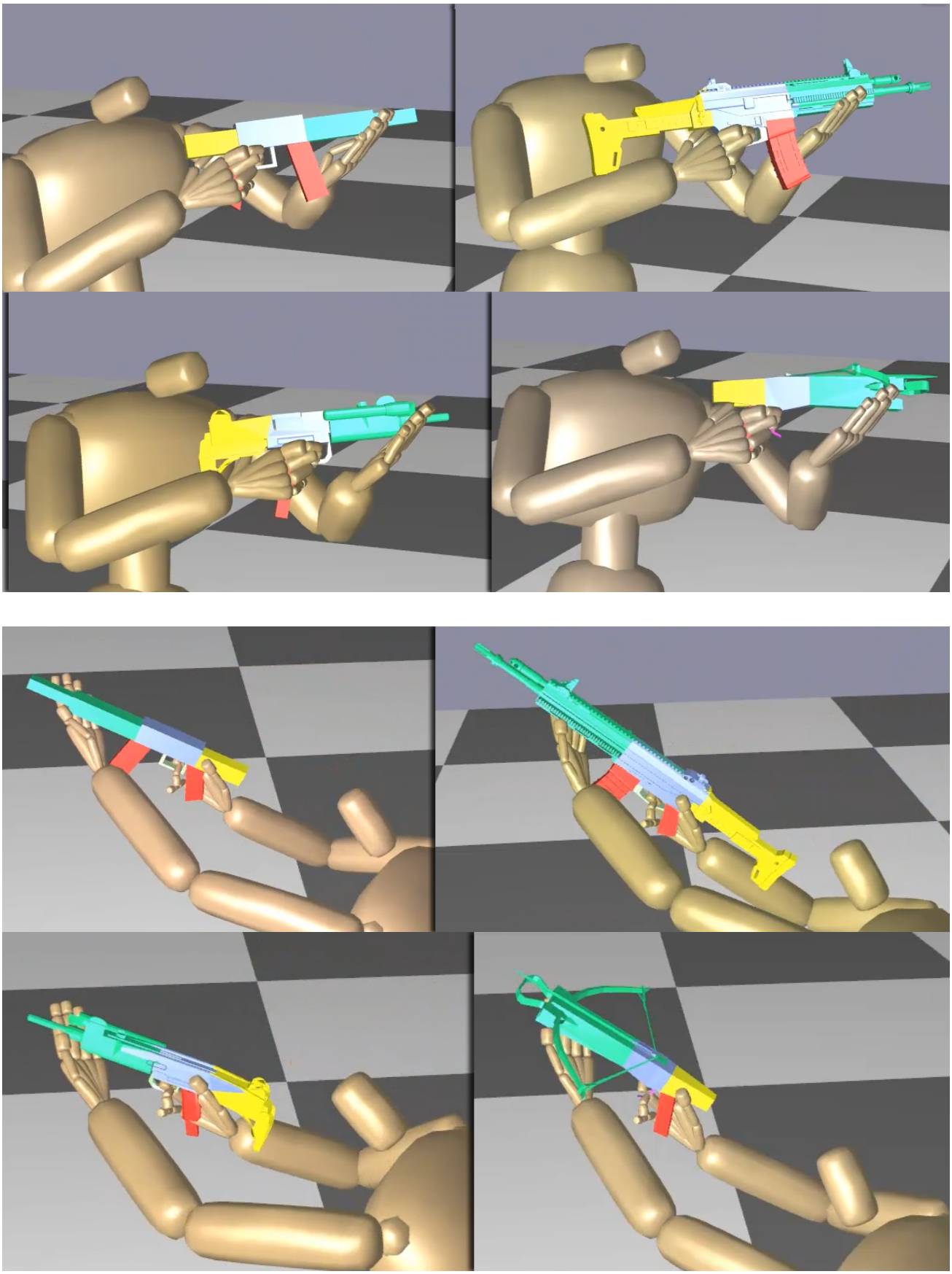


Fig. 11. Demo 2: Rifle manipulation motions adapted to different co-segmented guns at different frames

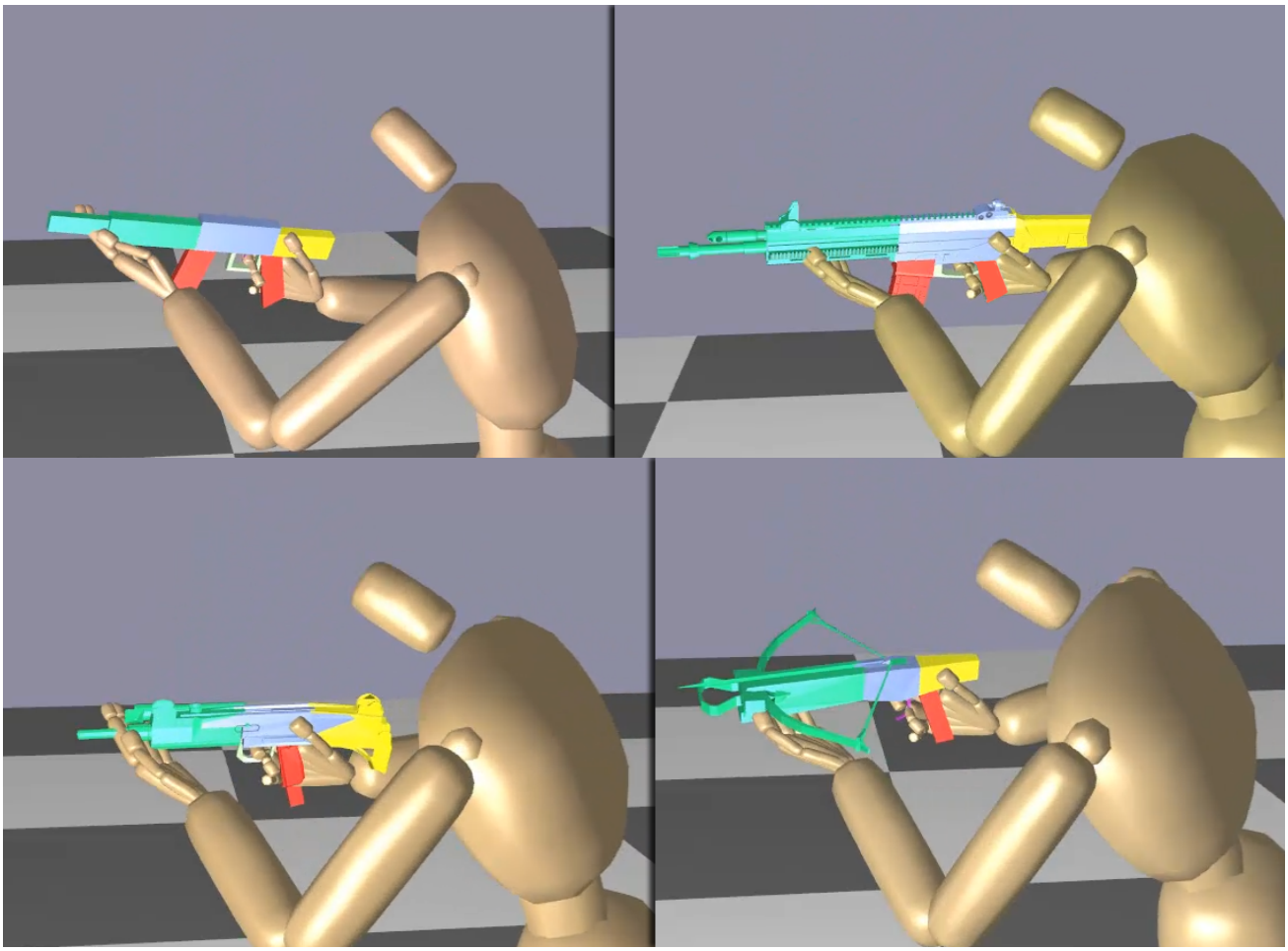


Fig. 12. Demo 2: Rifle manipulation motions adapted to different co-segmented guns at different frames

also include: an assault rifle, an uzi which is missing a second rifle grip, and a crossbow. With the exception of the assault rifle which shares all the segments of the template, the other rifles are all missing some segments contained in the template, and this is where using only the intersection of segments existing in both is important for all our feature calculations.

Like the animal demo, symmetry is an issue here too, hence determining sidedness information is important again. On the other hand there are a number of strong distinct candidate points for posture transfer of certain joints, namely near the trigger of the gun which are unique for both SCF and DSEG features relative to the fingers. This demo showcases the transfer of fine interactions.

5.3 Car Scene

In this demo we transfer an animated sequence of a character interacting with the components of a constrained car environment (see Fig. 13). The character is seated, manipulates the steering wheel and gear stick, and extends his body

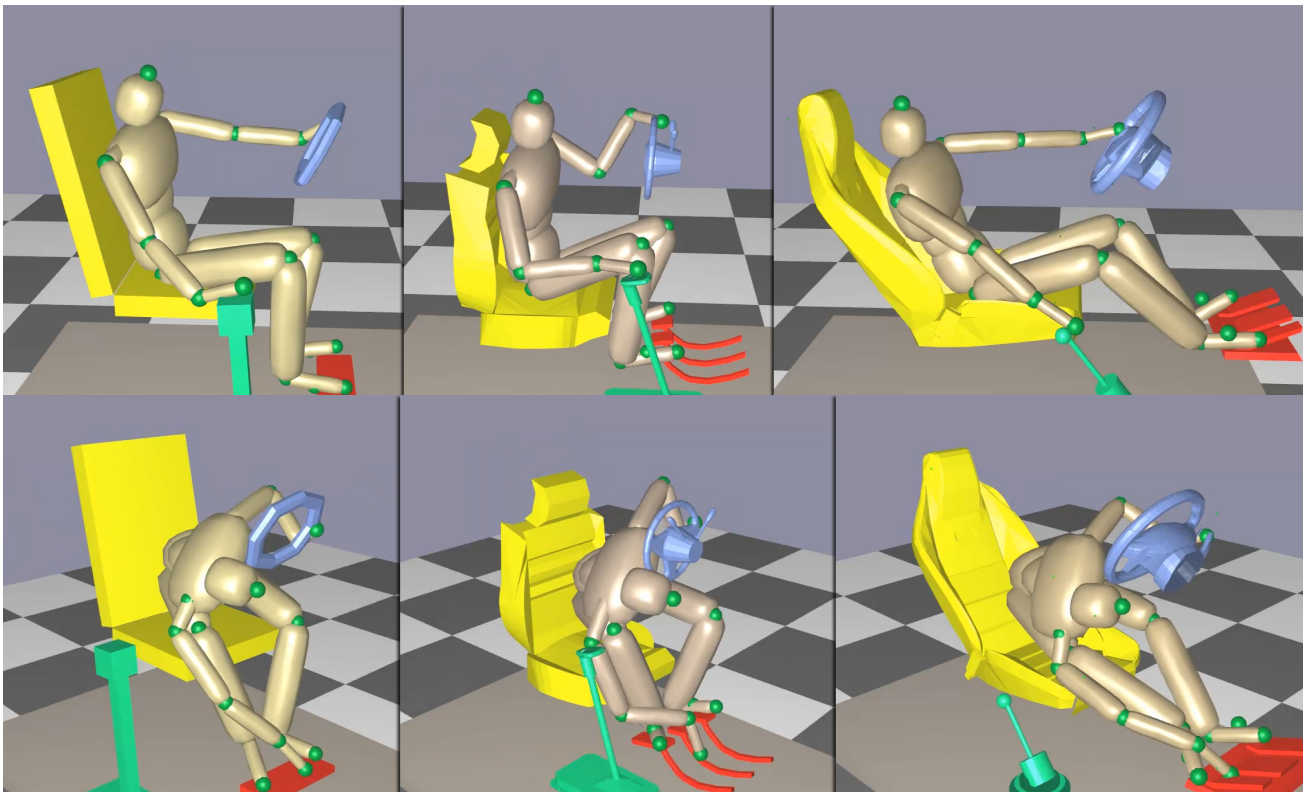


Fig. 13. Demo 3: Car interaction motions adapted to different co-labelled car scenes at different frames

and arms to pick up a mobile phone from the ground. The original object is a labelled car scene containing these labels: 'chair', 'gearstick', 'wheel', 'pedals'. The novel scenes are completely different in shape and topology but equally co-labelled. Using only this coarse high-level label information as initialization, the motion can be entirely mapped to a new car which shows the robustness of the system when faced with very coarse initialization. it's important to note that if only simple IK was used to adapt the motion, then not all interactions and spatial relations may be considered, such as the head relative to the wheel as the character picks up the phone from the floor which must be preserved while adapting to avoid collisions.

5.4 Desk Scene

In this demo we transfer an animated sequence of a character interacting with a labelled constrained desk scene (see Fig. 14, Fig. 15). The character walks towards the chair and seats himself whilst placing his hands upon two surfaces for support, a front desk and side table. We adapt this motion to novel co-labelled scenes whose components have similar spatial arrangements but completely different topologies. By updating the relationship descriptors, the entire motion including the character walking towards the chair is adapted, where the

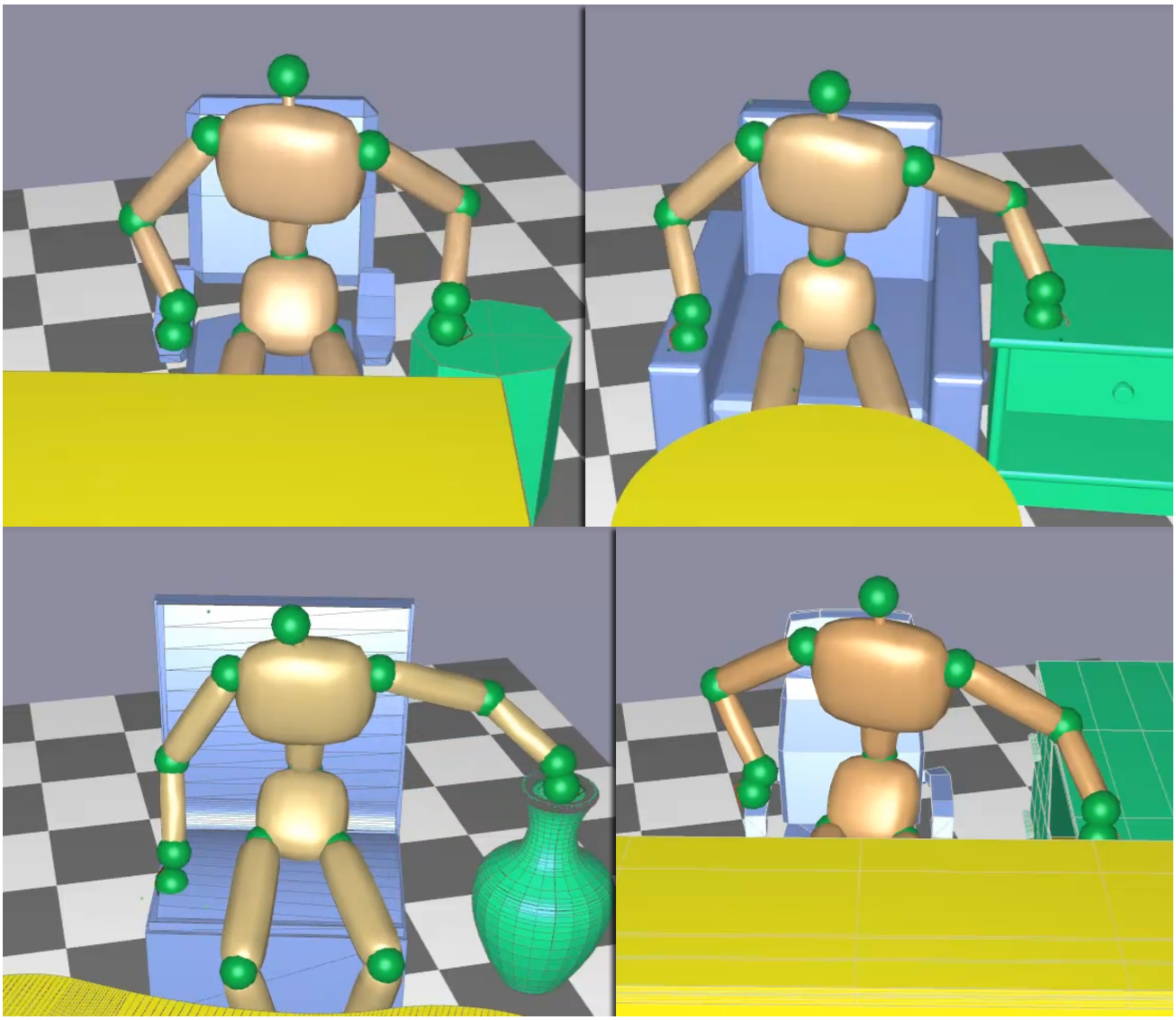


Fig. 14. Demo 4: Desk interaction motions adapted to different co-labelled desk scenes at different frames

approach is considered alongside contacts. Traditionally older methods would only adapt a single pose where there are strict contacts. without consideration for spatial relations during approach along a window of time.

6 Conclusion and Discussions

We proposed a novel effective way in which to adapt entire animations of interactions from template to novel objects of different shapes and topology with minimal initialization. Unlike previous methods where only a single posture is adapted, with our method an entire motion can be adapted continuously across all the frames of the animation. While the method is effective for several use cases there are a number of key limitations.

Limitations and Future Work

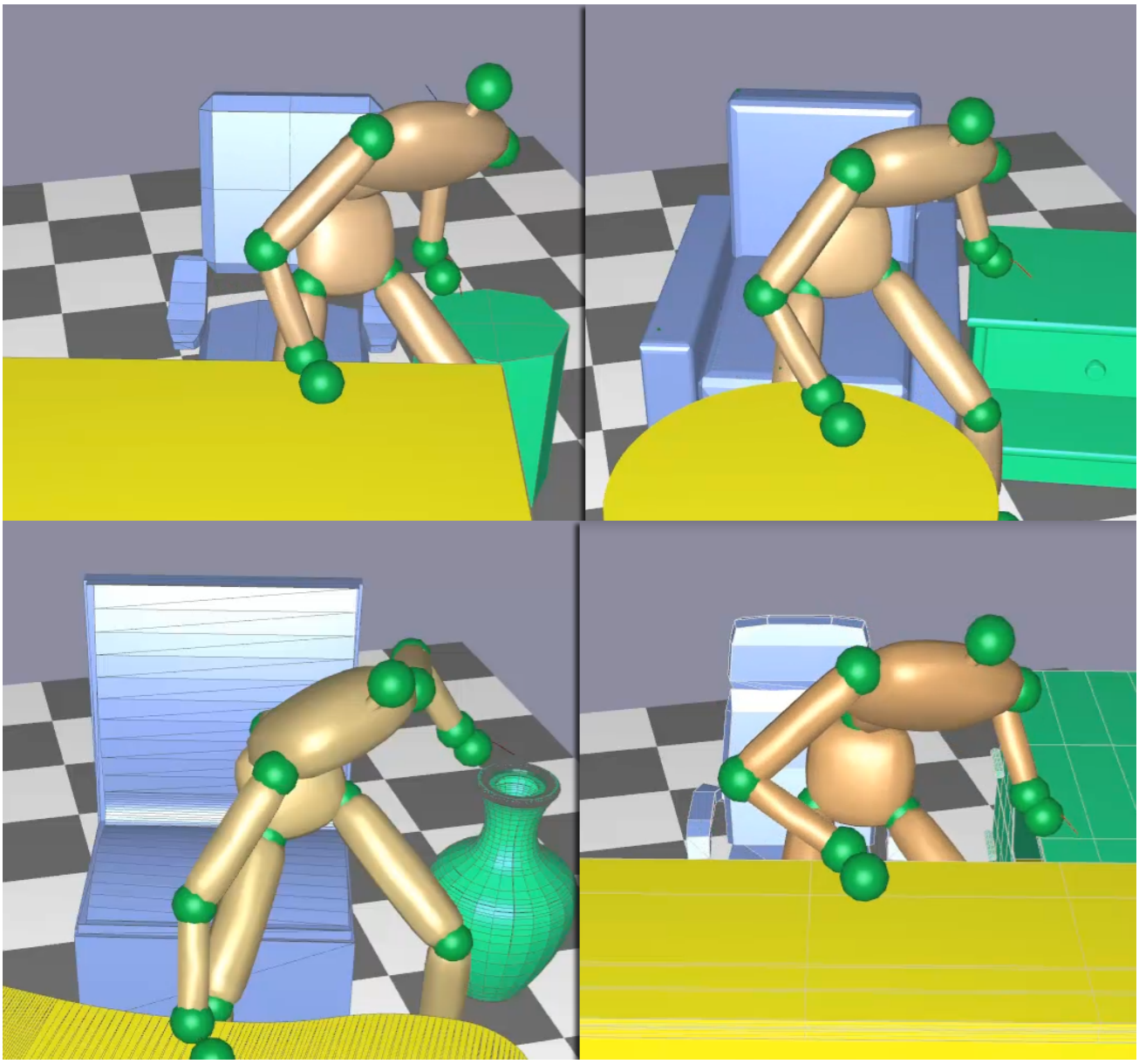


Fig. 15. Demo 4: Desk interaction motions adapted to different co-labelled desk scenes at different frames

For one, our system largely depends on the initial establishment of correspondence between objects of the same class to be correctly initialized in the co-analysis phase even if loosely. If the objects aren't correctly co-labelled the adaptation results will be entirely wrong.

Furthermore, we only use a single reference interaction to encode the open-space features for the original interaction such as SCF. This is largely limiting as it means that novel objects will have to have open-space features around them which are very similar to those around the original object otherwise the posture fitting won't be accurate. This limits the amount of variation we can handle in our novel objects. In order to be able to handle an even greater amplitude of variations for novel objects, we would ideally need to use a larger

amount of original interaction animations between the character and different template objects which would act as training data to guide our mapping to novel objects using a probabilistic model. This is a field for further work.

For two, the system only works when novel objects or scenes are introduced that are of the same overall class as the template object. If they're not the adaptation will fail. Again, determining this would require the use of databases of objects which are trained and classified beforehand, and an object classifier to evaluate whether new objects can actually be fitted or not. This, alongside the use of probabilistic methods of co-analysis and interaction encoding using lots of training data, would be a field of further work and a natural next step for the research problem, but isn't covered here. Otherwise all the above limitations hold.

Another key limitation is the fact that our method considers only spatial relation based features and is blind to physics related attributes of the objects such as mass, density, balance etc. and the adaptation is not necessarily momentum preserving. Furthermore it isn't guaranteed that after the relationship descriptors are adapted such that the motion passes through the new fitted posture that other poses during the animation are absolutely free of collisions and penetrations.

Comparison and Performance

As far as comparison to other systems, as we described earlier the previous work in this realm only map 'snapshot' poses to new geometry hence there's little way to qualitatively or even quantitatively compare the results of our methods except that our method enjoys the advantage of transferring an entire animation across its whole time window. On the other hand the goals are different in that the posture generation in previous methods motivate shape analysis whereas in our method the goal is the actual transfer of the entire motion which makes it a bit more unique in this regard and difficult to compare to existing work.

As far as performance, the online adaptation phase of the system is real-time. The offline pre-processing stage is where the expensive tasks lie. What this means is that if a novel non-segmented object is introduced on the fly in a real-time environment such as a game then performance can become problematic if it hasn't been already preprocessed offline beforehand. The most expensive part is the co-analysis of objects and the computation of the space coverage feature for sample points. Depending on the density of candidate points and of the object triangulation this can vary but all in all the full pipeline per object in our tests took around 30 seconds on average on one core of a Core i7 2.67GHz

CPU with 2GB of memory. Warping the posture to the candidate locations and optimizing the descriptors themselves to do the actual mapping barely take a second in practice and the bulk of the pre-processing time is in the co-analysis stage that produces the segments as well as computation of the space coverage feature for the sample points hence so long as this is pre-computed beforehand then everything else can be real time. More experiments are needed for a proper study into the performance and breakdown of the processing times for each stage.

REFERENCES

[1] RAMI AL-ASHQAR, TAKU KOMURA, MYUNG GEOL CHO: *Relationship Descriptors for Interactive Motion Adaptation* Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2013.

[2] XI ZHAO: *Spatial relationship based scene analysis and synthesis* Edinburgh University phD thesis, 2014.

[3] MANOLIS SAVVA, ANGEL X. CHANG, PAT HANRAHAN, MATTHEW FISHER, MATTHIAS NIENER: *SceneGrok: Inferring Action Maps in 3D Environments*. SIGGRAPH Asia, 2014.

[4] MANOLIS SAVVA, ANGEL X. CHANG, PAT HANRAHAN, MATTHEW FISHER, MATTHIAS NIENER: *PiGraphs: Learning Interaction Snapshots from Observations*. SIGGRAPH, 2016.

[5] ZIZHAO WU, YUNHAI WANG, RUYANG SHOU, BAOQUAN CHEN, XINGUO LIU: *Unsupervised Co-Segmentation of 3D Shapes via Affinity Aggregation Spectral Clustering, plus COSEG database*. Proceedings of SMI, 2013.

[6] VLADIMIR G. KIM, SIDDHARTHA CHAUDHURI, LEONIDAS GUIBAS, THOMAS FUNKHOUSER, *Shape2Pose: Human-Centric Shape Analysis*. ACM Transactions on Graphics (Proc. SIGGRAPH), 2014.

[7] G. FRITZ, L. PALETTA, R. BREITHAUPT, E. ROME: *Learning predictive features in affordance based robotic perception systems*. Intelligent Robots and Systems, 36423647, 2006.

[8] T. HERMAN, J. REHG, A. BOBICK: *Affordance prediction via learned object attributes*. ICRA, 2011.

[9] SUN, J., MOORE, J. L., BOBICK, A., AND REHG, J. M.: *Learning visual object categories for robot affordance prediction.*. The International Journal of Robotics Research, 2009.

[10] SUN, J., MOORE, J. L., BOBICK, A., AND REHG, J. M.: *Learning visual object categories for robot affordance prediction.*. The International Journal of Robotics Research, 2009.

[11] STARK, M., LIES, P., ZILLICH, M., WYATT, J., AND SCHIELE: *Functional object class detection based on learned affordance cues*. Computer Vision Systems, 2008.

[12] GRABNER, H., GALL, J., , AND VAN GOOL, L.: *What makes a chair a chair?* CVPR, 2011.

[13] GUPTA, A., SATKIN, S., EFROS, A. A., AND HEBERT, M.: *From 3D scene geometry to human workspace.* In IEEE CVPR, 2011

[14] JIANG, Y., KOPPULA, H. S., AND SAXENA, A.: *Hallucinated humans as the hidden context for labeling 3D scenes*. CVPR, 2013.

[15] JIANG, Y., LIM, M., AND SAXENA, A.: *Learning object arrangements in 3D scenes using human context*. ICML, 2012.

[16] JIANG, Y., AND SAXENA, A.: *Hallucinating humans for learning robotic placement of objects*. ISER, 2012.

[17] JIANG, Y., AND SAXENA, A.: *Infinite latent conditional random fields for modeling environments through humans.* RSS, 2013.

[18] P. KOSTELEC: *Spharmonic Kit - S2Kit*.
`http://www.cs.dartmouth.edu/ geelong/sphere/`

Chapter 5

Publication 3: [EuroGraphics 2016]

Character Contact Repositioning Under Large Environment Deformation

Steve Tonneau[1†], Rami Ali Al-Ashqar[2†], Julien Pettré[3], Taku Komura[2], Nicolas Mansard[1]

[1]LAAS-CNRS / UPS
[2]University of Edinburgh
[3]Inria Rennes

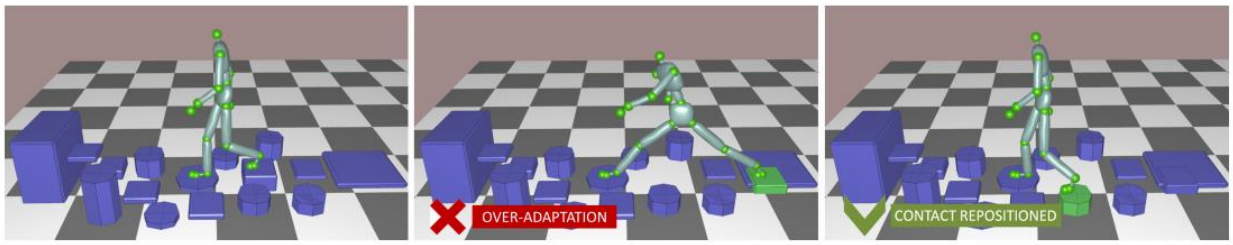


Figure 1: *Left: Reference motion. Middle: The user moves away one of the step stones. The motion is over-adapted based on the relationshi descriptors technique. Right: the contact is re-positioned, resulting in a more plausible motion.*

In all of our character-object motion adaptation demos to this point, we are limited to deforming the 3d objects involved in the interactions within a limited local window or amplitude. If the deformations are too extreme then the animations will fail to adapt in plausible ways with respect to balance, joint limits, collisions, and reachability. Extreme object deformations can take more than one form, from over-scaling to extreme stretching and repositioning of object components within the scene. In a more extreme sense, what happens if we deleted an entire object component within the scene, particularly one with which the character makes physical contact with a limb? In that sense, how do we adapt motions to extreme changes in the objects within our scene?

Up to this point we only ever considered spatial relationship descriptors within our interactions. Spatial relationships alone are not enough however to

deal with deformations of extreme amplitudes, and greater measures must be taken to deal with the aforementioned cases. It's therefore worth noting that characters can only be physically suspended and balanced within a scene based on physical contacts between the character's end effectors and limbs with surfaces in the scene.

Not only are spatial relationships changing within an interaction but so are contacts. Therefore in order to make significant changes to the adapted character in the case of extreme deformations, we would need to reposition contacts entirely in cases where adapting spatial relations alone isn't enough. For example, if there is a panel in the original scene with which the character establishes contact by placing their hand upon it, then if we delete the topology of that panel entirely or apply extreme repositioning to it, the character would either have a hand floating in the air without contact, or would be extremely overstretched violating joint limits, balance, and causing collisions etc. The only way in which to resolve the issue in such a case would be to reposition the contact of the hand to an entirely different surface location within the scene in such a way that the new resulting contact and posture don't violate those properties. The repositioning of the contact would then have to be propagated across all the frames of the animation to maintain continuity between frames. This motivates the introduction of contact descriptors which will be introduced in the next paper and will be used alongside relationship descriptors to achieve motion adaptation with extreme scene deformations.

The next attached paper is a collaboration with LAAS-CNRS France, and takes my method further by combining relationship-descriptors with procedural animation techniques to make animations adaptable to even more extreme deformations by repositioning contacts when limits are violated in the adapted motion and then propagating the change across all the animation frames by performing an optimization within the relationship descriptors space to change the contact whilst preserving the continuity and style of the interaction motion.

I am a co-first author alongside Steve Tonnaue from LAAS. This means we are both equally responsible for ideation, exploring related work and literature etc. I however generated all the results seen in the publication and performed all the relationship descriptor based work and optimizations to achieve the style and continuity aspect. I also performed the majority of the integration work since I used Steve's modules and did all the data transfer back and forth to generate the results. Steve's modules that I integrated are responsible for the contact descriptors, validation, and generating new repositioned contacts etc. as well as the root trajectory adaptation, and the mocap was captured in the LAAS lab.

EUROGRAPHICS 2016 / J. Jorge and M. Lin
(Guest Editors)

Volume 35 (2016), Number 2

Character contact re-positioning under large environment deformation

S. Tonneau, R. Al-Ashqar, J. Pettre, T. Komura, N. Mansard

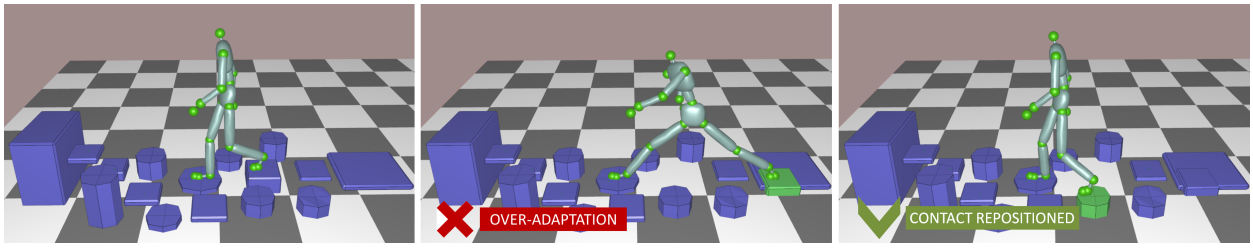


Figure 1: *Left: Reference motion. Middle: The user moves away one of the step stones. The motion is over-adapted based on the relationship-descriptors technique. Right: the contact is re-positioned, resulting in a more plausible motion.*

Abstract

Character animation based on motion capture provides intrinsically plausible results, but lacks the flexibility of procedural methods. Motion editing methods partially address this limitation by adapting the animation to small deformations of the environment. We extend one such method, the so-called relationship descriptors, to tackle the issue of motion editing under large environment deformations. Large deformations often result in joint limits violation, loss of balance, or collisions. Our method handles these situations by automatically detecting and repositioning invalidated contacts. The new contact configurations are chosen to preserve the mechanical properties of the original contacts in order to provide plausible support phases. When it is not possible to find an equivalent contact, a procedural animation is generated and blended with the original motion. Thanks to an optimization scheme, the resulting motions are continuous and preserve the

style of the reference motions. The method is fully interactive and enables the motion to be adapted on-line even in case of large changes of the environment. We demonstrate our method on several challenging scenarios, proving its immediate application to 3D animation softwares and video games.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

One of the long standing objectives of computer animation is to synthesize plausible motions for characters interacting with complex environments. Plausibility is usually obtained by animating characters using motion capture data. Unfortunately, this process lacks flexibility, as the content of a motion capture is fixed. To overcome this limitation, several motion editing methods have been proposed. They perform geometric adaptations of the reference motion to small variations of the environment. In general however, large variations result in unnatural deformations, because of collisions, joint limit violations, or loss of balance in the resulting animation.

Our objective is to tackle these large deformation scenarios, and thus to extend the validity domain of motion adaptation techniques. The hard issue on which we focus is the re-positioning of contacts that become invalidated by the environment edition, while preserving the style of the reference motion. For instance consider a motion where a character stands up from a chair by creating a contact with his hands and the armchairs. How to play the motion in a context where the chair has no armchairs? In most cases, simply removing the contacts is not sufficient because of balance issues. The invalid contacts either have to be re-positioned, or the motion adapted to respect balance constraints.

Our approach to the problem is incremental. We start from a state of the art motion editing method [AAKC13], and use it to adapt the motion to small deformations of the environment. While the motion is adapted, we automatically determine whether the deformation is acceptable regarding collision, balance, and joint limits, by considering each limb in contact individually. This is done efficiently thanks to a new method we propose, based on the reachable workspace of the considered limb.

If one criterion is not verified, the contact phase is invalidated, and a new contact location

/

is computed. The search for new contacts is based on a fast heuristic-driven search among possible surfaces of the environment. Among the candidates, the new contact must verify the mentioned conditions. Regarding force exertion, the new contact must also contribute to the overall motion in the same manner that the original contact did. This is verified with a new heuristic, the force actuation profile. In a last step, if contact re-positioning fails, we use a motion planner to adapt the root trajectory to the environment. The planner is biased to preserve the style of the original motion and to minimize contact invalidation.

These adaptations are smoothly blended within the original motion, thanks to a local optimization method that minimizes the differences between the original and adapted motion.

Our contribution can be summarized as follows:

- An automatic and efficient method to detect unnatural motion adaptations. The method is generic and can be applied with any motion editing technique.
- An interactive contact re-positioning method that handles large deformations of the environment.
- A low-dimensional sampling based method able adapt a root trajectory to large deformations when contact re-positioning fails.
- A novel optimization scheme that preserve the continuity and style of the adapted motion.

In the remainder of the paper, we first situate our contribution with respect to the related work in Section 2. We then provide an overview of the method in Section 3. One section is then dedicated to each step of the method (Section 4 to Section 7). Section 8 presents our experimental results, and provides support for a discussion on the method in Section 9. Section 10 concludes the paper with a discussion on current applications and future works.

2. Related work

This paper presents a procedural approach to adapt continuously an existing motion captured example. Our work is thus related to several approaches: physically-based procedural animation, data-driven motion synthesis and motion adaptation.

Physically-based Procedural Animation These methods aim at controlling a character using physical models. They often rely on finite state machines and PD control to guide the motion [Tho91, vdPL95, YLvdP07]. Designing such controllers for complex environments is not trivial: the required transition rules and states are less obvious than for cyclic

walking patterns, and much more numerous. To address the limitation, methods based on reinforcement learning have been proposed, though they suffer from a huge amount of memory usage and precomputation. [PBvdP15]. Contact invariant control [MTP12], on the other hand, does not require designing a state machine. A set of contact variables is considered within a constrained optimization problem, such that contact phases are automatically discovered along the generation of the motion. However, the resulting motion is not dynamically consistent, is sensitive to local minima, and requires several minutes of computation. Interactive motion synthesis can be achieved using sequential Monte Carlo approaches [HET*14] or particle belief propagation [HRL15]. Although they provide an exciting direction of research, as of today the synthesized motions lack naturalness, especially for motions involving close environment interactions.

Data-driven motion synthesis Using motion capture data is the most straight-forward way to synthesize plausible human motion. Simply replaying the captured motion lacks flexibility. A number of techniques such as motion blending [WP95, RBC98] and inverse kinematics [BB04] are proposed to increase the flexibility of the captured data. These methods enable adapting a motion by interpolating or warping motions to meet user-defined constraints. Motion graphs couple such approaches with a data structure where nodes are motion clips, connected if a continuous transition exists between them. This enables synthesizing complex motions that result from the concatenation of the traversed nodes [KGP02, LCR*02, AF02].

Various data-driven approaches based on motion capture data are proposed to control characters in novel environments. Methods based on PRMs [CLS03, PLS03] are effective to evaluate the areas where the characters can move into and find the set of available movements at such areas. Kim et al. [KHKL09] propose an approach to spatial-temporally edit the motion while preserving the context of interactions, by inserting/removing gait cycles when the trajectories are stretched/shortened, and curving the trajectories by Laplacian editing when it is bent. Safonova and Hodgins [SH07] parameterize the movements by interpolating the motions in the same category, and compute the sequence of movements by dynamic programming. Min and Chai [MC12] introduce such a framework in the probabilistic domain. Levine et al. [LLKP11] extend the idea of sample-based planning to the temporal domain, and compute the optimal series of movements even taking into account

/

dynamic obstacles. Lee et al. [LCL06] parameterize and associates movements to spatial building blocks called motion patches, and produce large scale scenes by stitching them together.

Motion Adaptation of Close Interactions Highly-constrained cases may however result in failures of data-driven methods. For instance, when entering a car, maintaining the contacts and avoiding collisions are contradictory objectives due to the proximity of obstacles, thus hard to satisfy when the geometry is modified. Such scenarios may require to use importance-based techniques [SLSG01] to take into account the relations of all adjacent objects. Ho et al. [HKT10] represent the context by producing a volumetric graph between the character's joints and the environment. The motion is adapted using Laplacian mesh deformation [SCOL*04], that preserves the local geometry of the interaction. Relationship descriptors by Al-Ashqar et al. [AAKC13] propose a frame-by-frame adaptation scheme, that fits the requirements of real-time applications. The limitation of these methods is reached when the collision avoidance and contact maintenance constraints become incompatible (due to a large deformation), so that contact re-positioning becomes necessary. Generating automatically a replacement contact is possible [TPM14], but the existing method does not consider motion continuity, nor the style of the motion.

State of the art analysis Generating plausible motions that involve close interactions is a recognized open issue. Optimization-based techniques that take into account physics requires off-line computation in this context. Data-driven methods give plausible results thanks to local deformations of a reference natural motion. However the amplitude of adaptations that can be applied to the reference motion is a crucial question to decide which method to use. In close interaction scenarios, especially when concavities exist, this amplitude is low due to the limited space available for the character. In this context large deformations require contact re-positioning, because collisions prevent contact maintenance, or because the contact geometry has been deleted! For existing methods to be applied in the context, two important issues have to be solved: the automatic detection of these failure cases, and the automatic contact re-positioning that solves them while preserving motion continuity. In this paper we demonstrate this feasibility: we extend the work of [AAKC13], by addressing these two issues.

/

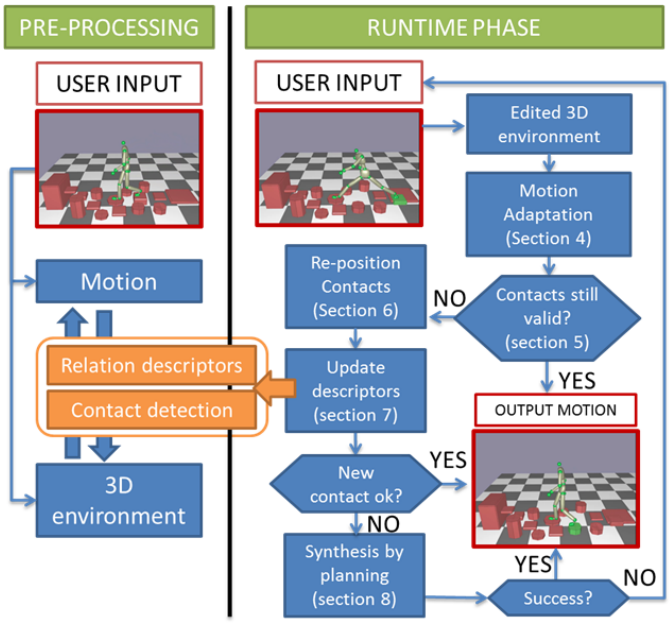


Figure 2: *Work-flow of the method.* ***Left****: an automatic preprocessing analysis of the motion allows to describe relationship and contact descriptors.* ***Right****: Depending on the deformation, the interactive phase can take up to three steps, from motion deformation to larger contact and body re-positioning.*

3. Overview and organization

Our method consists in one preprocessing phase, and one interactive phase. The workflow is illustrated in Fig. 2. The interactive phase is designed to provide an appropriate answer to the environment modifications: large adaptations are only applied if small adaptations fail first. The method is fully automatic.

During preprocessing, a static analysis of the original animation is performed. It extracts the features that describe the interactions with the environment: information regarding the proximity between each body part and obstacles is stored (relationship descriptors); contact phases are identified, and characterized regarding their contribution to the motion (contact descriptors).

At runtime, if an object of the environment is deformed, the character motion is adapted using relationship descriptors. The importance of the adaptation depends on the proximity

/

between the character's bodies and the object in the original motion, as proposed by Al-Ashqar et al. [AAKC13].

Our contribution extends this method. In parallel, efficient routines check a set of kinematic and dynamic conditions that ensure that the adaptation remains plausible. When a condition is violated, the motion is invalidated, and locally replaced with an equivalent one. Specifically, new contact configurations are chosen to provide force actuation capabilities similar to the original ones. If contact re-positioning fails, the last step consists in locally re-positioning the root trajectory of the character. This is achieved using a random sampling scheme, biased with a "reachability condition". This condition favors root positions close to the original ones, that avoid collisions, and preserve balance. In any case, the continuity of the motion is always preserved thanks to a local optimization scheme.

4. Preprocessing of the reference motion

Adapting a reference motion to environment deformation requires addressing two subproblems. First, at the kinematic level, to identify the relationship between the character joint positions and the environment, to understand how the deformation of the environment affects the motion. Then at the physical level, to identify the contact phases regarding their length and their contribution to the motion. This information allows to re-position the contacts while preserving the two properties. We compute these two descriptors (relationship descriptors, contact descriptors) automatically from the reference motion as follows.

4.1. Relationship Descriptors

Representation based on relationship descriptors [AAKC13] allows a smooth adaptation of an animation to new contexts, when the geometry (of the environment or of the character) is deformed. We quickly recall it here for completeness. Rather than representing the configuration of the avatar by a configuration vector $\mathbf{q}$, the body positions are described by their relative translations from a static set of points called descriptor points (Fig. 3). The descriptor points are sampled on the surface of the environment. The closer the body is to the surface, the denser the sampling is going to be.

In this representation, the joint position $\mathbf{p_j}$ of joint j is represented by the following

/

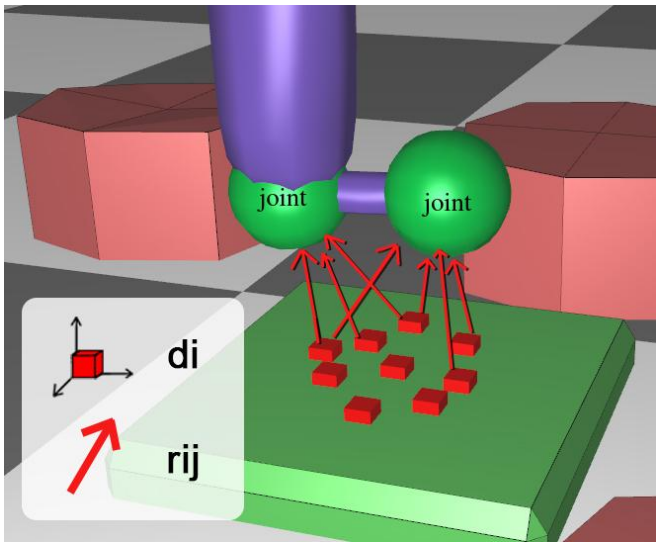


Figure 3: *The joint positions are described as a weighted sum of relative vectors originating from descriptor points embedded in the environment.*

equation:

$$\mathbf{p}_j = \sum_i^N w_{i,j}(\mathbf{d}_i + \mathbf{r}_{i,j}) \tag{1}$$

where $\mathbf{d_i}$ is the position of the i-th descriptor point, $\mathbf{r}_{i,j}$ is the vector between the i-th descriptor point and the j-th joint, and $w_{i,j}$ is the weight of each vector whose value fades out according to the length of $\mathbf{r}_{i,j}$ and its angle with respect to the normal vector at the descriptor point. See the App. 1 for the calculation of the weights.

Rather than storing the initial trajectory as a collection of joint positions $\mathbf{p_j}$, we store, for each body, a reference to the descriptor points $\mathbf{d}_i$ along with the coordinates of $\mathbf{r}_{i,j}$ and the weights $w_{i,j}$, because it encodes the spatial relations between the body and the environment, i.e., the context of the scene.

4.2. Contact Descriptors

We first identify the contact phases of the motion. We then compute a force actuation profile to characterize each contact phase.

/

4.2.1. Definitions

A character R has l limbs $R^k, 1 \leq k \leq l$ attached to the root (i.e. for a humanoid, 2 arms and 2 legs).
$\mathbf{q}^0 \in SE(3)$ is the position and orientation of the root of R;
$\mathbf{q}^k$ is the set of internal joint angle values of a limb of R^k;
$\mathbf{p}^k(\mathbf{q}) = \mathbf{p}^k(\mathbf{q}^0, \mathbf{q}^k)$ is the world position of the k-th end-effector;
$\mathbf{p}^k(\mathbf{q}^k) = \mathbf{p}^k(0_{SE3}, \mathbf{q}^k)$ is the same position in the root frame;
The value of a variable for a frame s is noted with the subscript $_s$.

4.2.2. Contact phases

The detection of a contact might be difficult because of the noise in the input data (i.e. the end-effector is not exactly in contact with the environment or not perfectly static). We thus assume that a contact occurs at frame s if both the distance to the environment and the velocity of the effector are close to zero. This means that the following condition holds:

$$d_s < \varepsilon_0 \quad \text{and} \quad v_s \leq 2\varepsilon_1 \tag{2}$$

where d_s is the minimal Euclidian distance between the end-effector body and the environment, $v_s = ||\mathbf{p}^k(\mathbf{q}^k_{s+1}) - \mathbf{p}^k(\mathbf{q}^k_{s-1})||$ is the instantaneous velocity of the end-effector and ε_0, ε_1 are user defined threshold variables (empirically set to 0.01 and 0.001).

4.2.3. Force actuation profile of a contact phase

Each contact is used by the character to exert a force that contributes to the motion. To re-position a contact, we must allow the character to exert a similar force, otherwise the motion is not plausible. To characterize this contribution, one possibility is to explicitly reconstruct the contact forces with inverse dynamics. However, such methods are very sensitive to noise (they rely on an estimation of the joint acceleration) and are not observable (e.g. when both feet are on the ground, it is not possible to decide what are the respective tangential forces at each foot). We propose an efficient dual approach, that does not compute explicitly the contact forces, but measures the bounds on these forces in any direction.

To compute these bounds, we first follow the steps of Yoshikawa [Yos84]. We then extend its formulation to take into account the quality of the contact surfaces.

/

For brevity, we omit the k exponents in the equations. The relationship between end-effector and joint velocities is given by:

$$\dot{\mathbf{p}}(\mathbf{q}) = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{3}$$

where $\mathbf{J}(\mathbf{q})$ is the jacobian matrix. For a force vector $\mathbf{f}$ and an equivalent joint torque τ, the mechanical work is defined as:

$$\dot{\mathbf{q}}^T \tau = \dot{\mathbf{p}}^T \mathbf{f} \tag{4}$$

We then consider the unit ball in the joint space:

$$||\tau||^2 \leq 1 \tag{5}$$

and map it to the Cartesian space using Equations 3 and 4:

$$\mathbf{f}^T(\mathbf{J}\mathbf{J}^T)\mathbf{f} \leq 1 \tag{6}$$

Equation 6 defines the so-called force ellipsoid [Yos84]. It describes the set of achievable forces $\mathbf{f}$ subject to the constraint 5. The length of an axis measures the ratio between the joint torques and the resulting force applied along its direction. Thus the longer the axis, the bigger the force along the axis can be.

Three examples of force ellipsoids are given in different configurations in Figure 4. The two first configurations present a similar, almost isotropic ellipsoid (ie. close to a sphere); it is slightly deformed so that the force actuation capabilities are more important in the "forward up" direction (the longest axis). In the right frame, the ellipsoid is highly anisotropic, which indicates that the configuration is close to singularity: high force actuation capability in the "forward up direction", but almost no capability in the other directions. If we want to re-position the first configuration, between the second and third configurations, we claim that the best candidate to replace the third contact configuration is the second one.

We now characterize a contact configuration, given the force ellipsoid and the surface of contact. The resulting "force actuation profile" allows us to compare candidate contact replacements.

Since we do not know in which direction the force is applied at a contact, we consider

/

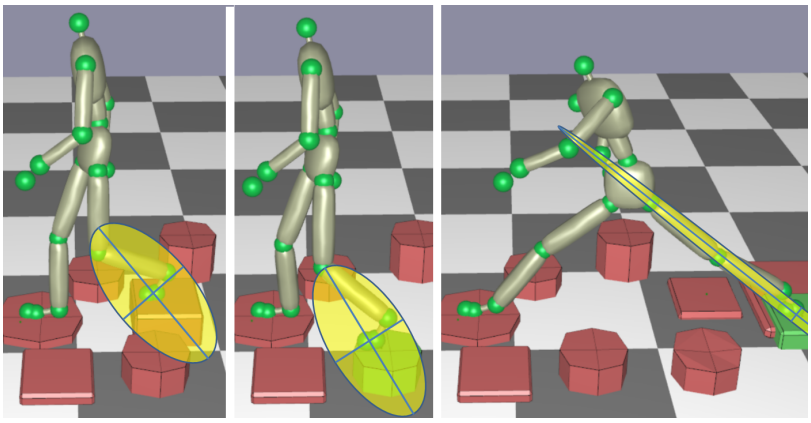


Figure 4: *2D illustration of the force ellipsoid (scale is 0.3). The 2 main axes are drawn with blue lines. The original (left) and re-positioned (middle) contacts present different yet similar force actuation profiles, contrary to the deformed one (right).*

the three main directions of the force ellipsoid $\mathbf{u_1}, \mathbf{u_2}, \mathbf{u_3}$. They are given by the normalized Eigen vectors of $\mathbf{J}\mathbf{J}^\intercal$. Another constraint on each $\mathbf{u_i}$ to enforce balance is given by Coulomb's non-slipping condition :

$$\mathbf{u}_{ti} \leq \nu_0 \mathbf{n}^\intercal \mathbf{u_i} \tag{7}$$

where $\mathbf{u}_{ti}$ is a tangential component of $\mathbf{u_i}$, and ν_0 and $\mathbf{n}$ are the friction coefficient and the normal of the contact surface. Thus, we define the force actuation profile of a contact configuration $\mathbf{q}^k$ as $[\alpha^{\mathbf{u_1}}, \alpha^{\mathbf{u_2}}, \alpha^{\mathbf{u_3}}]$, with:

$$\alpha^{\mathbf{u_i}} = \alpha(\mathbf{q}^k, \mathbf{u_i}, \nu_0, \mathbf{n}) = \frac{1}{\sigma_i}(\nu_0 \mathbf{n}^\intercal \mathbf{u_i}) \tag{8}$$

where $\sigma_i = \sqrt{\mathbf{u_i}^\intercal (\mathbf{J}\mathbf{J}^\intercal) \mathbf{u_i}}$ is the singular value corresponding to $\mathbf{u_i}$. $\alpha^{\mathbf{u_i}}$ is the EFORT measure function, introduced in [TPM14].

The proposed force actuation profile is thus a low-dimensional characterization of the force actuation capabilities of a contact.

5. On-line motion adaptation through contact re-positioning

The contact and relationship descriptors are used in the on-line phase to adapt the motion to deformations of the environment. Whenever the environment is moderately deformed,

/

the relationship descriptors generate a smooth adaptation. In parallel to the automatic adaptation from the relationship descriptors, efficient tests are performed to check whether the deformation is too large, and a contact is invalidated if needed. In that case an equivalent contact is automatically selected using an efficient contact generator. The descriptors are consequently updated with the new contact information, resulting in a smooth motion adaption.

5.1. Adaptation using relationship descriptors

When a surface is deformed, the descriptor points move rigidly with respect to that surface. To maintain the spatial relationship between a body part and a descriptor point, the associated relative vector must also move rigidly and stay invariant after the deformation. However, keeping the relative vectors between all the joints and the descriptor points invariant is in general impossible. Thus the new joint positions are computed such that they are close to the weighted average of the end-points of the relative vectors originating from the descriptor points:

$$\min_{\mathbf{p}_0 \cdots \mathbf{p}_n} \quad \sum_{i=1}^{n} \left(\mathbf{p}_i - \sum_j w_{i,j} (\mathbf{d}_j + \mathbf{r}_{i,j}) \right)$$
$$\text{s.t.} \quad ||\mathbf{p}_i - \mathbf{p}_{i-1}|| = l_i \quad \forall i$$

where l_i is the length of the body segment i. The constraints thus preserves the rigidity of the body parts. The dedicated inverse-kinematics scheme used to solve this minimization problem is proposed in [AAKC13]. For completeness, it is recalled in App. 2.

As a result, a new character posture is obtained that integrates the deformations of the environment.

5.2. Detecting contact invalidation

We propose a novel approach to detect when the deformation of the environment is significant, and thus results in contact invalidation. Outside of collision, contact invalidation can occur for two reasons: joint limits violation and loss of balance. The difficulty of invalidating joint limits comes from the fact that relationship descriptors do not compute the joint values. Rather than performing inverse kinematics to determine these joint values along the

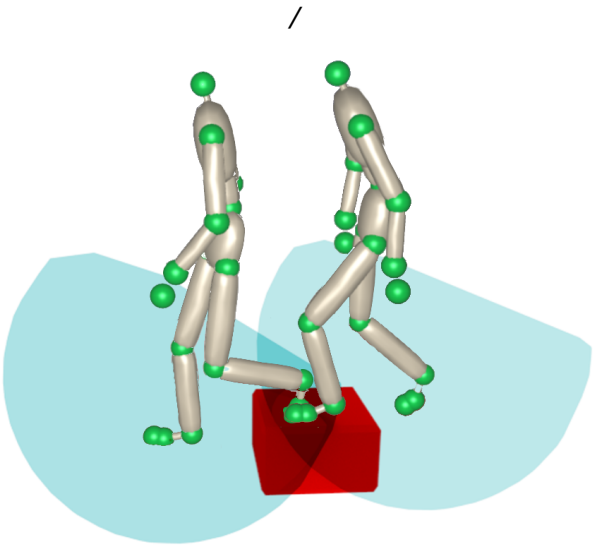


Figure 5: *Reachable workspace of the left foot of the character (blue), at the start and end of a contact phase. The contact can be maintained during the phase because its location belongs to the non null intersection of the workspace at these two different steps.*

motion, we propose a novel, more efficient approach: we verify that the contact location remains in the reachable workspace of the effector during the contact phase.

Joint limit violation. We define the reachable workspace W^k:

$$W^k = \left\{\mathbf{x} \in \mathbb{R}^3 : \exists \mathbf{q}^k, \mathbf{p}^k(\mathbf{q}^k) = \mathbf{x}\right\} \tag{9}$$

We denote by $W^k(\mathbf{q}^0)$ the volume W^k translated and rotated by the rigid displacement $\mathbf{q}^0$. Figure 5 illustrates W^k for the left leg of the character in two different configurations. A contact position $\mathbf{p_c}$ is out of reach from a limb R_k if

$$\mathbf{p_c} \notin W^k(\mathbf{q}^0) \tag{10}$$

We approximate W^k by off-line sampling 100000 configurations and taking the convex hull of the end-effector positions of this set of configurations.

As an over-estimation of W^k, the convex hull introduces false positives (invalid configurations that are not detected). Under this approximation condition 10 is sufficient, but no longer necessary (Figure 6). From a pragmatic point of view, these false positives correspond to configurations close to the body, and are likely to be invalidated by the collision

/

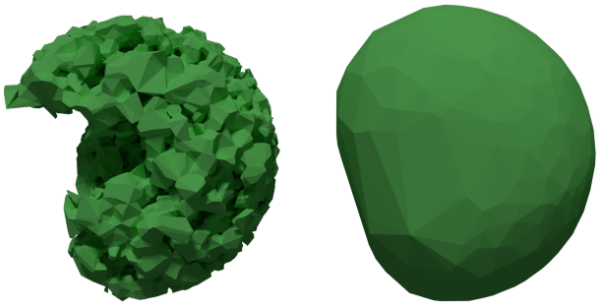


Figure 6: *Two approximations of the range of motion of the right arm. Left: non convex-hull, computed with the powercrust algorithm [ACK01]. Right: convex hull of the reachable workspace.*

tests. We performed 100000 tests, where we compared joint invalidation with our approach versus explicit invalidation with inverse kinematics. As a result the convex hull introduced in average 4% of false positives. We consider this trade-off as an acceptable cost for efficiency.

Detecting loss of balance. Regarding balance, we use an efficient polytope formulation of the problem [QEMR11]. For a given set of contacts, we are able to define a 6 dimensional polytope, that bounds the admissible center of mass positions and accelerations of the character. We can thus trivially test whether balance is lost upon deformation for the whole contact phase. The algorithm to compute the polytope is provided in [QEMR11].

In summary, we can efficiently determine if a contact phase is invalid: First, testing whether the contact point is reachable, and then test the configurations for collision and balance.

5.3. Contact re-positioning

We now address the issue of re-positioning an invalid contact. The new contact must preserve the properties of the original contact, in terms of duration, force actuation profile and balance. Our novel approach consists in an efficient and accurate heuristic to track the force actuation profile of a contact configuration.

We use a sampling based approach to generate candidate configurations. Details of the method can be found in [TPM14], but we recall the principle for completeness. For each

/

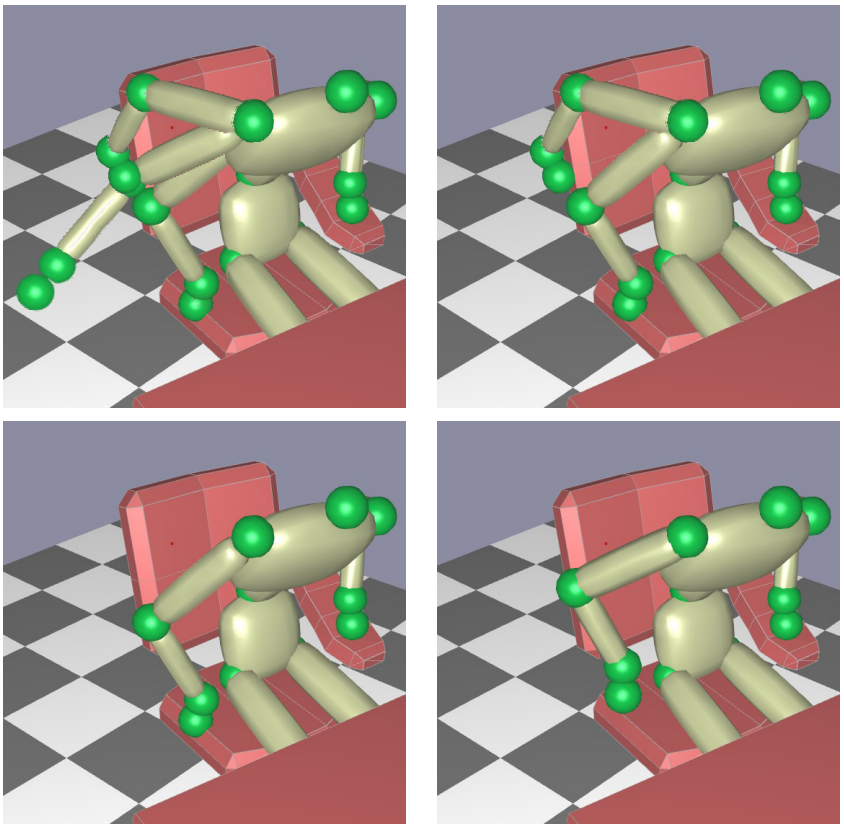


Figure 7: *Contact selection process for the right arm.* ***up-left****: a database of* $N = 3$ *candidate configurations* ***up-right****: a proximity query eliminates the candidate too far from contact.* ***bottom-left****: the best candidate regarding the force actuation profile is chosen.* ***bottom-right****: After optimization, the contact configuration is modified to be blended in the motion.*

limb 10000 valid configurations are stored off-line in an octree, and indexed by the effector position expressed in the root frame (Figure 7–up-left). On-line, the octree is moved to the current root position, and a proximity query is performed to retrieve the configurations close to contact (Figure 7–up-right). Then, the best candidate is chosen according to a user-defined heuristic (Figure 7–bottom-left). In the rest of this section, the candidate list is thus considered to as an input.

Our contribution regarding [TPM14] is the heuristic that we propose here to satisfy the problem constraints, as well as an optimization scheme to preserve motion continuity.

/

Maximizing the force actuation profile similarity

We consider the issue of ordering the candidates to choose a contact with the same force actuation profile as the original contact. We define n^+ and ν_0^+ as the normal and friction coefficient of the contact candidate, and $\mathbf{q}^{k+}$ as the corresponding configuration. Then we define the score force actuation profile heuristic h as:

$$h_1 = \sum_{i=1}^{3} \alpha^{\mathbf{u_i}} \alpha(\mathbf{q}^k, \mathbf{u_i}, \nu_0^+, \mathbf{n}^+) \tag{11}$$

The interpretation of h_1 is the following: The score associated with a given configuration is higher if the principal axes of its force ellipsoid are aligned with the ones of the original contact, and if their length are similar. The score is also higher if the contact allows to respect the non slipping condition.

Verifying contact duration

If the original contact phase existed for frames s to g, ideally the new contact should last for the same frame window. We enforce this by using the reachable workspace of the limb along the motion (Figure 5). The contact candidate is generated at frame $m = (g-s)/2$, and is located at $\mathbf{p_c}$. We define $s^* \leq m \leq g^*$ as the longest frame sequence verifying:

$$\forall i, s^* \leq i \leq g^*, \mathbf{p}_c \in W^k(\mathbf{q}_i^0) \tag{12}$$

The normalized contact duration score h_2 is then simply defined as:

$$h_2 = (g^* - s^*)/(g-s) \tag{13}$$

Global score function

The global score function is a trade-off between the two heuristics:

$$h = wh_1 + (1-w)h_2, w \in [0,1] \tag{14}$$

We empirically set $w = 0.7$.

With h, we obtain an ordered set of candidates. Starting with the best candidate, we verify the balance and collision free conditions sequentially. We return the first candidate that satisfies both.

/

With the presented scheme, the contact generation is biased towards the most promising contact candidates. This results in physically valid configurations in little computation times. The bias introduced does not discard potential candidates, making the method exhaustive relatively to the set of contact candidates.

If the re-positioning is successful, the contact descriptors are updated accordingly, as well as the relationship descriptors, as described in the following section. The failure case is addressed in Section 7.

6. Recalculation of the descriptor points

Upon contact repositioning, the global motion must remain smooth and continuous. To preserve this continuity, the relationship descriptors are updated by solving an optimization problem to blend the original and new motions (Figure 7–bottom-right).

Cost function

Let us assume the position of the joints in the original posture $\mathbf{p}_j$ has been updated to $\mathbf{p}_j{}'$ due to repositioning the contacts. Our aim here is to modify the original attributes of the descriptors (describing $\mathbf{p}_j$) so that they now correspond to the new position $\mathbf{p}_j{}'$. We adjust the origin $\mathbf{d}_i$ and orientation $\mathbf{R}_i$ of the descriptors while using the relative vectors $\mathbf{r}_{i,j}$ in the original scene, such that the spatial relations between the environment and the body in the original scene is preserved. The main term of the cost function is then:

$$E_J = \sum_{i,j} \|\mathbf{p}_j{}' - \sum_{i}^{N} w_{i,j}(\mathbf{d}_i + \mathbf{R}_i \mathbf{r}_{i,j})\|^2 \tag{15}$$

In addition to Eq. (15), we consider the following regularization function that represents the difference in translation and rotation between the descriptor and the surrounding ones:

$$E_S = \sum_{i} \sum_{n_i} (\|\mathbf{d}_i - \mathbf{d}_{n_i}\|^2 + \|\mathbf{R}_i - \mathbf{R}_{n_i}\|_F^2) \tag{16}$$

where n_i is an index for the k-nearest neighbors (here we use $k = 3$), and $\|\cdot\|_F$ is the Frobenius norm. This regularization tends to make the behavior of the collection of descriptors spatially coherent.

/

Finally, a second regularization term is added to penalize scaling/shearing and large updates from the previous iteration:

$$E_R = \sum_i \|\mathbf{R}_i - \mathbf{R}'_i\|_F^2 \tag{17}$$

where $\mathbf{R}'_i$ is the rotation matrix computed in the previous iteration, that is initialized as an identity matrix. This term alone will not manage to guarantee that $\mathbf{R}_i$ remains a rotation matrix after several iterations; but it is doable by a projection scheme in the minimization scheme, explained below.

We compute the optimal translation and rotation of the relationship descriptors by minimizing the following error function:

$$\min_{\substack{\mathbf{d}_1 \ldots \mathbf{d}_n \\ \mathbf{R}_1 \ldots \mathbf{R}_n}} \quad E_J + w_S E_S + w_R E_R, \tag{18}$$

where w_S and w_R are weights described below.

Minimization scheme

Eq. (18) is a non linear minimization problem that we solve by a dedicated iterative approach. At each iteration of the minimization, the problem is reorganized as a linear least-square problem in the form of:

$$E_{\{J,S,R\}} = \|\mathbf{A}_{\{J,S,R\}}\mathbf{b} - \mathbf{c}_{\{J,S,R\}}\|^2$$

where the decision variable $\mathbf{b}$ is a vector that includes all the elements of $\mathbf{d}_i$ and $\mathbf{R}_i$, i.e., $\mathbf{b} = (\mathbf{d}_1, \mathbf{R}_1^1, \mathbf{R}_1^2, \mathbf{R}_1^3, \ldots, \mathbf{d}_n, \mathbf{R}_n^1, \mathbf{R}_n^2, \mathbf{R}_n^3)^\intercal$, $\mathbf{R}_i^m$ is the m-th row of $\mathbf{R}_i$, and $\mathbf{A}_{\{J,S,R\}}, \mathbf{c}_{\{J,S,R\}}$ are the coefficient matrices and constant vectors from each error function. As a result, we can compute the $\mathbf{b}$ that minimizes an integrated error function $E_J + w_S E_S + w_R E_R$ by

$$\mathbf{b}_o = \arg\min_{\mathbf{b}} E_J + w_S E_S + w_R E_R = (\mathbf{A}^\intercal \mathbf{A})^{-1} \mathbf{A}^\intercal \mathbf{c} \tag{19}$$

where $\mathbf{A} = (\mathbf{A}_J^\intercal, w_S \mathbf{A}_S^\intercal, w_R \mathbf{A}_R^\intercal)^\intercal$ and $\mathbf{c} = (\mathbf{c}_J^\intercal, w_S \mathbf{c}_S^\intercal, w_R \mathbf{c}_R^\intercal)^\intercal$.

As mentioned above, the resulting rotation matrix (computed by solving Eq. (19)) includes scaling/shearing. Therefore, we first apply singular value decomposition (SVD) to

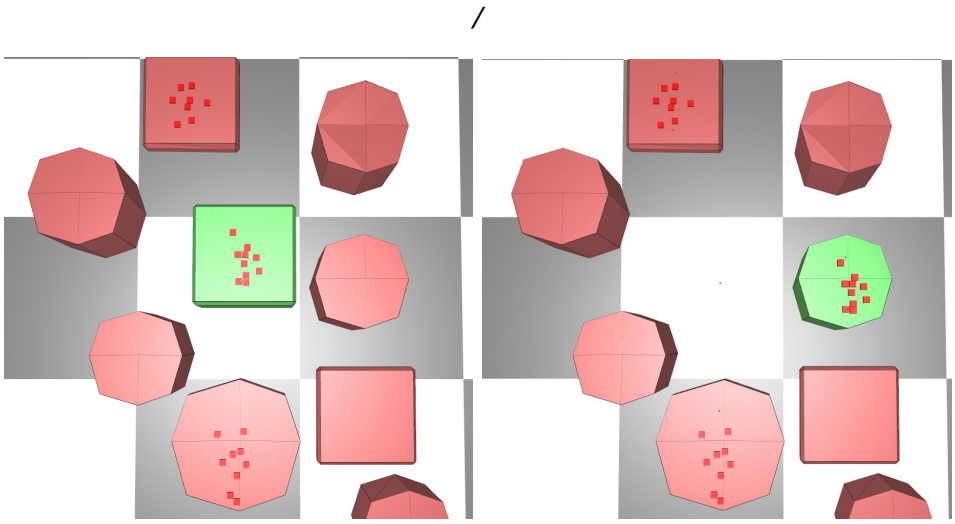


Figure 8: *The distribution of relationship descriptors before and after removing the stepping-stone in the center.*

the computed matrix: $\mathbf{R}_i = \mathbf{U}\Sigma\mathbf{V}$, and then extract the rotation matrix by $\mathbf{R}'_i = \mathbf{U}\mathbf{V}$. This matrix is fed back into Eq. (17) for the next iteration, until E_J is below the threshold.

Regarding the weights of the terms, w_J and w_R are set to 1, while w_S is initially set to 100 and reduced by 1 at each iteration, where the iterations are run 100 times in average. This is to impose more rigidity at the beginning of the optimization and gradually increase the flexibility such that the descriptors can adjust their locations as they converge.

The cost due to rotation difference in Eq.16 and 17 should be ideally measured as rotation difference. This can be done by approximating each rotation by a skew-symmetric matrix, which is a linearization of the rotation matrix [SCOL*04]. We have adopted the current approach due to simplicity of the implementation.

Fig. 8 gives an example where the descriptor points are recomputed after removing a stepping stone in the middle of the environment where the character steps onto. The contact repositioning scheme produces a posture where the character steps on the cylinder at the right. The descriptor points shift over the top of cylinder to correspond to the planned posture. Later displacement of this cylinder will result in a natural adaptation of the planned posture.

/

7. Root trajectory adaptation

If contact re-positioning fails, in a last step we use a sampling-based motion planner to find a solution despite the large environment changes. Such planners are commonly used to generate motions; however the resulting motions tend to lack realism [LKJ99].

Our objective is to preserve the style of the original motion while regenerating a large part of it to face some drastic changes of the environment. We use a dedicated reachability-based planner [TMP*15] to interactively generate new trajectories for the root $\mathbf{q}^0$. In this work the planner is implemented as a RRT for performance. More importantly the search is biased to increase the likelihood of finding contacts compatible with the reference motion, while remaining complete.

7.1. The reachability condition

For a discrete frame, our objective is to find a new root configuration $\mathbf{q}^0$ satisfying three conditions: a) the resulting configuration is collision-free; b) the valid contacts remain valid; c) the invalidated contacts can be replaced. We formally define each condition on the configuration, from which we infer a reduced search space for our planner.

a) A necessary condition to collision avoidance. The valid root configuration is included in C^0_{free}:

$$C^0_{free} = \{\mathbf{q}^0 : W^0(\mathbf{q}^0) \cap O = \emptyset\} \tag{20}$$

where O is the set of obstacles in the environment, and W^0 is the bounding box of the character's head and torso.

b) A necessary condition to contact maintenance. To maintain existing contacts, the root configuration is included in C_{keep}:

$$C_{keep} = \{\mathbf{q}^0 : \forall \mathbf{p}^k_\mathbf{c} \in \mathcal{C}, \quad \mathbf{p}^k_\mathbf{c} \in W^k(\mathbf{q}^0)\} \tag{21}$$

where $\mathcal{C}$ is the set of contacts maintained.

c) A necessary condition for contact re-positioning. To be able to re-position a contact k, the root configuration is included in C^k_{rep}:

$$C^k_{rep} = \{\mathbf{q}^0 : W^k(\mathbf{q}^0) \cap O \neq \emptyset\} \tag{22}$$

/

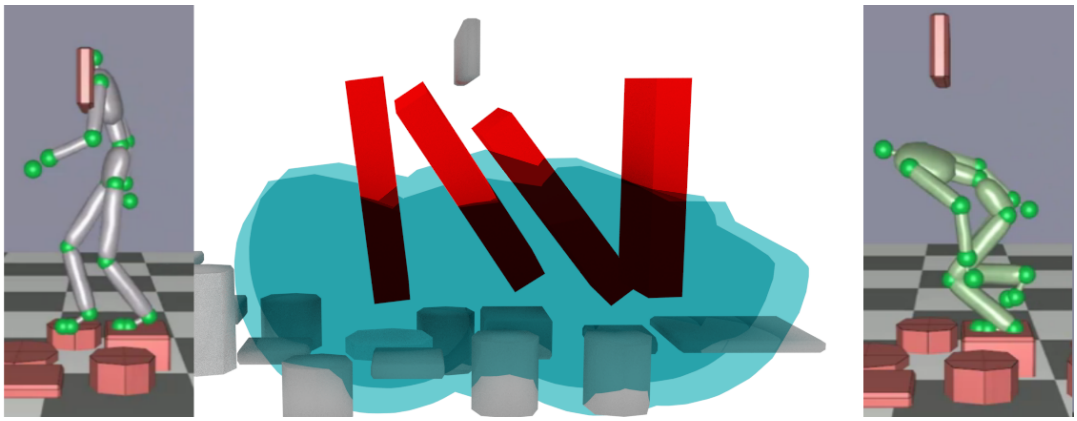


Figure 9: *Reachability planner.* ***Left****: The insertion of the top obstacle leads to invalidation of the deformation.* ***Middle:*** *The reachability RRT is used to compute a new trajectory for the root.* ***Right****: Using the proposed contact generator, a new motion, balanced and collision free, is generated.*

Therefore, the search space for valid root positions verifies the reachability condition, given by $C_{reach} = C^0_{free} \cap C^k_{rep} \cap C_{keep}$. The reachability condition is illustrated in Fig. 9: the red rectangular shape encapsulates the torso and head of the character, while the blue shapes denote the reachability volumes of its legs.

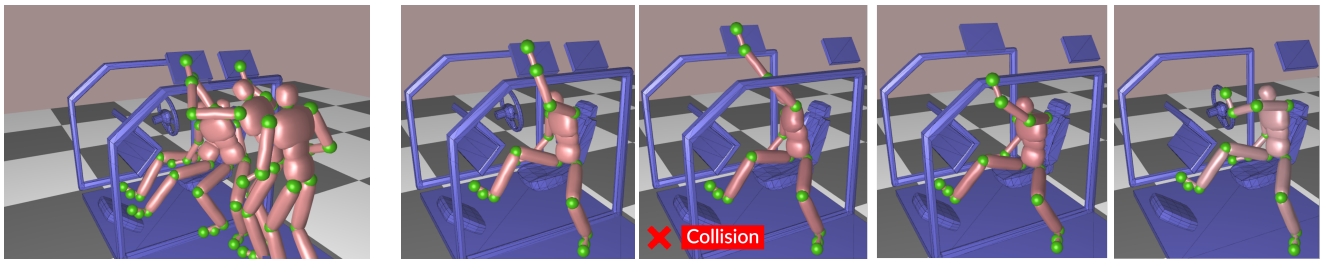
7.2. Biased generation of a root path

To adapt a root trajectory, we thus implement a standard RRT planner, where the configuration generation is modified to only output paths contained in C_{reach}, without loss of generality. Testing the inclusion to C_{reach} is inexpensive, hence we simply generate candidate root configurations in the neighborhood of the original motion, and consider those that satisfy the reachability condition.

7.3. Full body motion generation

When a substitute root path is found by the planner, it is converted into a trajectory for the complete character. First we try to re-position the invalidated contacts as in Section 7. The other end-effector trajectories are computed to track those of the reference motion. To do so we use an inverse-kinematics scheme for each limb separately. This local adaptation

/



(a) (b) (c) (d) (e)

Figure 10: *Car-ingress movement.* ***(a)*** *Original captured motion (4 postures).* ***(b)*** *Original hand position with original car geometry.* ***(c)*** *For large environment changes, the local adaptation fails (here because of a collision).* ***(d)*** *Successful contact re-positioning.* ***(e)*** *The re-positioning enables the character to find a valid motion despite larger changes in the environment (cabriolet-like car).*

is prone to failure (i.e. collision, joint limits, balance) but is cheap and keeps a large part of the style of the reference movement. In case of failure, the problematic contacts are re-positioned using the above method. The adaptation may arbitrarily loop between contact re-positioning and root re-planning but ultimately may fail, which marks the limits of our method.

Our method is thus able to not only re-position contacts upon deformation, but also to modify the root trajectory under dramatic environment updates, while preserving the original motion style.

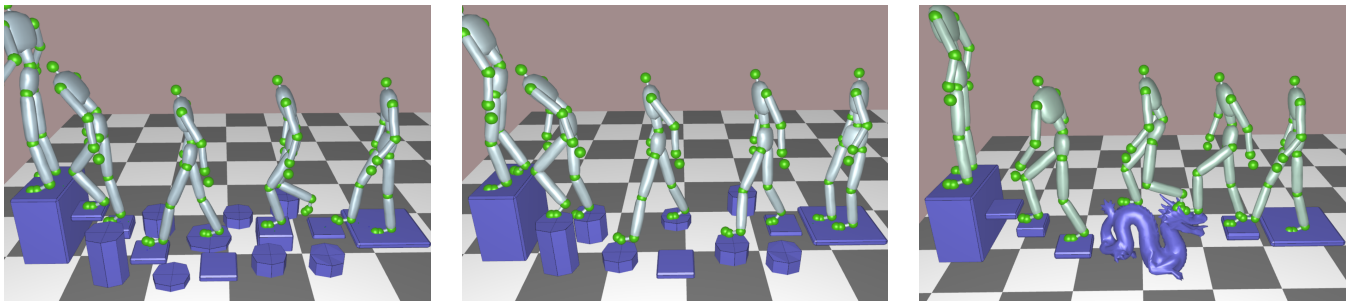
8. Results

We demonstrate our approach in three different environments. Several examples in each context are demonstrated in the companion video, that is an important complement to the figures shown in the paper. All the examples reported in the video have been recorded in real-time: adaptation, re-positioning and re-planning are performed on-line while the user manipulates the environment geometry.

8.1. Car-ingress environment

This environment was used in [AAKC13]. We use it to show how our contribution extends the approach. In Fig. 10, the user changes the car geometry by elevating the roof of the car.

/



(a) (b) (c)

Figure 11: *Step-stone movement.* ***(a)*** *Original captured motion (5 postures) showing reference contact positions.* ***(b)*** *Adaption after several stones have been relocated. The second and fourth contacts have been re-positioned.* ***(c)*** *The Stanford dragon replaces several previous stones. New contacts are generated to step on this geometry.*

The motion is adapted accordingly, the left hand contact position gets higher too, and the arm enters in collision with the door frame. This situation cannot be handled correctly by any previous adaptation method.

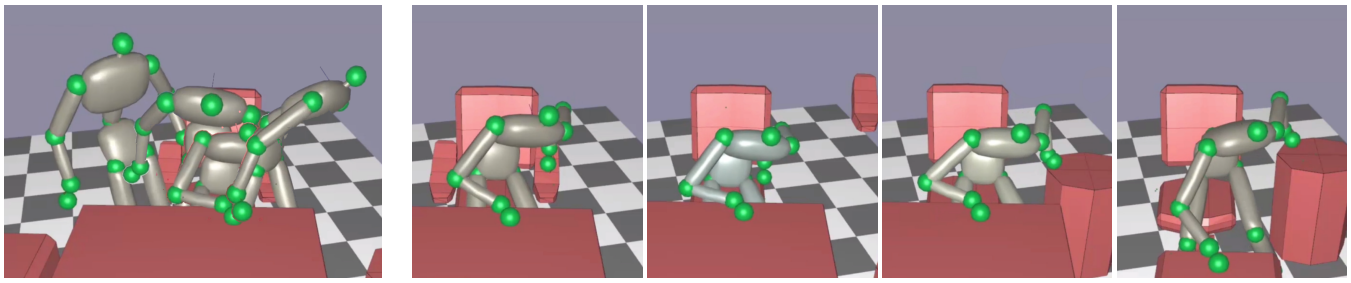
The collision is detected by our system, that automatically re-positions the hand contact on the door frame. The user makes the problem even more difficult by suppressing the door frame and roof geometries. Our solution successfully re-positions the left hand contact, finding that the turning wheel is now the best option.

8.2. Step stones

This environment demonstrates how our method handles locomotion tasks. In the real capture environment, several landmarks were placed on the floor at different heights, leading to a three-step ladder. The human actor stepped on the landmark before climbing the stepladder. The resulting motion capture is illustrated in Fig. 11-left.

The relationship descriptors handle local deformations of the environment (positions and heights of the stones and ladder). Large deformations of the step stones bring the adaption to its limit (Fig. 1-left): a large translation of the green stone results in a unnatural motion. The proposed method detects this situation and re-positions the contact, putting the left foot on the green cylinder. As we show in the video, the transition between the adapted movement and the rest of the sequence is smooth and natural.

/



(a) (b) (c) (d) (e)

Figure 12: *Chair-table environment.* ***(a)*** *Original captured motion (4 postures).* ***(b)*** *Initial posture for standing up. The next images show the re-positioning obtained after several deformations:* ***(c)*** *both armrests are removed, the character makes contact with the chair seat;* ***(d)*** *the character takes advantage of a new obstacle;* ***(e)*** *the table is lowered and the cylinder enlarged.*

The method can handle very large deformations of the environment, as shown in Fig. 11, (middle and right images). Even when all the original step stones have been removed or the shape of the contacts are drastically changed.

8.3. Chair-table environment

The real capture environment is composed of a table and an armchair: the actor sits, interact with the table and gets up to walk away (Fig. 12-left). Complex interactions occur in this cluttered environment, while the character often change contacts to take support on the armrests or the table.

This example demonstrates that our method is robust to large changes in the geometry of the environment. The user sequentially suppresses the armrests and the table. The missing contacts are replaced by other objects of very different dimensions. The character uses the cylinder on its left hand when the left armrest is removed, or even re-position both contacts on the chair when needed. Our method is robust to these large changes and re-position hands correctly. The new adapted motions are plausible and continuous.

9. Discussion and Limitations

Motion plausibility. The quality of the motion we generate is a major aspect of our work. The motion is adapted progressively to the environment deformations, in a manner de-

/

signed to preserve motion plausibility at best. Our work is based on the hypothesis that preserving the geometrical and physical interaction context of a reference motion is essential to generate plausible motion adaptations when contact re-positioning is required.

As a limitation to motion plausibility, we do not consider any semantic information in the contact re-positioning phase. In the car environment, our companion video shows that a large displacement of the turning wheel generates new contacts with a dashboard, but these contacts can have a different meaning.

Robustness. As shown in the Fig. 2, our method can lead to failure cases under extreme user manipulations of the environment geometry. There cannot be a guarantee that our method will find an acceptable motion adaptation, even if a solution may exist. However, we believe that our contact re-positioning strategy extends the possibilities of motion adaptation techniques. Indeed, few methods are able to automatically detect failure cases and find the appropriate contact replacements in the specific context we address.

For larger deformations, we recover new contacts using a sampling based planner (illustrated in the chair-table environment in the video). This insertion of purely-procedural elements can, in some case, lead to less plausible motions. This explains why the reachability RRT is used in the last step of our method.

9.1. Performances

Our solution is interactive, with up to one-second to adapt the motion in the most complex cases. Our demonstration scenarios are representative of typical usage in terms of number of contacts and environment complexity.

The performance of the method is however sensitive to the number of triangles in the scene (when checking for collisions and searching for new contacts). Before considering performance evaluation in future work, it would be relevant first to improve our implementation by designing a parallel version of the contact re-positioning part.

10. Conclusion

We present a method to interactively adapt motion capture animations when major changes occur in the environment geometry. Our method extends previous work, in particular [AAKC13], in two novel manners:

/

We propose objective criteria to determine when a adaption scheme produces unnatural motions (because of collision, joint limit violation, or loss of balance). When this failure happens, we propose a procedural scheme that allows to re-position the contacts involved in the motion. To do so, we introduce the force actuation profile, a heuristic that allows us to replace an invalid contact with an equivalent one regarding its contribution to the motion. Upon drastic changes, we ultimately rely on a biased sampling strategy, the reachability based planner, to adapt the root trajectory.

Building on the relationship descriptors method, we propose an optimization scheme that guarantees despite contact and root re-positioning, the motion remains continuous and smooth.

The ability to re-position contacts extends the amplitude of changes a user can make to the environment geometry. The current work immediately yields two possible applications. The first one is scene editing. An animator can adapt a geometry as he sees fit, without requiring to manually edit new contact positions when necessary. Furthermore, the performances of the method allow him to see the results of his edition immediately, thus allowing for an interactive editing loop. The second one is real time applications. In video games, a limited set of motion capture animations is used in various scenes. Those scenes are however often very similar due to the limitations of the methods of the industry. Contact re-positioning allows to introduce more variations in the scenes and is less restricting to the level designer creativity. For instance, the standing up from a chair motion that we demonstrate can be automatically played with a large variety of chairs, with or without armchairs for instance, thus allowing for a larger diversity. It would be interesting to experimentally evaluate the gain in animation time and creativity this can generate through a user-study.

A challenging issue to address for future work is the adaptation of a motion to sudden changes in the environment. This involves being able to find a continuous solution ahead of time, that anticipates the future state of the system. A promising direction is to combine our method with Model Predictive Control to achieve this, as proposed by [HRL15].

As a last remark, another direction for future research would be to isolate each contact interaction and consider the couple surface /limb independently from the rest of the body. Following the idea of motion graphs, we could then analyze long motion sequences to

/

design "interaction graphs". This approach would allow us to decompose the motions and generate contact phases independently for each limb, thus addressing the combinatorial issue of contact creation in complex environments.

References

[AAKC13] AL-ASQHAR R. A., KOMURA T., CHOI M. G.: Relationship descriptors for interactive motion adaptation. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2013), SCA '13, ACM, pp. 45–53. URL: http://doi.acm.org/10.1145/2485895.2485905, doi:10.1145/2485895.2485905. 2, 5, 7, 12, 22, 25

[ACK01] AMENTA N., CHOI S., KOLLURI R. K.: The power crust. In *Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications* (New York, NY, USA, 2001), SMA '01, ACM, pp. 249–266. URL: http://doi.acm.org/10.1145/376957.376986, doi:10.1145/376957.376986. 14

[AF02] ARIKAN O., FORSYTH D. A.: Interactive motion generation from examples. *ACM Trans. on Graphics 21*, 3 (2002), 483–490. 4

[BB04] BAERLOCHER P., BOULIC R.: An inverse kinematics architecture enforcing an arbitrary number of strict priority levels. *The visual computer 20*, 6 (2004), 402–417. 4

[CLS03] CHOI M. G., LEE J., SHIN S. Y.: Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Trans. on Graphics 22*, 2 (2003), 182–203. doi:10.1145/636886.636889. 4

[HET*14] HÄMÄLÄINEN P., ERIKSSON S., TANSKANEN E., KYRKI V., LEHTINEN J.: Online motion synthesis using sequential monte carlo. URL: http://doi.acm.org/10.1145/2601097.2601218, doi:10.1145/2601097.2601218. 4

[HKT10] HO E. S. L., KOMURA T., TAI C.-L.: Spatial relationship preserving character motion adaptation. *ACM Trans. Graph. 29*, 4 (July 2010), 33:1–33:8. URL: http://doi.acm.org/10.1145/1778765.1778770, doi:10.1145/1778765.1778770. 5

[HRL15] HÄMÄLÄINEN P., RAJAMÄKI J., LIU C. K.: Online control of simulated humanoids using particle belief propagation. In *Proc. SIGGRAPH '15* (New York, NY, USA, 2015), ACM. 4, 26

[Jak01] JAKOBSEN T.: Advanced character physics. *In Game Developers Conference Proceedings* (2001), 383–401. 28

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. In *ACM Trans. on Graphics* (New York, NY, USA, 2002), vol. 21, ACM. 4

[KHKL09] KIM M., HYUN K., KIM J., LEE J.: Synchronized multi-character motion editing. *ACM transactions on graphics (TOG) 28*, 3 (2009), 79. 4

[LCL06] LEE K. H., CHOI M. G., LEE J.: Motion patches: building blocks for virtual environments annotated with motion data. *ACM Trans. Graph. 25*, 3 (2006), 898–906. URL: http://doi.acm.org/10.1145/1141911.1141972, doi:10.1145/1141911.1141972. 5

[LCR*02] LEE J., CHAI J., REITSMA P. S. A., HODGINS J. K., POLLARD N. S.: Interactive control of avatars animated with human motion data. *ACM Trans. Graph. 21*, 3 (July 2002), 491–500. URL: http://doi.acm.org/10.1145/566654.566607, doi:10.1145/566654.566607. 4

[LKJ99] LAVALLE S. M., KUFFNER J. J., JR.: Randomized kinodynamic planning, 1999. 20

[LLKP11] LEVINE S., LEE Y., KOLTUN V., POPOVIĆ Z.: Space-time planning with parameterized locomotion controllers. *ACM Transactions on Graphics (TOG) 30*, 3 (2011), 23. 4

[MC12] MIN J., CHAI J.: Motion graphs++: a compact generative model for semantic motion analysis and synthesis. *ACM Transactions on Graphics (TOG) 31*, 6 (2012), 153. 4

[MTP12] MORDATCH I., TODOROV E., POPOVIĆ Z.: Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. on Graphics 31*, 4 (2012). 4

[PBvdP15] PENG X. B., BERSETH G., VAN DE PANNE M.: Dynamic terrain traversal skills using reinforcement learning. *ACM Transactions on Graphics (to appear)* (2015). 4

[PLS03] PETTRÉ J., LAUMOND J.-P., SIMÉON T.: A 2-stages locomotion planner for digital actors. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2003), SCA '03, Eurographics Association. 4

[QEMR11] QIU Z., ESCANDE A., MICAELLI A., ROBERT T.: Human motions analysis and simulation based on a general criterion of stability. In *Int. Symposium on Digital Human Modeling* (2011). 14

[RBC98] ROSE C., BODENHEIMER B., COHEN M. F.: Verbs and adverbs: Multidimensional motion interpolation using radial basis functions. *IEEE Computer Graphics and Applications 18* (1998), 32–40. 4

[SCOL*04] SORKINE O., COHEN-OR D., LIPMAN Y., ALEXA M., RÖSSL C., SEIDEL H.-P.: Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (2004), ACM, pp. 175–184. 5, 19

[SH07] SAFONOVA A., HODGINS J. K.: Construction and optimal search of interpolated motion graphs. *ACM Trans. Graph. 26*, 3 (July 2007). URL: http://doi.acm.org/10.1145/1276377.1276510. 4

[SLSG01] SHIN H. J., LEE J., SHIN S. Y., GLEICHER M.: Computer puppetry: An importance-based approach. *ACM Transactions on Graphics (TOG) 20*, 2 (2001), 67–94. 5

/

[Tho91] THOMAS J. J. (Ed.):. *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1991* (1991), ACM. 3

[TMP*15] TONNEAU S., MANSARD N., PARK C., MANOCHA D., MULTON F., PETTRÉ J.: A reachability-based planner for sequences of acyclic contacts in cluttered environments. In *Int. Symp. Robotics Research (ISRR), (Sestri Levante, Italy), September 2015* (2015). 20

[TPM14] TONNEAU S., PETTRÉ J., MULTON F.: Using task efficient contact configurations to animate creatures in arbitrary environments. *Computers & Graphics 45*, 0 (2014). 5, 11, 14, 15

[vdPL95] VAN DE PANNE M., LAMOURET A.: Guided optimization for balanced locomotion, september 1995. URL: http://maverick.inria.fr/Publications/1995/VL95. 3

[WP95] WITKIN A., POPOVIC Z.: Motion warping. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1995), SIGGRAPH '95, ACM, pp. 105–108. doi:10.1145/218380.218422. 4

[YLvdP07] YIN K., LOKEN K., VAN DE PANNE M.: Simbicon: Simple biped locomotion control. *ACM Trans. on Graphics 26*, 3 (2007), Article 105. 3

[Yos84] YOSHIKAWA T.: Analysis and Control of Robot. Manipulators with Redundancy, 1984. 9, 10

A1: Relative Vector Weights

Here we describe how we compute the weights in Eq. (1) that determine how each relative vector contributes to the joint position. For each descriptor points we first compute:

$$w'_{i,j} = \frac{\mathbf{n_i} \cdot \mathbf{r}_{i,j}}{\|\mathbf{r}_{i,j}\|}. \tag{23}$$

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

$$w''_{i,j} = w'_{i,j} f(\|\mathbf{r}_{i,j}\|), \tag{24}$$

$$f(d) = \begin{cases} 0 \ (r_2^j \leq d) \\ 1 - \frac{d - r_1^j}{r_2^j - r_1^j} \ (r_1^j < d < r_2^j) \\ 1 \ (d \leq r_1^j), \end{cases} \tag{25}$$

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:

$$w_{i,j} = \frac{w''_{i,j}}{\sum_i w''_{i,j}}. \tag{26}$$

A2: Fitting the Kinematic Skeleton to Target Joint Positions

Here we describe about the iterative inverse kinematics scheme based on [Jak01], where we fit the kinematic body structure to the target joint positions obtained from Eq. (1). This scheme can be roughly broken down into the following four steps: (1) the computation of

/

the affinity, which determines how strongly the joints must be pulled towards the target positions, (2) the force accumulation and integration step, where the effect of external forces such as pushing, pulling etc. are taken into account, and (3) the bone-length constraint step, where the joint positions are updated such that the distance between the adjacent joints are kept constant.

Affinity Calculation:

The affinity values are computed by summing the weights of the associated descriptors on the surface computed in Eq. (26) and normalizing them:

$$s_j = \frac{\sum_i^N w_{i,j}}{\sum_j^{N_j} \sum_i^N w_{i,j}} \tag{27}$$

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

Force Accumulation and Integration:

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:

$$\mathbf{f}_j = k(\mathbf{p_j^{tar}} - \mathbf{p_j^{cur}}) + \mathbf{f^{ext}}. \tag{28}$$

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. (1), $\mathbf{p_j^{cur}}$ 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.

The joint target positions are then computed by Verlet integration:

$$\mathbf{p_j^{new}} = \mathbf{p_j^{cur}} + d(\mathbf{p_j^{cur}} - \mathbf{p_j^{prev}}) + \frac{1}{2}\mathbf{f_j}\frac{1}{N_s^2} \tag{29}$$

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 constraint 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:

$$\Delta \mathbf{p_j} = \frac{s_k}{s_k + s_j} \frac{\mathbf{p_j} - \mathbf{p_k}}{\|\mathbf{p_j} - \mathbf{p_k}\|} (l^0 - \|\mathbf{p_j} - \mathbf{p_k}\|) \tag{30}$$

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.

A3: Other Techniques

Rapidly Exploring Random Trees (RRT):

A rapidly exploring random tree (RRT) is an algorithm designed to search high-dimensional spaces by randomly building a space-filling tree. Samples are randomly drawn from the search space and are biased to grow towards unsearched areas of the space. The tree expands incrementally from samples already drawn from the search space.

Detecting Loss of Balance

Every contact between a character limb or end effector and a surface within the system consists of a combination of a force and a torque, known as a wrench when combined as a 6D feature. Given a set of such contacts which suspend the character we are able to define a 6 dimensional polytope often known as the external wrench space based on those contacts or wrenches. In order for the system to be stable the forces or wrenches from the contact points must suspend the center of mass of the character affected by gravity. The polytope generated by those contacts bounds the admissible center of mass positions and accelerations of the character. We can thus trivially test whether balance is lost upon deformation for the whole contact phase by evaluating the distance of the dynamic perturbation to the boundary of the polytope and ensuring it's within a particular range. The algorithm to compute the polytope is provided in [QEMR11].

Chapter 6

Conclusion & Summary

In summary, there is a high demand for methods that are able to handle animations involving close interactions and to adapt them particularly in real time environments and previous methods are not able to do this effectively, and if they are then not in real time.

This thesis raises the question of whether there's an effective representation based on spatial relations that can encode interactions but that in particular can be used to solve motion retargeting problems in a real-time environment. Our experiments have so far given a positive answer on this front, and proven to be able to effectively adapt motions to varying scene configurations and deformations of objects and character morphologies.

The system fails in cases of extreme scene deformations but even on this front, we have shown that combined with procedural animation techniques such as repositioning contacts we can detect violations and change key poses and contacts yet still be able to preserve the continuity and style of the interaction with the relationship descriptors space we propose. Furthermore we can go further and map entire animations to novel geometries rather than mapping single poses only due to the fact that we only need to solve the mapping problem in the space of the relationship descriptor space in the spatial sense removing the need to solve over a temporal window. This way we can recycle existing source animations to generate all kinds of adapted variations that conform to new or deformed meshes without the need to manually animate every combination of motions for every object and character morphology configuration, particularly in game packages which contain many variations.

However, it must be noted that the method still suffers key limitations. For one, it assumes that proximity, distance and spatial relations are key to defining the context of an interaction when often this is not the case semantically and can give wrong adaptations. For example, a character may interact with a desk. There are spatial relations between the character and all elements of the chair and desk. Some of those spatial relations are semantically meaningful and should be preserved, such as the character's fingers relative to the keyboard, or the character's bottom with respect to the chair they're seated upon. Other spatial relations are meaningless on the other hand. There could be a random prop near

the elbow of the character on the desk. It may be adjacent, but that doesn't mean it should influence the interaction. If it is moved to a different location then the character's elbow may be outstretched towards it a meaningless way to preserve the distance to it. While avoiding the prop may be important, following it when it moves away may not be. Due to the generalized nature of our proposed method, all geometry and spatial relations are treated equally. In order to improve the results on this front, there must be a way to add labels upon the scene elements to determine the semantics and spatial-relation type and whether proximity has meaning for certain parts of an object or environment.

For two the system fails in cases of extreme deformations to the scene. Even though the introduction of contact descriptors helps alleviate this problem and increases the range of amplitudes of deformations that can be handled, there will always be cases where it's not possible to achieve a natural posture when adapting. For the car example, when removing the steering wheel we attempt to compensate that by changing the contact of the hand to another random object in the scene to preserve balance, but this new contact may hold no meaning in the scene anymore at that point.

Furthermore the method only produces quasi-physical effects and is not truly momentum preserving. Adapted motions can hence violate physical properties and constraints which can be an issue. An example of this is the ballistic swing demo which we adapt based on spatial relations without considering physics and momentum which can cause unnatural looking behaviour.

Ultimately, our proposed method is ideal to solve a particular problem domain in motion retargeting and is not suitable for everything. Finally the adapted results don't absolutely guarantee collision-free results even though they implicitly biased away from collisions.

When it comes to mapping animations from template to novel objects on the other hand, we are still quite limited to using co-segmented objects to initialize a coarse correspondence for one, and due to the fact that we obtain spatial relation data with respect to only a single template object and from only a single source animation rather than training it over a larger data-set also limits the range of shape variations of novel objects that we can effectively adapt the animation to.

Our results, observations and limitations do motivate a number of directions for future work. For one, it would be useful to explore adding additional semantic information to our spatial relationships, perhaps based on labels added to objects in scenes. This way we can guide the adaptation only to

spatial relations that matter, while other spatial relations can be ignored or behave differently.

For two, it would be interesting to use the descriptors to synthesize novel animations such as achieved in [Mordatch et al. 2012]. We are also interested in learning movements in our representation which can be an interesting direction because interactions learned by observation can be applied in a broader range due to the adaptability of the scheme. It would be worth labelling and learning interactions over a large set of interaction data between template animations and template objects based on the spatial relationship descriptors. That way when we introduce a novel object of a certain class, either provided or determined by a classifier, we would be able to adapt certain animations to this novel object more effectively since we'd have a larger dataset which was trained to determine the adaptation in a probabilistic sense more accurately and effectively.

Furthermore, despite the fact that 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 which is another field for future work. For example, an interesting possibility would be to produce a PD servo based on our relation descriptors. In such a case, the adapted motion would represent target locations for the joints in the system and we can attempt to solve for torques to exert upon the joints in order to achieve the adapted postures which act as references, subject to additional physics constraints, to create a physically plausible version of the animation.