UC Merced

UC Merced Electronic Theses and Dissertations

Title

Data-Based Motion Planning for Full-Body Virtual Human Interaction with the Environment

Permalink

https://escholarship.org/uc/item/2314d59w

Author

Juarez Perez, Alain

Publication Date

2018

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA, MERCED

Data-Based Motion Planning for Full-Body Virtual Human Interaction with the Environment

A *Dissertation* submitted in partial satisfaction of the requirements
for the Ph.D. degree

in

Electrical Engineering and Computer Science

by

Alain Juarez Perez

Committee in charge:

Professor Marcelo Kallmann, Chair
Professor Stefano Carpin
Professor David Noelle

2018

Copyright ©

Alain Juarez Perez, 2018

All rights reserved

The dissertation of Alain Juarez Perez is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Approved: _____
Professor David Noelle

Approved: _____
Professor Stefano Carpin

Approved: _____
Professor Marcelo Kallmann, Committee Chair

University of California, Merced
2018

To my family and Sandy

Contents

List of Figures vii

List of Tables x
Acknowledgments xi
Vita xii
Abstract xiii

Chapter 1. Introduction 1
1.1. Definitions 1
1.2. Character Animation 2
1.3. Data acquisition 2
1.4. Overview 3

Chapter 2. Literature Review 6
2.1. Character Animation 6
2.2. Motion Graphs 7
2.3. Physics-based animation and control 8
2.4. Data-based approaches 9
2.5. Motion Planning 10
2.6. Environment Interaction 11
2.7. Conclusion 12

Chapter 3. Locomotion 13
3.1. Related Work 13
3.2. Data-Based Mobility Controllers 14
3.3. Coverage-Quality Maps 19
3.4. Results and Discussion 19
3.5. Conclusion 23

Chapter 4. Locomotion Planning 25
4.1. Introduction 25
4.2. Related Work 26
4.3. Outline 27
4.4. Locomotion Behaviors 27
4.5. Behavioral Path Planner 28
4.6. Behavioral Layered Path Planner 38
4.7. Discussion 47
4.8. Conclusion 48

Chapter 5. Coordination 50
5.1. Introduction 50
5.2. Related Work 50
5.3. Overview 52
5.4. Encoding Coordination Features 54
5.5. Real-Time Motion Synthesis with the Coordination Model 58
5.6. Results and Discussion 62
5.7. Conclusion 67

Chapter 6. Conclusions 69
6.1. Limitations 69
6.2. Future Work 70

References 72

List of Figures

2.1 Running animation by Hodgins et al. [1995]. ©1995 ACM 6
2.2 Motion Graph generated animation by Kovar et al. [2002]. ©2002 ACM 7
2.3 Different physical models used by Zimmermann et al. [2015]. ©2015 ACM 9
2.4 Peng and van de Panne [2017]. ©2017 ACM 10
2.5 Agrawal and van de Panne [2016]. ©2016 ACM 11

3.1 Deformations over a cycle. The left-most diagram corresponds to the original clip cycle. The next ones represent the outcome of each deformation operation 16
3.2 Blending operations according to cycle phases. 18
3.3 Coverage-quality map for one forward step cycle motion. The heatmap coloring represents the overall computed validity loss over the original data. 20
3.4 Final body orientation (after forward step) coverage-quality map. 21
3.5 Coverage-quality map of one right lateral step. 22
3.6 Volumetric coverage-quality map for a frontal step. The horizontal layers are planar maps equivalent to the one in Figure 3, with the vertical axis representing changes to the final body orientation deformation parameter. 23
3.7 Example of real-time control of locomotion forming a circular trajectory. 24

4.1 Parameterization coverage maps for the frontal regular walking (top) and lateral walking (bottom) locomotion behaviors. The maps are on polar coordinates. The angle coordinate represents motion deformation controlling the direction of the motion. The radial coordinate represents the orientation deformation that is applied to specify the final orientation of the character's body, at the end of one step cycle. The black boundary delimits the radius representing no deformation on the final character orientation. 29
4.2 The blue positions represent the projected joint position of the limbs that collided with obstacles. The green positions are the projected character root positions on the floor at the moment of each collision. The red positions are the point-obstacles that were inserted in the LCT navigation mesh as additional constraints to be considered in subsequent path queries. The final path (marked by the dark red corridor) takes into account the new point-obstacles that were inserted. 33
4.3 Example solution path found by our method. The blue path sections can be executed with the preferred behavior $\mathcal{B}_3$. The green section with $\mathcal{B}_2$ and the red sections with $\mathcal{B}_1$. The obstacles are colored cyan. 33

4.4 System overview 34

4.5 Left: The obstacle was avoided by adding extra point obstacles to steer away the path. Right: The corridor is narrow so an arm avoidance behavior was needed. 34

4.6 Behavior transition in a narrow corridor. From top to bottom: arm avoidance behavior employed in the first section. In the middle section regular walking could be used because no collisions were detected even though the path has low clearance. In the final section lateral side-stepping is required. 35

4.7 Top: dense environment. Bottom: normal environment. Computed solution corridors are shown in both environments for a same goal point. 36

4.8 Performance evaluation. The graph presents the standard deviation and average times required to compute paths of different lengths on dense and normal environments. 37

4.9 Left: the frontal walking behavior requires the torso clearance larger than the bottom clearance. Center: with arms constrained frontal walking requires only slightly more torso clearance than legs clearance. Right: lateral walking requires the least clearance and the torso and legs clearances are the same because arm movement occurs only along the motion direction. 39

4.10 Two layers are used in this method. The bottom layer (bottom images) represents all obstacles while the top layer (top images) only represents the tall obstacles which are mostly important to detect torso collisions. 40

4.11 Example solution path found by our method. The blue path sections can be executed with $\mathcal{B}_3$, the cyan sections with $\mathcal{B}_2$, and the red sections with $\mathcal{B}_1$. 41

4.12 Apartment scenario. 44

4.13 Narrow corridor behavior selection. From top to bottom: arm avoidance is employed in the first section, regular walking is correctly employed in the middle section given the high clearance at the torso layer, and lateral side-stepping is required in the last, narrowest, section. 45

4.14 Dense (top) and Normal (bottom) environments. 46

4.15 Average times, deviation and outliers when computing paths of different lengths on a dense environment. 47

5.1 This sequence of opening and passing through a door synthesizes the upper-body motion in coordination with a generic walking controller. This result demonstrates complex anticipatory spine and arm movements which are key for successfully performing the action without collisions. 51

5.2 Overall method overview. Green boxes correspond to locomotion operations and blue boxes correspond to the main modules of the coordination model. 53

5.3 *Left:* Set of minimum vectors to the environment considering all body parts independently. *Right:* Set of minimum vectors with respect to only the body sections of interest. 55

5.4 GPLVM visualization of different CFV encodings. **A:** joint angles, **B:** full feature vector; **C:** proposed feature vector. The motion used in this validation example is a door opening motion (bottom image). The color bar on top of the motion is used to label each of the poses by vertically projecting on the bar the skeleton's root position at each pose. These colors identify the poses corresponding to each curve point in images A, B and C. 57

5.5 The image represents for every simulated frame the distance value v_j of each element in the set of k-nearest neighbors. Every non-black pixel in a column of the image corresponds to one of the selected k-nearest neighbors for the corresponding simulated real-time frame. The image shows that the k-nearest neighbors are most often composed of adjacent frames in the training data, indicating high correlation between the training data and the action being synthesized. 61

5.6 In left-right order: the example motions had approach angles of: 0°, 90° and 180° degrees. The postures in this image were synthesized with our coordination method given that differences in the character size and door dimensions lead to collisions with the considered environment when displaying the original data. 63

5.7 By allowing obstacles to be placed anywhere in the scene the approach direction to open a door can be arbitrary (left) instead of always front-facing (right). Our examples motions cover different approach directions and the most suitable coordination model is automatically selected by the system. 64

5.8 Drawing sequence for a generic shape. This example was synthesized with a coordination model built from straight line drawing examples in attentive style. 65

5.9 Left: Speed control turned off. Center: All modules turned on. Right: Spine deformations turned off. 66

List of Tables

4.1 Average times in milliseconds to compute paths with our method based on layered queries versus using full 3D collision checking. 44

4.2 Comparison table with behavioral planning methods. 48

5.1 Average computation times, in miliseconds, taken by the main modules of our method. 66

Acknowledgments

I want to begin by expressing my gratitude to Professor Marcelo Kallmann. He helped me steer and mature my ideas into what this text presents. I have greatly improved under his guidance and I know that my PhD years will be of utmost importance to my Professional future. I would like to extend my gratitude to my committee members, Professor Stefano Carpin and Professor David Noelle for their guidance and directions throughout my qualifying examination, meetings and my defense.

I want to thank the Mexican National Council of Science and Technology (CONACYT) and the The University of California Institute for Mexico and the United States (UC MEXUS) for their support to my studies. I would like to thank the University of California, Merced for providing me with the opportunity to work as a Teaching Assistant. I would also like to express my gratitude to the University of Southern California Institute for Creative Technologies (ICT), for offering me an internship in Summer 2014.

I was lucky to have met and lived with so many good friends in my time in Merced. I will keep great memories from all of them.

Last but not least I want to thank my parents, my sister and my girlfriend. Even though we just met a few times a year over this past decade I know you were always by my side supporting me.

Vita

Education.

Bachelor of Science(June 2012) in Computer Science, Universidad de Guanajuato, Guanajuato, Guanajuato, Mexico.

Publications.

Juarez-Perez, A. and Kallmann, M. (2018). Fast Behavioral Locomotion with Layered Navigation Meshes. *In Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '18)*. ACM.

Juarez-Perez, A. and Kallmann, M. (2017). Coordinating full-body interactions with the environment. *In Proceedings of the ACM SIGGRAPH/ Eurographics Symposium on Computer Animation (SCA '17)*. Poster Paper, ACM.

Juarez-Perez, A. and Kallmann, M. (2017). Full-body behavioral path planning in cluttered environments. *In Proceedings of the 9th International Conference on Motion in Games (MIG '16)*. ACM.

Juarez-Perez, A. and Kallmann, M. (2016). Modeling data-based mobility controllers with known coverage and quality properties. *The 4th International Digital Human Modeling Symposium (DHM '16).*

Juarez-Perez, A., Feng, A., Kallmann, M., and Shapiro, A. (2014). Deformation, parameterization and analysis of a single locomotion cycle. *In Proceedings of the Seventh International Conference on Motion in Games (MIG '14)*. Poster Paper, ACM.

In preparation.

Juarez-Perez, A. and Kallmann, M. (2018). A Coordination Model for Full-Body Interactions while Walking.

Alain Juarez-Perez
Electrical Engineering and Computer Science
University of California, Merced, 2018

Abstract

Autonomous virtual characters are important in a growing number of applications ranging from simulation-based training to computer games. In this dissertation I propose a new data-based mobile manipulation framework for achieving real-time autonomous characters able to perform full-body interactions with the environment. The overall approach relies on few example motions in order to guide the generation of complex movements that replicate human-like characteristics.

The proposed framework is based on three major components. The first one consists of a fast locomotion module that generates controllable models from single motion examples, and is capable of independent control of direction, orientation and velocity, with known coverage and quality characteristics. The approach is computationally efficient and achieves high controllability of stepping behaviors, thus addressing key properties for supporting a variety of whole-body manipulation tasks.

The second component relies on the proposed locomotion controller applied to multiple locomotion behaviors in order to plan multi-behavior paths around obstacles. A new locomotion planning approach is proposed where the behavioral capabilities of the character are considered during the path planning stage, in order to address trade-offs related to path length and preferred navigation behavior when selecting narrow passages to take. The approach relies on new types of operations with planar navigation meshes, reaching fast execution times suitable for real-time applications.

The last component focuses on the coordination between locomotion and upper-body manipulation. The proposed approach is based on learning spatial coordination features from example motions and on associating body-environment proximity information to the body configurations of each example motion. Body configurations then become the input to a regression system which in turn is able to generate new interactions for different situations in similar environments. The regression model is capable of encoding and replicating key spatial strategies with respect to body coordination and management of environment constraints. Obtained results successfully synthesize complex full-body actions such as opening doors and drawing in a wide whiteboard.

The models proposed in this dissertation achieve new interactive controllers able to synthesize coordinated full-body motions for a variety of complex interactions requiring body mobility and manipulation.

CHAPTER 1

Introduction

Computer Animation is the branch of Computer Graphics that explores motion generation. The area analyzes the algorithmic synthesis, the capture of motion and its reproduction. Historically, Animation has been mostly produced by talented artists, however the complexity of the animated characters and their environments has reached a point in which the generation of a minute of animation can become a tedious and arduous work. One of the objectives of the Computer Animation field is to minimize or eliminate the need of extensive artist hours to create mobile characters. Known approaches explore a multitude of possible motion synthesis techniques with varied objectives such as computation speed, realism and ease of control.

1.1. Definitions

When animating virtual objects, a simple classification of them would focus on the amount of its moving parts. Rigid objects contain none of them, however it is possible to attach them with virtual joint generating articulated bodies. If enough rigid objects and virtual links are created, it is possible to represent an articulated character, which is the main focus of this dissertation.

A standard representation of articulated characters consist of a data structure that mimics a skeleton. Each of the moving parts can be represented by a joint and the connected rigid bodies can be simplified into lower dimensional representations. In this description, the joint information controls the final configuration of the character. Each joint can contain information about the position and orientation among the two rigid bodies it connects, usually the fixed 'bone' is called the parent of the one connected to the joint. This generates a tree structure in which the nodes are the joints and the edges are the bones.

In this dissertation I focus on representations that resemble a human character. I begin by defining the root joint as the root of the skeleton tree representation, this joint will be the only that contains the position and orientation of our character, all the other joints will only have orientation information. For simplicity, the rotation joints are represented by up to 3 independent values, each of them contains information about the rotation over one of the main axis, this representation is often called Euler angles. There are different representations for 3D rotations, like quaternions, axis-angle, swing-twist, rotation matrices, etcetera. Even though different representations are used internally throughout this work, this dissertation will refer to orientations as its Euler angles representation, which is a standard used on motion capture files.

A character pose is defined as the vector that contains all the position and orientation information of all of our joints. The number of elements on this vector

is referred as the degrees of freedom for this character. Two poses are considered continuous if the difference between its corresponding vectors is small and it doesn't introduce self-collisions when played. A set of continuous poses generates a motion clip. The amount of poses in a second of a motion clip is called the frequency. If the application requires a higher frequency, new poses are usually generated by linear interpolation for the degrees of freedom involving position and spherical linear interpolation (Slerp) for the orientation ones. Slerp refers to an interpolation done over the quaternion representation of 3D rotation such that all of the points among the interpolated values lie on the unit-radius.

1.2. Character Animation

The main interest of this work involves character animation. One of the most important goals in this area is to generate realistic motion, a difficult task given that it depends directly on human perception, something not easily measurable.

Intuitively, humans are capable of perceiving when something is physically plausible. This applies to character motion, even on computer animated characters. It is possible to generate valid and sometimes realistic motion following only physics rules, however simulating all of the degrees of freedom becomes computationally expensive. On the other hand, it is possible to generate motion by following a set of simple algorithmic rules, like predefined motion cycles or by solving analytic kinematics equations for specific objectives. The problem here is that without anything externally guiding the algorithm, the quality obtained is usually poor. Algorithmic and physics-based approaches have taken advantage of the capacity to record real character motion from the real world. Using this data can speed up physical computations as well as control algorithmic approaches. The methods explored in this dissertation are algorithmically data-based.

Motion capture is a procedure that involves optical, inertial or mechanical sensors that record the movement of a subject. Given the non-objectiveness of quality of motion, we are unable to obtain a measure for our results. Throughout this dissertation we will consider unaltered *motion capture* as the highest quality possible and thus, the proximity the motion capture data will become our way to measure the overall quality of the synthesized motion. The objective of motion capture based methods is to generate granular controllers capable of synthesizing "*motion capture* quality level" motions. However, these methods are prone to generate unrealistic results when not done properly.

1.3. Data acquisition

Working with motion capture data is a challenge. The motion capture process is time consuming and there are a lot of considerations that need to be taken into account. Most motion capture systems give the data in a predefined ad hoc skeleton. Unfortunately, each system uses a different skeleton configuration and naming convention which complicate its use.

As mentioned before, any modification done over the original motion intrinsically reduces its quality, regardless of the method used. For this reason I decided early on

this work to use exclusively a single motion capture system. This allows the use of a unique naming convention and skeleton structure throughout all of this dissertation. A choice like this is common over animation systems and in no way reduces the generality of our methods.

One of the challenging tasks simulated in this work involves opening a door. Capturing open door motions from a person is difficult with any optical motion capture system, due to the visual occlusions that a door can generate. For this reason we decided to use the Xsens MVN Link, an inertial system for the capture of all of our data. It relies on 17 interconnected Inertial Measurement Units that track linear and angular motion using gyroscopes, magnetometer and accelerometers. Unfortunately, this system produces noisy data, particularly on the floor level joints and the global position of the character. The feet are affected due to the magnetic interference, even on rooms with no metallic objects around, it is common to have metallic pipes on floors and walls which often create difficulties on the capture process. Similarly, the global position that these systems compute is obtained by an estimation of the floor contacts and skeleton dimensions; the solution implemented introduces noise over the character location, even among two consecutive capture sessions.

The type of data that was used on this dissertation brings specific noise sources. This can be specifically addressed in my applications, however they will affect the generality of my solutions. Without the option of obtaining clean data or correcting it during process/synthesis, the only option available is to clean the data after capture and before any work on it. This can counteract the noise generated by each specific capture scenario. I did this in MotionBuilder, a professional motion editor, designed specifically for humanoid characters, however it has been a time consuming approach for every time that new data was needed.

1.4. Overview

In this work I developed a mobile manipulator engine capable of performing complex tasks while maintaining a certain level of quality and running at interactive rates. Commonly, research in Computer Animation has focused on solutions working on open spaces, ignoring completely the synthesis among obstacles and narrow spaces. Similarly, the environment interaction is usually performed by playing specific solutions over well-known problems. For example, when the task is to cross a door, usual approaches are to find a well-known location in which the task can be easily played and the playing it, however this ad-hoc solution avoids the problem completely and lacks flexibility to environment changes. A flexible approach needs to consider a great deal of independent tasks within the problem: walking to reach the door, slowing down at the approach moment, correct orientation of the body that allows the arm to reach the knob, avoiding the door while it is opening and finally traversing the opened gap without generating a collision on the environment. The framework presented addresses such complex scenarios. The proposed approach is divided in two main elements, one that controls the global motion of the character through locomotion and one that handles the interaction with the environment.

1.4.1. Locomotion. I begin with the development of a responsive locomotion control engine on chapter 3. Correct body placement is critical for performing a number of manipulation tasks and its importance can be observed even in very simple situations. For example, if a virtual human model is slightly far away from a target to be reached, the spine will need to bend towards the target in order for the target to be reachable. However, a simple step forward would probably eliminate the need to move the spine and thus bring the model to a more comfortable location to perform the action. Moving to a suitable position for manipulation is often preferable than performing a strenuous manipulation motion. One of the first problems that appears when working with locomotion controllers is that feet constraints are not respected automatically. Foot sliding, foot collision with the floor or legs self collision arise prematurely in several approaches. We know that cleaning up the data to avoid foot sliding is noticeable even at small scales of error [Pražák et al., 2011]. Similarly collision and penetration become noticeable in the context of motion quality. It is possible to observe that the majority of related work mentions several options such as Inverse Kinematics (IK) correction, annotations and feet constraints handled in order to achieve reported results. However, to my knowledge there is no dedicated in-depth study detailing and analyzing the methodology for removing common artifacts generated from motion synthesis. My work proposes a detailed methodology to well handle these problems and at the same time to measure how much motion validity is preserved in the target control space. The work that I present shows that by using only IK correction to remove artifacts, feet annotation and a single locomotion clip, a vast number of tasks can be performed and the resulting motion can keep certain desired properties. The presented solution appropriately deals with feet constraints and self-collision during clip deformation, and I introduce a methodology to quantify the covered area of a single clip under deformation with control over quality preservation metrics. This approach is also extended to different walking behaviors, which are needed for a the generation of more complex techniques.

1.4.2. Locomotion Planning. The locomotion controller presented in chapter 3 provides great flexibility over fine control and precise body location, allowing us to focus on a new approach for human motion planning that we explore on chapter 4. The main contribution on this area is our exploration of a different approach for character navigation with multiple behaviors. This problem is often addressed by decoupling in two independent stages: first, a path is planned in the free portion of the environment with a given clearance value, and second, steering behaviors are used to control a path following behavior while reactively adapting the behavior and trajectory of the character according to constraints detected during locomotion.

One considerable shortcoming of this approach is that the path planning stage does not take into consideration possible locomotion behavioral choices and trade-offs during the path computation. Therefore, behavior and trajectory adaptation are only decided reactively during path execution. When environments are cluttered with obstacles, this traditional approach often leads to characters employing difficult maneuvers in narrow passages while much more comfortable paths just slightly longer may be available.

We propose a new approach that incorporates behavioral capabilities of the character in a path planning stage efficiently executed at the navigation mesh level. The produced paths address trade-offs related to path length and navigation behavior for handling narrow passages. We present two different approaches to this method, the first one performs full-body collision detection during the planning while the second one generates a layered mesh that allows faster collision detection on 2D layers improving considerably the final computational speed.

1.4.3. Coordination. Finally, in chapter 5 we explore full-body interactions with the environment by integrating our motion planner with an upper body controller. Approaches to this problem are generally data-based. Unfortunately, in the case of complex interactions as the ones we addressed, data-based methods tend to require a large number of example motions to be carefully collected, which can easily degrade the overall system performance and the practicality of the solution. Furthermore, as motions are adapted to handle varied constraints, body coordination strategies embedded in the original data can be easily lost, leading to new motions which are either not able to perform the desired task or not plausible anymore for performing the intended interaction.

In order to overcome these challenges the proposed approach is based on learning a coordination model encoding spatial relationships between the upper-body, the environment, and body mobility. The main novelty of our method is that we synthesize full-body interactions by coordinating upper-body motions with a generic walking controller. In this way the proposed approach is able to achieve successful complex full-body interactions by applying a coordination model that is focused on learning upper-body coordination strategies while walking. The approach requires only a few example motions as input, and leads to a powerful and practical method suitable to real-time applications. For example, our method is able to replicate the complex orchestration of body movement strategies required to successfully open a door with a generic walking controller.

CHAPTER 2

Literature Review

Early works on computer animation, like Lasseter [1987], began by describing the principles used in traditional animation and how they translate to its algorithmic interpretations. Animation work involved a lot of manual labor and experience in order to be performed successfully. Since the 1930's decade there was a need for tools capable of performing the animation tasks in easier and faster ways. However, with the continuous evolution of the entertainment industry, the demand for streamlined methods for animation grew higher than the possibilities of technology of that time. With the surge of personal computers and graphical user interfaces, computer animation became available and computer generated films start to appear more on the media.

2.1. Character Animation

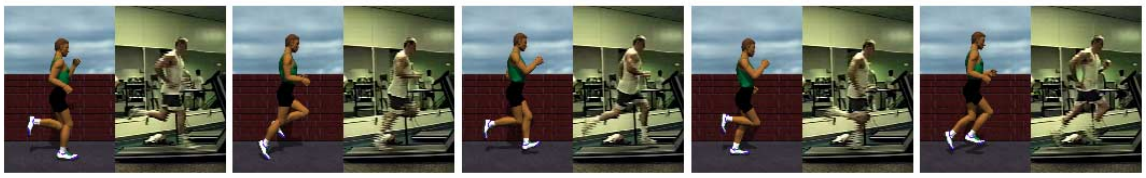

FIGURE 2.1. Running animation by Hodgins et al. [1995]. ©1995 ACM

This dissertation focuses on Character Animation. Early research on Character Animation was task-focused, for example Raibert and Hodgins [1991] showed a simplified model for single leg animation using physics-based controllers. Similarly, Hodgins et al. [1995] presented a series of algorithms for athletics animation with physics controllers in order to imitate real motions as we can see in Figure 2.1. This models evolved and integrated captured data on them, a recent summary is discussed in the following section.

One of the goals of Computer Animation is to reduce the complexity and processing time needed for the task of locomotion control. Most systems nowadays are not suitable in terms of implementation complexity and desired runtime. Many real-time applications assign very low computing power and memory to the locomotion control, and close to none for secondary characters. With very limited resources, most solutions lack the capability of performing a variety of tasks or depend completely on the captured data. There are a number of known approaches to the problem, usually they rely on a combination of precomputing, big memory storage and demanding computing power. Nonetheless none gives a simple, low resource solution with an analysis on how well it performs under given orders. There is an extensive amount of

past work on motion capture synthesis and locomotion, and the related work listed here is limited to a selection of representative works in the area.

2.2. Motion Graphs

One of the goals in the synthesis of Character Animation is to generate not only a feasible solution, but the most natural one. This calls for the use of motion capture data. However, data itself falls short in most tasks. Creating controllable and realistic motion was achieved by *Motion Graphs* [Kovar et al., 2002; Arikan and Forsyth, 2002; Lee et al., 2002; Arikan et al., 2003; Gleicher et al., 2003]. The idea behind is that human motion is intrinsically cyclic and that there is a lot of repetition, specially if the data is analyzed pose-by-pose. This allows an abstraction of the human movement by segmenting datasets of motion capture into discrete motion clips that can be connected among themselves creating a graph in which each node is a motion clip and each directed link represents that a motion can be concatenated after one connected to it. In this way, any random graph walk could generate a feasible motion and specific graph walks could achieve complex results like the on shown in Figure 2.2.

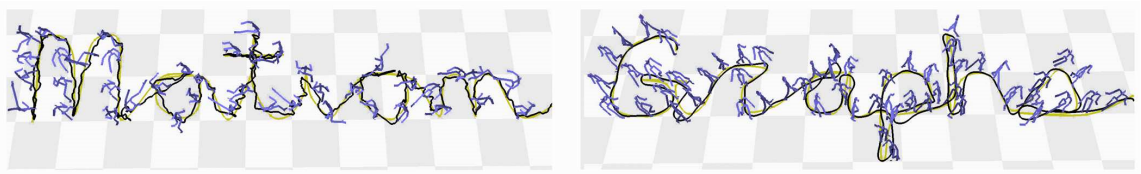

FIGURE 2.2. Motion Graph generated animation by Kovar et al. [2002]. ©2002 ACM

Motion Graph based methods suitable for real-time applications will typically rely on a collection of motion clips specifically organized for being concatenated in order to achieve a continuous stream of motions. Gleicher et al. [2003] and Park et al. [2004] worked on finding extra links in the graph by generating extra transitions among different clips using motion blending. Interpolation of motion paths explored the solution space of the approach [Safonova and Hodgins, 2007] but at the expense of increasing the size of the motion graph. Further improvements were proposed in Zhao and Safonova [2008] and Zhao et al. [2009] but mostly to construct motion graphs with improved connectivity. Approaches such as the work of Lau and Kuffner [2006] have relied on search pre-computation of navigation behaviors in order to obtain interactive search performances. Kovar and Gleicher [2004] automatically extracted similar motions from large databases to then build parameterized motion skills. Rose et al. [1998] constructed blend spaces with radial basis functions in order to generate motions of different styles. van Basten and Egges [2011] built a coordination model which realistically combined spliced motions. Heck and Gleicher [2007] explored the parametrization of the motion clips in the nodes of the graph to increase the flexibility of the graph without increasing its size.

The overall approach can also incorporate parameterization strategies for achieving precise control of the modeled behaviors [Mukai and Kuriyama, 2005; Heck and

Gleicher, 2007; Min and Chai, 2012]. Such approaches can be suitable for controlling the mobility of virtual characters; however, motion parameterization still requires a significant amount of well-designed motion clips to be interpolated in order to achieve an effective parameterization of the involved motions. Procedural warping of motion with blending was accomplished in Johansen [2009]. In van Basten et al. [2010] they interpolate from a database to generate footstep locations, they have a great control over the movement but it requires a dense database and postprocessing after motion interpolation. Feature-based motion graphs were developed by Mahmudi and Kallmann [2011] and Mahmudi and Kallmann [2013] which used geometric features to create transitions among the motion graph and introduce semantical information within the motion graph with an non-degrading deformation model.

The main drawback of motion capture based methods is that they involve time-consuming search procedures with limited applicability to real-time applications. Another difficulty faced by some of these methods is that a significant amount of motion capture data may be required in order to be able to address complex environments or other types of constraints.

2.3. Physics-based animation and control

Another technique used for locomotion synthesis and control relies on the fact that the physical nature of the movement of a lot of characters can be described by physical equations. A *perfect machine* would produce optimized motion such that the effort made by its actuators is minimum. Humans perform a similar optimization on the motion they perform. Therefore, an attempt to minimize an objective function that could describe the human motion could in theory generate realistic motions. Methods like this started early, on simplified models for character legs that were moving in a 2-dimensional space [Raibert and Hodgins, 1991; van de Panne et al., 1994]. This allowed the computers back then to compute the necessary forces to simulate realistic motion, however the computational constraints of these methods limited their growth. Nonetheless frameworks like the one presented by Faloutsos et al. [2001] integrated physics on character animation allowing the growth in this field.

Once mature physics-based frameworks were available in the research comunity, animation systems allowing dynamic control gained popularity. Yin et al. [2007] showed elaborated systems that allowed the integration of locomotion and PD control with the advantage of interactivity with unexpected environmental disturbances. A more general system was presented by Coros et al. [2010] which extended the capabilities to upper body actions and varied character proportions. On Bai et al. [2012] they present the problem as a dynamic upper body manipulator that integrates a simplified physics simulation with a multitask planner to achieve a list of interaction tasks in a motion-plan.

Zimmermann et al. [2015] presented a motion system that segments a motion planning framework into a hierarchical structure depending on the overall problem complexity. This allows adaptability to multiple constraints while executing difficult tasks. The proposed subdivision can be seen in Figure 2.3, each structure uses a

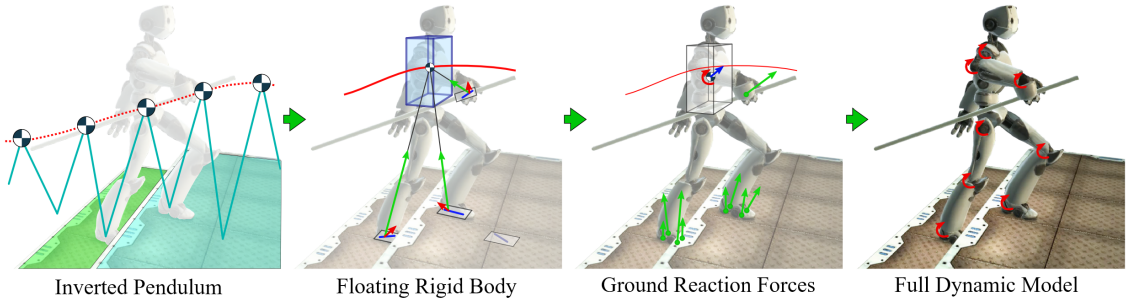

FIGURE 2.3. Different physical models used by Zimmermann et al. [2015]. ©2015 ACM

physical model representation of varied complexity to solve the synthesis with an inversely proportional constrain for time computation.

Physics-based animation techniques share the need to work with high-dimensional problems that often require numerical optimizations. Modern games have taken advantage of the computing power of current technologies allowing the integration of some physics simulations, however dynamic controllers have not been fully implemented into commercial systems due to the considerable computational power that they demand.

2.4. Data-based approaches

The use of motion capture data is not restricted to elaborate reproduction and system integration. Treating motion capture data as a series of vector values has brought interesting solutions in computer animation.

2.4.1. Motion Blending. An initial approach is to interpolate different motion capture vectors to find suitable in-between poses, this is called Motion Blending. Rose et al. [1998] proposed creating an interpolation space from motions of similar actions performed on different styles. Kovar and Gleicher [2004] focused on the automatic discovery of potential groups to build parameterized actions by proposing a new similarity metric. In some cases, like on van Basten and Egges [2011], the motion blending is used to merge motion from different sections of the body.

2.4.2. Reinforcement Learning. A recent trend on character animation control has approached the problem as a policy problem. The state is represented by the type of control that is desired(such as speed, direction, avoidance, etc.) and the action is shown as the concatenation of clips. This generates a cost that is based on the desired transition given a required strategy selected.

Treuille et al. [2007] initially presented this solution showing a greedy graph based search for control selection. Lo and Zwicker [2008] used Motion Planning over a sampling tree generated with Reinforcement Learning for the navigation. Lee et al. [2009] encapsulated a set reinforcement learning based controllers to allow for more than one goal. Lee et al. [2010] extended this idea to apply it in motion representation. Levine et al. [2011] explored the generation of motion plans using this

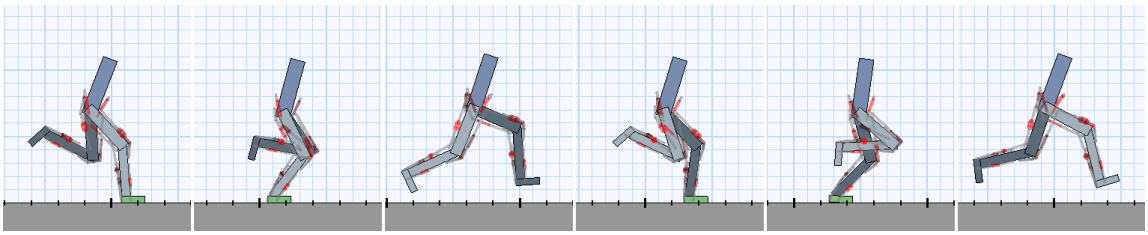

FIGURE 2.4. Peng and van de Panne [2017]. ©2017 ACM

type of controllers. In Levine et al. [2012], the authors focused on representing the motion in a low dimensional embedding. While Shum et al. [2014] focused on preparing the current action to blend seamlessly with the following one. Peng and van de Panne [2017] recently presented a control mechanism based on deep reinforcement learning using physics-based synthesis. Figure 2.4 shows some generated poses as well as a representation of the leg actuators.

Besides being complex implementations, these works rely on big databases and considerable precomputation time, which have limited their use on commercial systems.

2.5. Motion Planning

A number of planning-based methods have been developed with the purpose of generating character locomotion among obstacles. One of the first approaches was based on deforming animation data to cope with constraints while searching for paths using a probabilistic roadmap [Choi et al., 2002,0]. Another approach was also developed for supporting cooperation between characters [Esteves et al., 2006]. A more recent approach uses specialized deformation techniques achieving a powerful mechanism that can be explored during navigation [Choi et al., 2011]. Another approach is to plan the whole-body motion and then employ a data-based procedure to improve the realism of the result [Zhang et al., 2009].

Other planning methods have been proposed based on synthesizing arm manipulations on top of motion graphs [Mahmudi and Kallmann, 2015], based on a hierarchical model decomposition with sampling-based planning [Pan et al., 2010], and based on arm planning synchronized with unmodified motion capture clips [Shapiro et al., 2007].

Agrawal and van de Panne [2016] propose techniques based on footstep templates on relatively large motion databases in order to be able to achieve coordinated tasks, but depends on the availability of specific footstep templates for each action. We can see an example of a footstep plan in Figure 2.5.

On the other hand, efficient path planning techniques on navigation meshes have been mostly developed independently from full-body locomotion behaviors, and only considering clearance on the 2D floor plan of a given environment [Kallmann and Kapadia, 2016].

Several planning methods have been developed in the robotics domain. In particular, a multi-modal planning approach was used by Hauser et al. [2007] to generate motions for humanoid robots. Kallmann et al. [2010] introduced the coordination of parameterized motion skills to generate mobile manipulations.

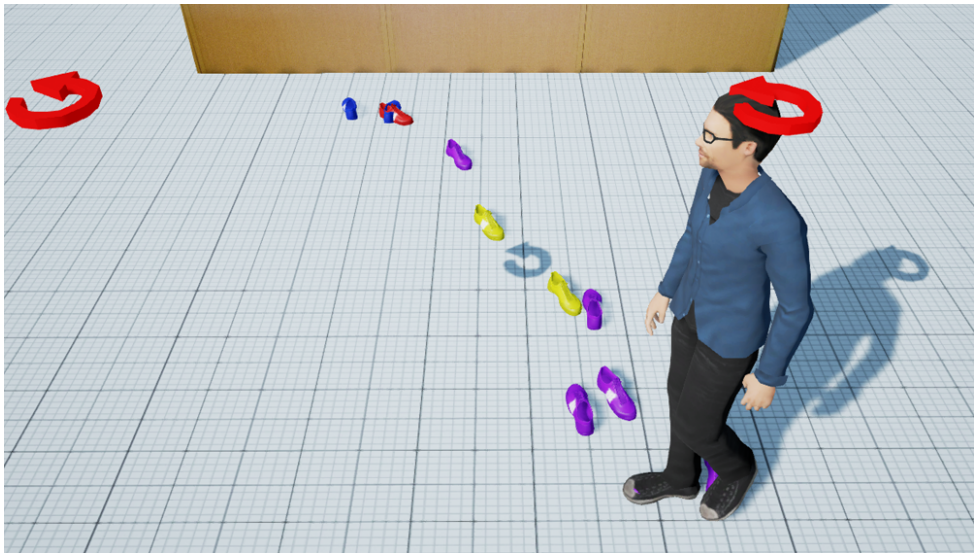

FIGURE 2.5. Agrawal and van de Panne [2016]. ©2016 ACM

Unfortunately, motion planning becomes computationally hard as the dimension of the problem increases. Limited by this and the amount of degrees of freedom of a normal character, the area explores instead decomposing the problem into a treatable one and integrating it them into the final synthesis. Another limitation on these methods is that most rely in some type of randomized sampling, which affects the realism of the result. There are blending techniques that can alleviate part of this problem, however generating realistic results requires more than just smooth motions.

2.6. Environment Interaction

Environment and object interaction introduces a new degree of complexity in motion synthesis. Data-based approaches need to introduce the interaction as a constrain required for any modification, while interaction for physics based solutions add external forces into dynamics computation.

One approach for achieving object manipulation is to first plan the motion of the object being manipulated, and then adapt upper-body postures to follow object movement, this was presented by Yamane et al. [2004]. Lee et al. [2006] work on includying the environment interaction into an interaction motion graph. Shapiro et al. [2007] approached the problem by modifying the motion capture data using a randomized planner to adapt the recorded interaction to new situations. Ho et al. [2010] proposed a distance mesh between the object and the character to describe the interaction, while Al-Asqhar et al. [2013] expanded the idea into a spacial based approach.

Object interaction under physics was explored initially on an acton-dependant basis. First by involving hand manipulations [Liu, 2009]. Then upper-body manipulations [Jain and Liu, 2009]. Finally, it moved into full-body manipulations [Abe and Popović, 2006; Jain and Liu, 2011], demonstrating that realistic results can be

achieved. Bai et al. [2012] proposed a framework that includes locomotion and planning for environment interaction.

Currently, there is no way to successfully generalize all possible object and environment interactions that can be performed by a virtual character. All of the presented solutions (and many more) work over the direct examples to which they were designed and they cannot adapt too far from its main application without the need of specific data or an ad hoc model.

2.7. Conclusion

In summary, while many works have provided the capability of adaptive locomotion and real-time controllers, adapting and interacting over a complex environment still remains a challenge. Our approach addresses this challenge by integrating locomotion, body motion planning and a coordination module into a complete framework capable of addressing a number of complex situations.

CHAPTER 3

Locomotion

While reaching and grasping constitute the main components of an object manipulation, there are several situations in which precise coordination between manipulation and body mobility is crucial for achieving a virtual human model capable of positioning itself in a suitable location for performing the intended manipulation.

Body positioning for manipulation is particularly important in whole-body manipulation tasks like the ones we will explore on chapter 5. A number of factors influencing a chosen position have been observed [Land et al., 2013], indicating a complex whole-body posture planning process in anticipation to a manipulation task.

In this chapter we focus on the particular problem of parameterizing primitive data-based mobility controllers with known coverage and quality properties, in order to allow higher level body planning mechanisms to select and employ the best mobility controllers to support a given whole-body manipulation task.

The proposed mobility controllers are based on deformation operations applied to a captured motion clip of a single step motion. The motion can be one cycle of a patterned locomotion clip, such as walking, or a single step towards a rest pose, such as by a lateral or frontal step. We use the term mobility to refer to both locomotion and stepping behaviors.

A small number of motion deformations are then defined in order to achieve full parameterization of the end location of each mobility controller. Controllers are quantified with respect to end location coverage and motion quality, providing valuable information for controller selection during whole-body manipulation planning. Initial results obtained from applying the proposed mobility controllers for the task of opening and passing through a door are also presented.

Our proposed approach is based on simple deformation strategies applied to a single motion capture clip. The simplicity of the approach allows it to scale well to multiple controllers integrated as sub-components under a higher-level whole-body motion planner. The proposed approach is based on motion adaptation techniques which can be efficiently computed and thus are suitable for achieving real-time results.

3.1. Related Work

In this chapter we are not only interested in achieving a real-time data-based mobility controller but also in defining a methodology for quantifying control coverage and quality of the produced motions. While previous work has not yet specifically addressed quantification metrics for mobility controllers, the area of evaluating perception of motion quality has received increased attention in recent years. For instance, Pražák et al. [2010] and Ryall et al. [2012] have evaluated the perception

of time-warping motion operations, and Pražák et al. [2011] have investigated the perception of foot-skating artifacts. While our current quality metrics are limited to validity tests, our methodology can be easily extended to also include perceptual metrics.

The primary interest of the proposed mobility controller is to offer precise control with known quality for whole-body manipulation planners. In computer animation different whole-body motion planners have been proposed [Bai et al., 2012; Huang and Kallmann, 2016] which require the existence of a suitable locomotion controller. Previous work in digital human modeling has also recognized the importance of mobility in manipulation tasks and the difficulty in addressing the mobility component. This proposes a methodology with the potential to improve generic approaches to full-body motion planners. The principles of this approach have been presented in [Juarez-Perez et al., 2014; Juarez-Perez and Kallmann, 2016b].

In summary, while several previous methods have investigated different techniques capable of achieving realistic mobility controllers, our proposed approach consists of a simple methodology that includes precise determination and quantification of coverage and quality properties in order to support the needs of whole-body manipulation planners.

3.2. Data-Based Mobility Controllers

Using the full-body motion capture suit described on Section 1.3 we have collected a series of stepping motions and we have devised a methodology for parameterizing captured stepping motions or locomotion cycles. Each step cycle of interest is segmented out of the motion and processed in order to obtain a parameterized primitive for stepping towards any point inside a neighborhood around the original end-point of the captured clip. Independently if the clip represents one locomotion cycle or a single stepping motion, clips can be self-concatenated, which means that the final frame of the clip should be similar to the first frame of the clip. We refer to the clip as the motion cycle.

3.2.1. Cycle Deformation and Parameterization. Each motion capture clip fully specifies the motion cycle. It describes the topology of the skeleton, the location of its joints, and the joint values for each frame of the motion. The joint values are the joint rotations at each frame, and a special joint, called the root joint, also contains the position of the skeleton in world coordinates. The root joint is usually placed between the hips of the character and its rotation values describe the overall skeleton orientation.

The root position and its rotation component around the vertical axis are the main elements manipulated by our deformation operations. By controlling these parameters we are able to deform the overall path that the animation follows. Any undesired artifacts (like feet sliding) that are eventually introduced in the motion are corrected algorithmically, as later described in this section.

Each captured cycle can be deformed in 3 different ways: forward stretching or compression, left-right trajectory turning, or left-right final body orientation change. These three operations introduce a parameterization of the original motion, but at

the cost of potentially reducing the realism of the original motion. Our analysis methodology, presented later, will represent the trade-off between coverage and quality. During real-time deformation operations, the system will always correct the root joint trajectory such that a continuous self-concatenated motion cycle is always achieved.

Deformations are applied over the root position and orientation of the character skeleton hierarchy. We consider the root to be the joint to which the hips and spine are connected to in the humanoid skeleton representation. Consider the motion cycle ground trajectory being represented by M as a vector containing indexed root position and orientation values $\rho_i = \{x_i, z_i, \theta_i\}$ for every frame i of the motion, where θ_i represents the yaw rotation (or the body orientation) of the character and (x_i, z_i) is the projected position of the root joint on the floor plane. Therefore the full motion trajectory is represented by $M = \rho_i$, $i = 1, \ldots, N$, where N is the number of frames in the motion cycle.

We perform three kinds of deformations:

- **Trajectory Turning.** To achieve turning, we update the motion frames with the result of applying a 2D rotation by a given rotation parameter ϕ to $v_i = (x_i, z_i) - (x_{i-1}, z_{i-1})$, and adding ϕ to θ_i. The 2D rotation by ϕ is a rotation around the vertical axis of the character. The trajectory of the character will turn to the left or to the right according to the rotation parameter.
- **Compression or Stretching.** Translation operations are applied at each frame i of amount s over the displacement vector $v_i = (x_i, z_i) - (x_{i-1}, z_{i-1})$, by updating the position of each (x_i, z_i) such that the length of the displacement vector is updated to become $||v_i|| + s$ after the displacement is applied. This is achieved by replacing (x_i, z_i) with $(x_i, z_i) + s\frac{v_i}{||v_i||}$. The resulting deformation will either compress or stretch the original motion. This transformation includes an optional speed adjustment, where the timing of the keytimes associated with the frames can be multiplied by a factor proportional to the deformation. This allows to maintain the speed as the original motion speed while stretching or compressing the motion, or to allow to change the speed according to the performed deformation.
- **Final Orientation.** In order to change the body orientation at the end of the clip, at every frame, the value of ψ is added to θ_i. The final orientation of the character changes but the trajectory of the motion remains the same.

The result of the described deformations is illustrated in Figure 3.1. In order to achieve a parameterization independent of the number of frames in a clip, the parameters sent to the entry point of each deformation procedure are considered to correspond to the whole motion and thus they are multiplied by a factor of $1/N$, where N is the number of frames in the motion clip, in order to obtain the per-frame deformation parameters that are used by each described deformation procedure.

These deformations can be applied to a large variety of stepping motions and together they are able to fully control the final location of the step cycle while maintaining the same style of the captured cycle. Note that the control is on the final position and orientation of the skeleton root joint position projected on the floor,

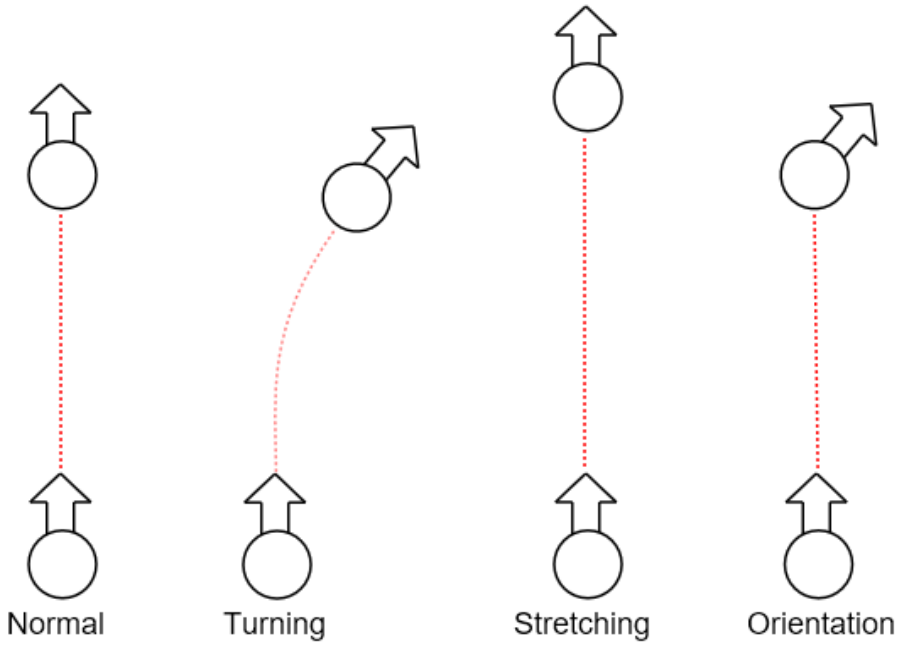

FIGURE 3.1. Deformations over a cycle. The left-most diagram corresponds to the original clip cycle. The next ones represent the outcome of each deformation operation

and not on individual footstep locations, what would impose a higher number of parameters to be controlled.

For example, with a 90 degrees turning angle and a 90 degrees orientation angle, we end up with a lateral stepping motion cycle, although not realistic and with self-collisions introduced. If we apply a compression deformation such that the total ground displacement becomes 0, and also apply a given trajectory turning deformation value, we obtain in-place turning. By providing a constant turning rotation, we achieve the character performing a circular trajectory. Clearly, *extreme* motions, like U-turns, will not look realistic and we later introduce coverage-quality maps in order to determine quality boundaries for the control space.

3.2.2. Enforcement of Feet Constraints. The presented deformation operations provide parameterization of the motion cycle but at the same time introducing several undesirable effects, in particular related to breaking original feet-ground constraints. These discontinuities are corrected with a Inverse Kinematics (IK) and blending operations.

We first annotate the moment at which the ball of the foot reaches a stable ground support position in the motion (achieving foot contact) and the moment at which it loses this property (foot release). In this way, the parameterization deformations will take advantage of a possible small twist motion introduced over the first contact point with the floor. Each foot will have four different states: Transitioning, Landing, Supporting and Departing.

By using IK corrections and blending operations over the stages we ensure continuity over the motion while maintaining the capability of deforming the motions as desired. Each foot cycle is processed as follows:

- **Transitioning Phase.** During this phase one foot is transitioning in the air between two ground contact points while the other foot remains fixed on the ground as the support foot. During this phase no constraint correction is needed for the transitioning leg; however, IK correction for the upcoming contact will gradually update the leg joints, in an ease-in blending fashion, as it approaches its next ground contact point. This will ensure that the transitioning foot lands exactly at the expected next ground contact point, which is re-computed at every pose based on the current user-controlled deformation parameters.
- **Landing Phase.** This phase starts at the first pose initiating ground-foot contact, right after the corresponding foot finishes its transitioning phase. During this phase the target end-effector for the IK continues to be updated according to any real-time deformations that may happen, and IK continues to be applied to correct the degrees of freedom of the leg until the foot stops and becomes completely fixed on the ground by IK.
- **Supporting Phase.** During this phase the supporting foot is completely enforced by IK correction to the position and orientation of the established ground contact. The original leg posture in the motion cycle data is used to inform the IK solver to produce a solution as close as possible to the original data.
- **Departing Phase.** This phase starts at the time of foot ground release, when the system gradually interpolates out of the IK-corrected configuration and returns towards the original data from the captured motion clip.

These four phases represent natural stages observed in human motion. Whenever we suddenly change speed or direction, the foot on the landing phase applies some extra force to maintain the balance while the position and orientation of the root is deformed, making a slight movement over the initial contact of the step. The Supporting phase is necessary to displace the body, and the Departing phase completes the cycle making the motion ready for concatenation. See Figure 3.2.

3.2.3. Continuous Cycles with Blending Operations. The process of concatenating the motion cycle in order to achieve continuous stepping control will naturally introduce discontinuities at the connection point because the original unprocessed motion cycle clip will never perfectly self-concatenate. To address this problem we apply motion blending at the transition points.

We perform a special blending operation for each step during the concatenation phase. The transitioning leg in the transitioning phase only requires an ease-in and ease-out of the motion but the leg that is in support/contact requires an extra correction. For this procedure we will need to know the specific position and orientation of the foot geometry, this can be represented in a 4x4 matrix in global coordinates which we call the global matrix. We compute the global matrix $\boldsymbol{F}$ of the contact foot at the end of the clip, and the global matrix $\boldsymbol{G}$ of the same foot at the start of the

subsequent clip. If concatenation blending is being performed, there is going to be a twisting and sliding motion at the foot during the concatenation, and so we use matrix $\hat{\boldsymbol{F}} = \boldsymbol{F}\boldsymbol{G}^{-1}$ over the subsequent clip in order to correct the global position of that foot and eliminate any undesired foot sliding or twisting effect. This makes the position and orientation of the contact foot at the point of concatenation a constant, avoiding discontinuity on the contact spot. The $\hat{\boldsymbol{F}}$ matrix computed above is then used by our IK solver to restore the end-effector positions of the motions being blended to their correct place, such that the next blending operation will not generate any errors.

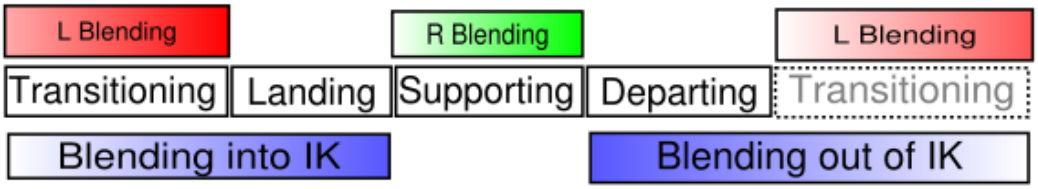

FIGURE 3.2. Blending operations according to cycle phases.

Figure 3.2 illustrates the main phases and the blending operations that are performed. Blending is used to perform two operations: concatenation and feet constraint enforcement. All corrections are performed in real-time so that an input motion cycle can be directly used by the system without the need of special pre-processing. IK correction is performed when there is a transition from and to a contact foot, and is always gradually applied, via blending with the original data values.

Algorithm 1 - Locomotion Controller

Input: A time t and values for the trajectory deformation.
Output: The pose at time t after blending and IK are applied.

```
1: procedure WALK(t)
2:     motion.APPLY(t)
3:     targets = GETIKTARGETS()
4:     BLEND(t)
5:     APPLYTRAJECTORYDEFORMATIONS()
6:   T = Transitioning foot
7:   S = Supporting foot
8:   Apply the IK to S using the corresponding target.
9:   Blend the current pose of T with a pose using the corresponding target.
```

Algorithm 1 describes the order of the necessary computations per frame that our controller uses. It begins by applying the pose of the current motion at time t into the current pose. Then **GetIKTargets** computes the support foot global matrix that is in contact with the floor and the transitioning foot global matrix from which the transitioning foot will blend in or out depending on the current time of the associated motion. After that **Blend** will blend the current motion with the one it is being concatenated to ensure continuity of motion. Then we apply all of the deformations that the controller is performing. Finally the IK corrections over the previously computed targets will be applied to avoid foot sliding for the support foot and maintain the continuity of the motion for the transitioning one.

3.3. Coverage-Quality Maps

Clearly, when large deformations are imposed on the motion cycle, a number of problems may arise. We perform several measurements in order to quantify the quality of the motion after a deformation is performed.

3.3.1. Constraint Enforcement. If the distance from the IK end-effector goal to the hip joint, when maintaining feet constraints, is greater than the maximum reachable distance, a solution will never be found. However, small errors might be acceptable and an approximate IK solution may be good enough for cases where the difference is small. Because of the nature of the error, the best way to quantify it is to measure the sum of the distance errors whenever the IK cannot compute the exact solution. This gives us a quality metric inversely proportional to the IK error.

3.3.2. Continuity. The second possible problem when applying IK is to ensure continuity over subsequent frames which were independently corrected. Every frame is corrected by IK seeking to be as close as possible to the original data; however, it is still possible that consecutive frames have a noticeable difference between them. Instead of relying on expensive IK formulations for improving smoothness, our IK operations are kept analytical and highly efficient, and our overall approach will discard the extreme deformations that generate discontinuous motions.

3.3.3. Collisions. Finally, some of the deformations will cause the legs to collide with each other, and a collision detection test is required in order to reject these cases and achieve correct results. In our system each rigid segment of the leg is represented by a geometric capsule primitive for the purpose of collision detection. Mesh-based collision detection can also be employed when working with skinned characters.

The metrics described above are then used to quantify coverage and quality. Given a motion clip to be parameterized and deformed, the entire parameter space can be sampled and the corresponding quality for each 3-tuple deformation parameter can be quantified according to the metrics described above.

3.4. Results and Discussion

Figure 3.3 illustrates a typical coverage-quality map obtained for a step forward mobility controller with respect to the stretching/compression and turning deformations. The dark blue region represents the coverage of the fully valid region

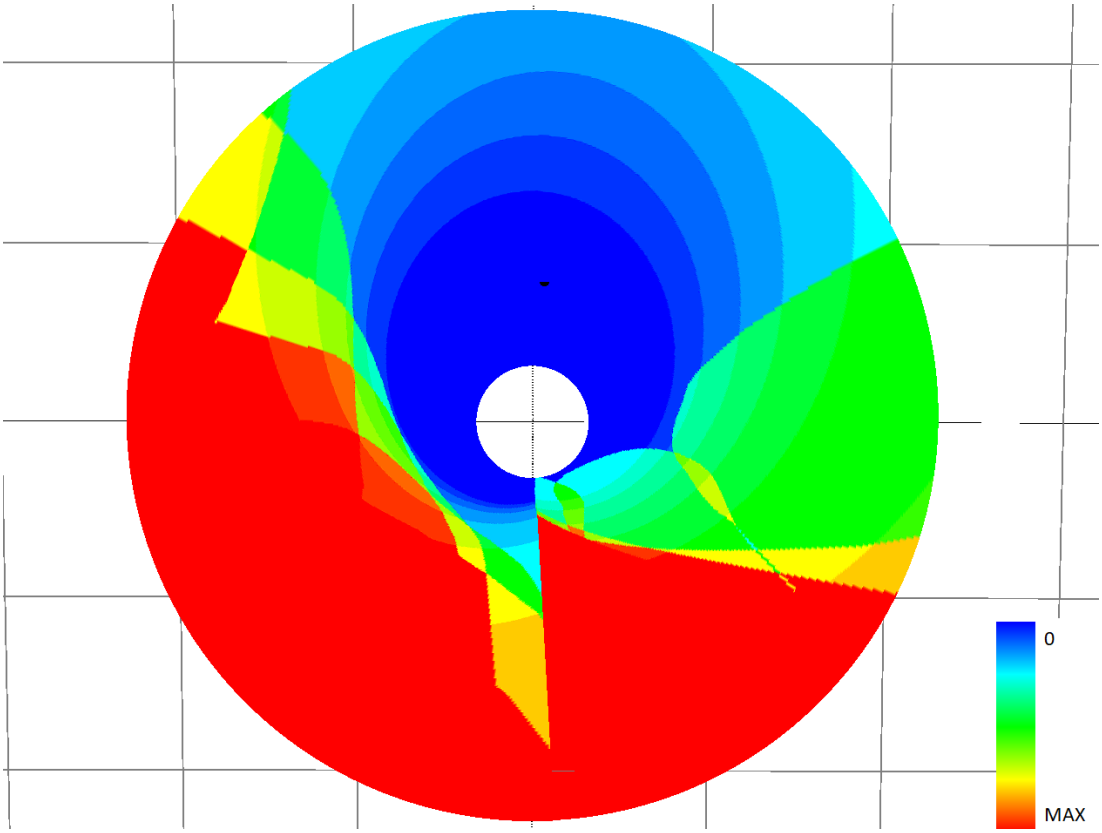

FIGURE 3.3. Coverage-quality map for one forward step cycle motion. The heatmap coloring represents the overall computed validity loss over the original data.

according to the employed metrics. The original motion clip is a single locomotion cycle forward step, where the starting position is at the center of the diagram with the virtual human model facing upwards. The dark blue region therefore indicates that a large frontal region can be reached by the presented deformation operations without imposing any loss of quantifiable quality, according to the employed metrics.

The colors that are not dark blue in Figure 3.3 represent regions where too much deformation was applied causing invalid IK operations or self-collisions to occur. The range of blue colors, from dark to light blue, represents the amount of IK errors detected, quantified as the number of frames where either a foot constraint or continuity between frames was not perfectly enforced. The range of colors green, yellow, orange and red represent the number of frames detected to have collisions. Green, yellow and orange represent 1, 2, and 3 frames with collisions respectively, and red means that 4 or more collisions were detected. The designed color visualization scheme allows a straightforward visualization of the coverage achieved for the desired motion quality output.

Figure 3.3 visualizes the coverage of the first two deformation operations. In Figure 3.4 we present the coverage achieved by the body orientation deformation.

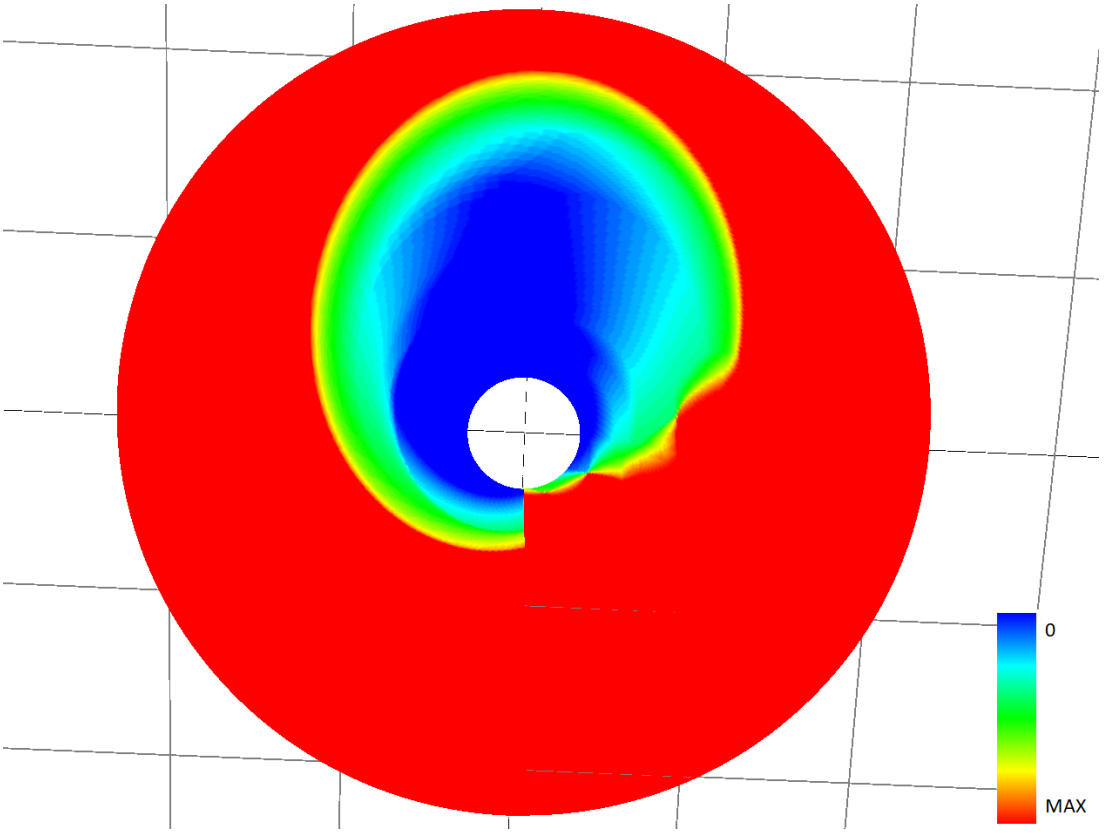

FIGURE 3.4. Final body orientation (after forward step) coverage-quality map.

The colors in Figure 3.4, for each final (x,y) body position, evaluate the quality of the full range of possible final body orientations. Dark blue represents positions at which the range of orientations between -75 and +75 degrees, sampled at 1 degree, are fully valid. Lighter blue to green are areas that have some invalid deformations but mostly valid ones. Red areas are the ones in which most of the deformation parameters return an invalid clip.

Figure 3.5 uses the same color scheme as in Figure 3, but applied to a lateral step. The original motion starts from the center of the diagram and performs a single lateral step to the right, with the end location being the point marked in black. It is possible to note that the valid deformation region is not symmetric and is larger in the area ahead of the character than in the area behind of the character. This corresponds to the body posture in the original motion clip slightly bending forward, in order to favor capacity to reach supporting location targets in the frontal region.

Figure 3.6 presents a volumetric visualization of the coverage-quality volume produced by the three deformation operations considered together.

One first utility of the presented coverage-quality maps is to inform how much control space can be addressed by a single mobility clip. The analysis easily allows us to determine new mobility clips in order to achieve a collection of clips fully covering the entire region around the character with high-quality (dark blue) motions. It is

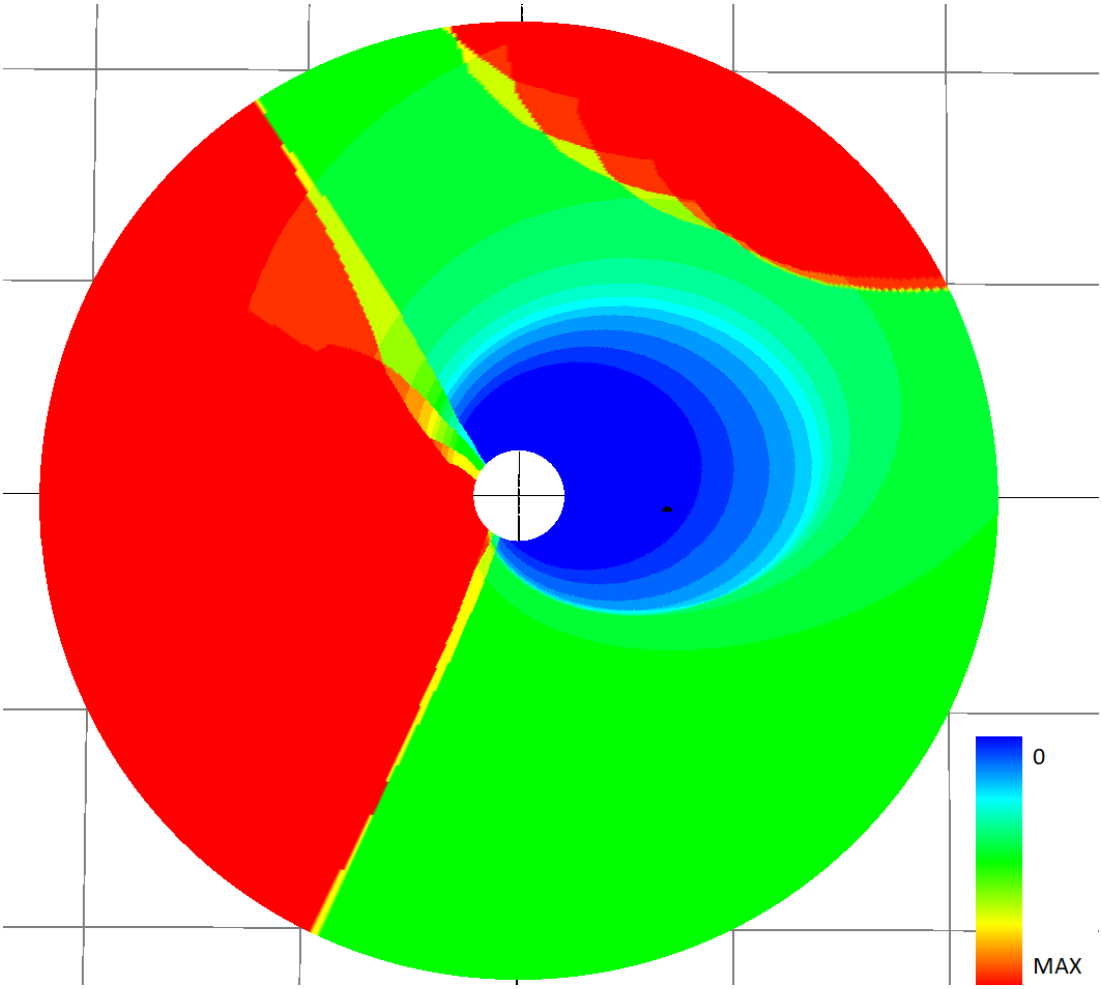

FIGURE 3.5. Coverage-quality map of one right lateral step.

possible to observe that a small set of forward, backward, and lateral steps will often fully cover the entire region around the character with dark blue.

The proposed set of deformation strategies enables not only parameterization of the end location of discrete steps, but also full on-line control of locomotion. For instance, Figure 3.7 illustrates our system being used to control a character turning around following keyboard commands executed by the user during simulation. Key presses are mapped to amounts of deformation in order to steer the character. Every time a key is pressed the current amount of deformation is incremented or decremented, and the corresponding deformation operation is applied instantly starting at the next frame being executed, with the corresponding blending operations being always activated.

The achieved flexibility makes the proposed mobility controller suitable for integration with higher-level planners for achieving whole body manipulations. This planners will be discussed in the following chapters.

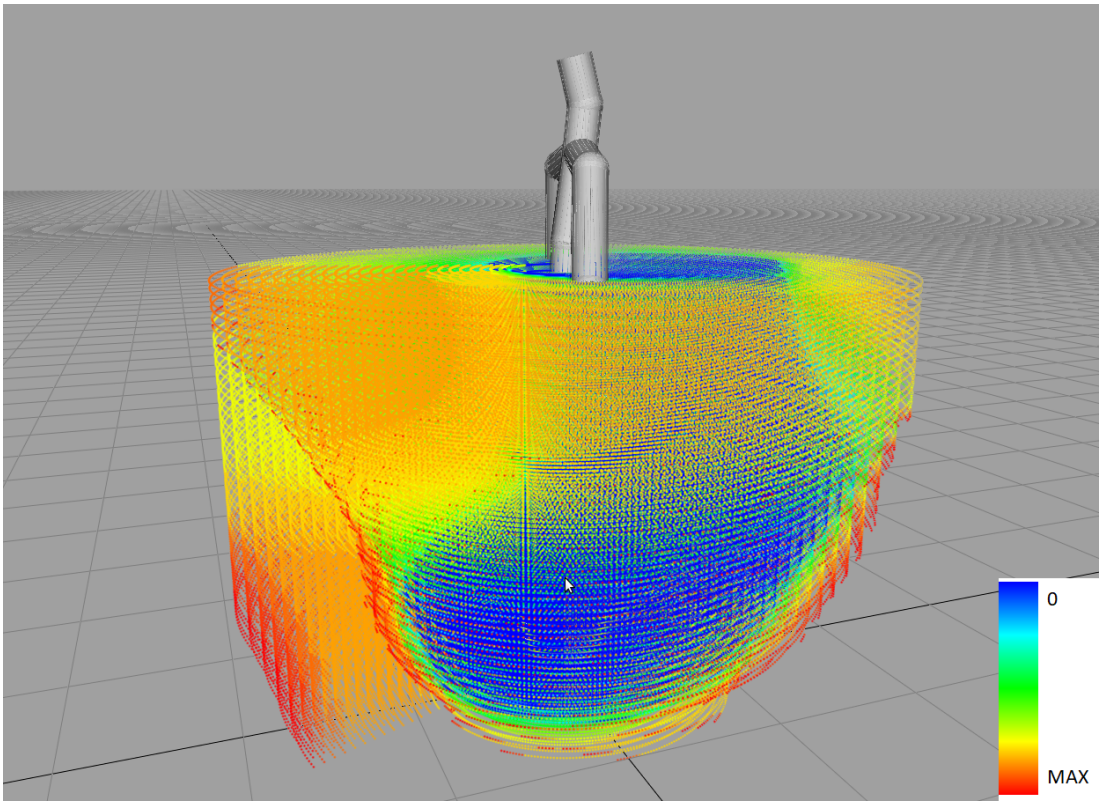

FIGURE 3.6. Volumetric coverage-quality map for a frontal step. The horizontal layers are planar maps equivalent to the one in Figure 3, with the vertical axis representing changes to the final body orientation deformation parameter.

3.5. Conclusion

The presented mobility controllers are able to successfully provide real-time control with known coverage and quality. The computation time required per frame is negligible considering that IK and blending is intrinsic in most character controllers. The coverage-quality maps can help into the design of the controller with prior knowledge of the potential results under extreme deformations.

The employed techniques represent computationally efficient solutions. The IK is computed efficiently using an analytical approach and the Motion Blending has an equivalent amount of operations. This makes the mobility controllers fast enough to be used to evaluate several possible trajectories for supporting whole-body manipulations.

This results allowed the creation of the motion planners discussed on chapter 4 as well as the full-body mobile manipulation explored in chapter 5.

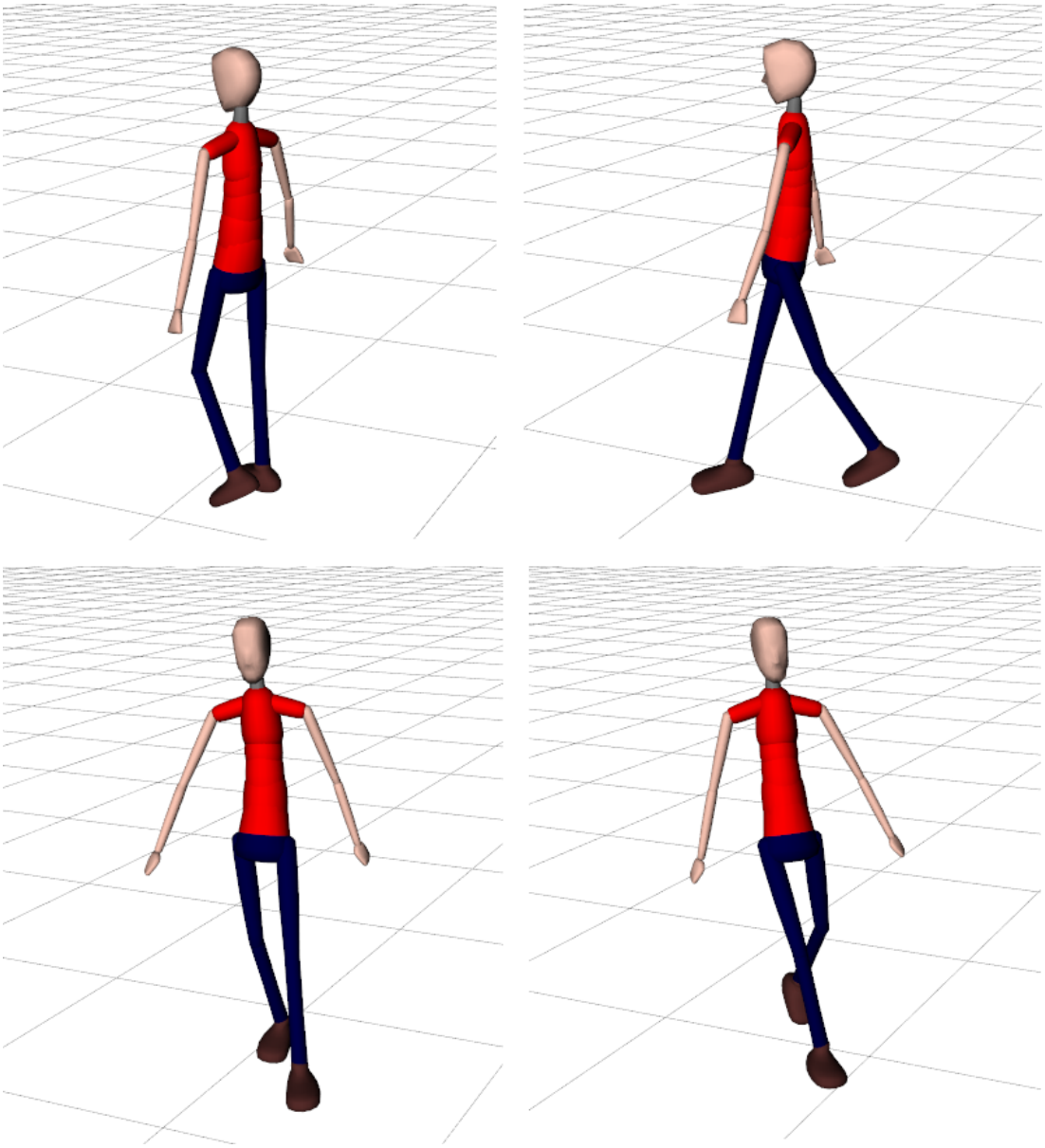

FIGURE 3.7. Example of real-time control of locomotion forming a circular trajectory.

CHAPTER 4

Locomotion Planning

In the previous chapter I presented a flexible locomotion controller capable of following curved paths precisely. This allows for a high-level control over the motion of our character. I can easily implement an application in which you can draw a path and the character would follow it precisely with the capability of returning a validity value over the followed path. A greater control would be to generate a system in which you only select initial and final locations and the character would find the most appropriate path and follow it. In this chapter I explore this type of application and I explore how to select which could be the most appropriate path to follow.

4.1. Introduction

Character navigation and locomotion in interactive virtual environments are often addressed by decoupling the overall problem in two independent stages: first, a path is planned in the free portion of the environment with a given clearance value, and second, steering behaviors are used to control a path following behavior while reactively adapting the behavior and trajectory of the character according to constraints detected during locomotion.

Unfortunately, when environments are cluttered with obstacles reactive approaches are less capable of addressing narrow passages and they can easily lead to characters employing difficult maneuvers in narrow passages while much more comfortable paths just slightly longer may be available. In the decoupled approaches the path planning stage does not take into consideration the behavioral choices of the character.

Our method relies on a character equipped with a collection of parameterized locomotion behaviors capable of performing path following. The output solution is an annotated path specifying which behavior to be used for each step along the path, according to the encountered environment constraints.

In this chapter we present two different approaches that combine the planning and reacting stage into a single motion planner. We follow the approach of performing successive path planning queries in a navigation mesh in order to plan paths incorporating the behavioral capabilities of the character.

Our first approach, published on [Juarez-Perez and Kallmann, 2016b], relies on full body collision detection. This is computationally expensive, therefore in order to remain efficient, our approach is based on planning over a customizable horizon in a greedy fashion, such that a first feasible path quickly found is continuously improved according to the defined behavioral preferences and rules, and at any given computational time threshold the current path solution can be returned.

We improve over our first approach on [Juarez-Perez and Kallmann, 2018b] by replacing the 3D collision checking by layered 2D clearance tests significantly improving computation times. This significantly improves the overall planning time making our method suitable for real-time applications and at the same time achieving paths addressing the trade-off between path length and preferred locomotion behaviors when handling narrow passages.

The main contributions of this chapter are three-fold. First, we propose a new method for planning navigation motions for interactive virtual characters in cluttered environments that require full-body adaptation to overcome narrow passages. Second, we introduce the concept of relying on fast navigation mesh techniques that dynamically update the navigation mesh in order to gradually impose behavioral and path constraints during the process of searching for the most suitable path. Finally, we present a method that simplifies the 3D collision queries into 2D layered clearance test which improves up to 20-fold on computation time speed-up in comparison to relying on full 3D collision checking.

4.2. Related Work

Navigation through an obstacle course has been explored by a number of motion planning based methods [Choi et al., 2003; Esteves et al., 2006; Choi et al., 2011]. While these methods are able to achieve impressive results, their focus is on the motion synthesis generation and not on improving the underlying trajectory planning method that is employed. Our proposed method focuses on this problem and proposes an improved path planner that considers constraints and trade-offs that are defined from the motion controllers.

Our proposed motion planning methodology can be applied to any given motion controller as long as it can follow paths with different navigation behaviors. While our motion controller has the advantage of relying on only one example animation per behavior, there are a number of possible data-based approaches based on larger motion databases [Heck and Gleicher, 2007; Lee et al., 2010], which have the potential to generate better-quality results. In any case, these methods could also be controlled by our proposed multi-behavior path planner.

Our multi-behavior path planner relies on path queries performed on an efficient navigation mesh representation called a Local Clearance Triangulation (LCT) [Kallmann, 2014]. It allows us to efficiently perform path queries and dynamic mesh changes during the process of searching for a path suitable for multi-behavior execution. Alternative navigation meshes can be employed as long as the employed operations are supported. Texts discussing approaches for navigation meshes are available from Kallmann [2016].

While several methods have explored the integration of planning capabilities with full-body motion synthesis, no previous work has addressed the approach of relying on an efficient navigation mesh representation in order to speed up the computation.

There is extensive work on motion planning applied in robotics like the works of Hauser et al. [2007] and Kallmann et al. [2010]. These works however did not include data-driven techniques.

Blending motion examples in order to generate in-between examples is an effective way to solve object manipulation problems [Kovar and Gleicher, 2004; Rose et al., 1998; van Basten and Egges, 2011]. While blending can generate a variety of motions, integration with a locomotion planner has been only addressed once [Huang et al., 2011]. Our present work improves the approach and effectively integrates heterogeneous motion skills in a single global planning framework.

In summary, while powerful behavioral planning approaches have been proposed in the past for planning multi-behavior navigation in cluttered environments, they quickly become too expensive for interactive applications. On the other hand efficient methods based on path planning with navigation meshes have not been extended to take into account constraints or trade-offs emerging from multi-behavioral locomotion. The methods proposed in this chapter address these issues in an efficient way with the introduction of layered clearance tests, which replace full collision checks and achieve significant computation time improvements, making our method suitable for real-time applications.

4.3. Outline

We begin the rest of this chapter by introducing the locomotion behaviors that we use in both of out approaches on section 4.4, then we describe our full body 3D collision approach on section 4.5. After this we present our layered 2D approach on section 4.6. Finally a conclusion for this chapter is presented on section 4.8.

Our method relies on the availability of a multi-behavior character navigation controller and an efficient clearance-based navigation mesh path planner. Our proposed approach is to associate the available navigation behaviors with their clearance requirements and then employ the path planner with the different requirements such that the solution path will allow collision-free multi-behavior execution while remaining close to the shortest feasible path.

4.4. Locomotion Behaviors

The motion generation relies on a set of deformable motion capture clips. In our case the clips define three locomotion behaviors: regular frontal walking, arm-constrained frontal walking and lateral walking. The motion capture set contains annotated clips of each of the behaviors as well as the necessary transition clips between each behavior. The arm-constrained frontal walking is implemented by modifying the trajectory of the arms during the regular frontal walking so that they remain close to the body. Examples of these behaviors being used in our system are shown in 4.6.

The first layer of our controller is based on the techniques presented in Juarez-Perez et al. [2014]; Juarez-Perez and Kallmann [2016b]. The controller requires annotated transition points between data-based behaviors and as well the general direction of motion. Motion deformations are employed with small motion modifications applied incrementally at each frame of the motions, allowing us to precisely parameterize behaviors for achieving smooth path following. Whenever we need to detect if a deformed motion introduces collision we test if a limb of the character collides with itself or the environment.

We have evaluated the parameterization space of the locomotion behaviors employed in this work, which are based on incremental rotation and orientation deformations. The evaluation is based on the quality maps as shown in Figure 4.1. In the maps, red represents the areas that generate self-collisions while blue and green represent areas that generate motions with slightly unreachable goals for the employed Inverse Kinematics (IK) corrections which control feet constraints enforcement during the motion parameterization. The blue and green errors are considered imperceptible and represent regions that still produce acceptable motions. If we limit our deformations to lie inside the the green and blue region of the quality maps, we can control the locomotion behaviors with flexibility beyond the needs of the path following capabilities that are encountered in this work. In this way our parameterization is guaranteed to precisely follow the produced paths with good quality and without generating self-collisions, and such that leg-leg collisions during sharp turns do not occur.

The controller can be instructed to switch to a different behavior at any time and the necessary transition will be automatically applied. Motion blending is applied to smooth out any possible misalignment introduced by the deformations. At each behavior concatenation point it is possible to start following a new path starting at the current position and orientation and/or to switch to a different behavior. The goal of the behavioral path planner is to generate a multi-behavior path that can be executed by the locomotion controller.

4.5. Behavioral Path Planner

Our behavioral path planner receives as input a goal position, and it then computes an annotated path from the current position of the character to the goal position. If the goal position lies beyond a distance threshold h, the planner will only perform the path annotations and modifications up to this given distance h. This distance controls the planning horizon of the overall behavioral planner.

We rely on a the local clearance triangulation (LCT) navigation mesh, which supports computation of paths with arbitrary clearance and dynamic insertion and removal of degenerate obstacles defined as a single point. This feature is used for adding constraints to the navigation mesh that reflect limitations of the locomotion behavior of the character, forcing the LCT path planner to search for alternate paths with new requirements.

We define the available behaviors of the locomotion controller as the set of n behaviors $\{\mathcal{B}_1, \ldots, \mathcal{B}_n\}$, each with its correspondent clearance value c_i, $i \in \{1, \ldots, n\}$, such that $c_1 < \cdots < c_n$. We define $\mathcal{B}_n$, our last behavior, to be the preferred locomotion behavior. In our case $\mathcal{B}_n$ represents regular frontal walking. Behavior $\mathcal{B}_1$ is defined as the least preferred behavior, usually an uncomfortable behavior to execute, but the most appropriate navigation behavior for narrow passages. In our case $\mathcal{B}_1$ performs lateral walk (or side-stepping). Any behavior in between will represent a trade-off between capabilities of navigation and preference of execution. In our current setup, we only have one intermediate behavior, $\mathcal{B}_2$, which is a regular frontal walk but with the original arms swing motions modified to stay close to the body of the character,

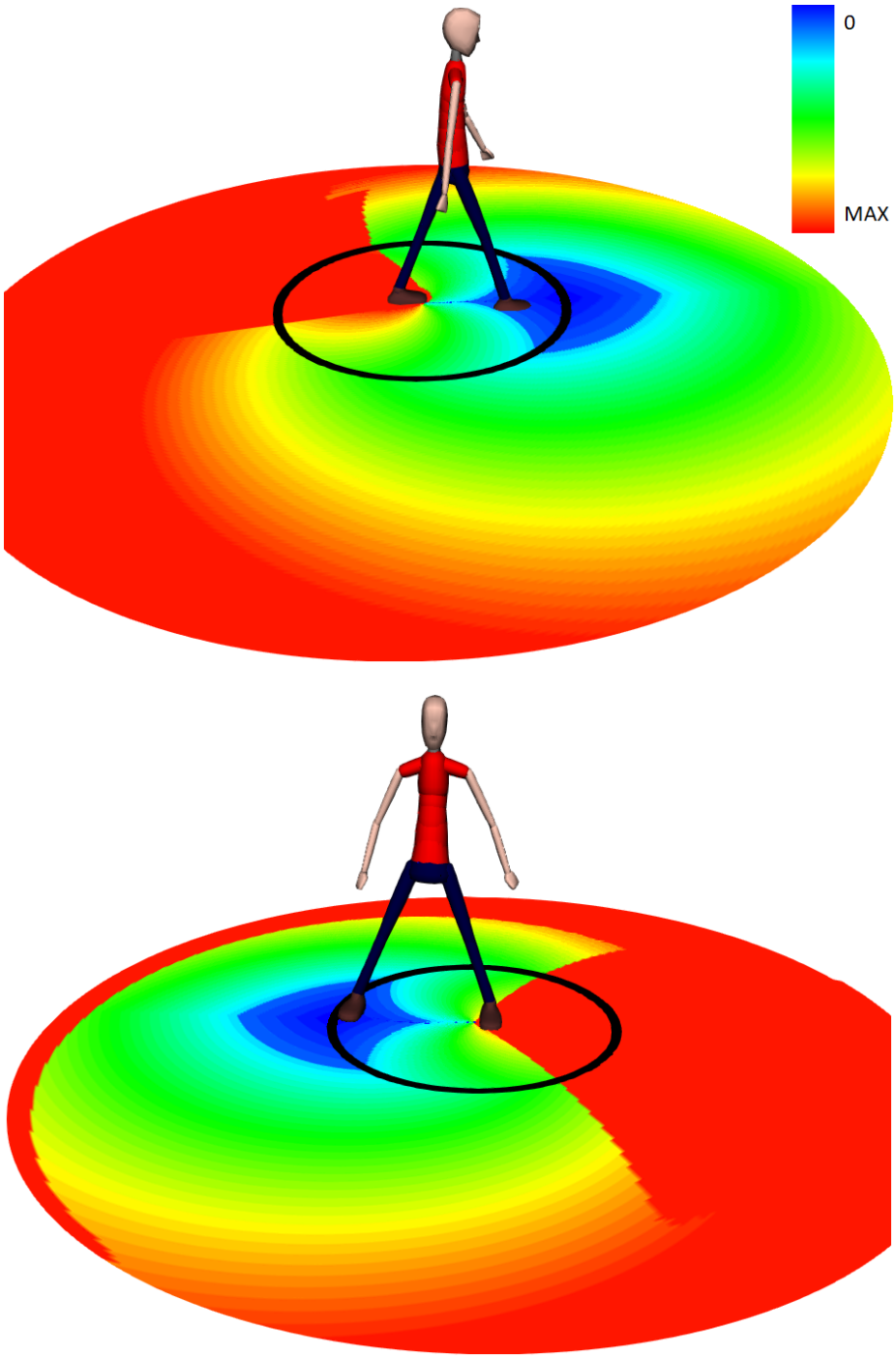

FIGURE 4.1. Parameterization coverage maps for the frontal regular walking (top) and lateral walking (bottom) locomotion behaviors. The maps are on polar coordinates. The angle coordinate represents motion deformation controlling the direction of the motion. The radial coordinate represents the orientation deformation that is applied to specify the final orientation of the character's body, at the end of one step cycle. The black boundary delimits the radius representing no deformation on the final character orientation.

achieving a walking with arms collision avoidance behavior. Additional navigation behaviors or styles can be easily added to our framework.

Each behavior is then associated with its clearance requirements. Let c_1 be the minimal path clearance that a character behavior can handle. In our case this corresponds to behavior $\mathcal{B}_1$ and its required clearance is equivalent to the radius of a cylinder enclosing only the body of the character in a straight posture. While this clearance is enough for passing narrow passages with confidence, we want to avoid using it as much as possible. Clearance value c_2 is the clearance required for $\mathcal{B}_2$, which executes frontal walking with arms collision avoidance. Finally, c_3 is the clearance required for the preferred unconstrained regular frontal walking behavior. Throughout this chapter we may refer to it as the normal behavior. The clearance values are expected to satisfy $c_3 > c_2 > c_1$, which implies from the assumption that the behaviors are ordered with respect to their capabilities of handling narrow passages. This order is also used as the inverse of the behavior preference order.

Algorithms 2 and 4 summarize the main steps of the overall behavioral path planner. They rely on several calls to the LCT path planner $\textbf{FindPath}(c; \boldsymbol{a}, \boldsymbol{b})$, which returns a free path π of clearance c between points $\boldsymbol{a}$ and $\boldsymbol{b}$. Paths are computed according to the needed clearance of each behavior and composed such that the solution path is defined as a sequence of path sections $\Pi = \{\pi_1, \ldots, \pi_k\}$, where each section π_i has an independent clearance value.

The overall algorithm is designed to minimize path length and at the same time maximize the employment of the preferred behaviors. These two objectives are contradictory because the preferred behaviors require more clearance than the non-preferred ones, and paths with high clearance will always be of longer or equal length than ones with low clearance.

Algorithm 2 starts by computing a path with the minimum accepted clearance c_1. The path is shortened according to the planning horizon h, and then Algorithm 4 is iteratively called in order to evaluate and consider longer path sections that can be executed with preferred behaviors, until the entire path is processed.

Algorithm 2 is designed to operate in two modes: interactive character control, in which case path sections are sent for execution as they are computed, or full path processing, where the entire multi-behavior path is composed and returned, if one exists.

Algorithm 3 relies on the following 3 additional procedures:

- $\textbf{TestCollision}(\Pi, \boldsymbol{p}, \mathcal{B})$ starts testing for collision points from point $\boldsymbol{p}$ onwards, along path Π, and employing behavior $\mathcal{B}$. The function returns the first collision point encountered or the goal point of the path in case there is none.
- $\textbf{Furthest}(\Pi, \boldsymbol{p}, \mathcal{B})$ first calls $\textbf{TestCollision}(\Pi, \boldsymbol{p}, \mathcal{B})$ to find the first collision point. The function will find the last step along the path such that there are no collisions when following with behavior $\mathcal{B}$.
- $\textbf{Closest}(\Pi, \boldsymbol{p}, \mathcal{B})$ finds the closest point $\boldsymbol{r}$ in the path Π beyond $\boldsymbol{p}$ such that $\pi(c_i; \boldsymbol{q}, \boldsymbol{r})$ is a feasible subpath of Π and the character can continue with the normal behavior $\mathcal{B}_n$ afterwards. The function is called in a moment in which we are sure that a behavior different than the preferred one has to be used,

Algorithm 2 - Behavioral Path Planner Entry Point

Input: current position $\boldsymbol{s}$, goal position $\boldsymbol{g}$, and the number of available behaviors n (3 in our current work.)

Output: path from $\boldsymbol{s}$ to $\boldsymbol{g}$ with annotated behavior, or null path if a feasible path does not exist.

```
1: procedure BhPathMain( s, g, n )
2:     Π_cur = FindPath(c_1; s, g);
3:     if ( Π_cur == null ) return null path;
4:     if ( length(Π_cur) > h ) Π_cur is shortened to length h;
5:     g_h = final point of Π_cur;
6:     Π_sol = {s};
7:     ŝ = s;
8:     while ( character not at g_h ) do
9:         Append BhPath(ŝ, g_h, Π_cur, n) to Π_sol
10:        if ( interactive mode ) then
11:            Update Π_sol on the character execution queue;
12:        ŝ = last point on Π_sol
13:        Π_cur = Π_sol
14:    return Π_sol;
```

so any attempted behavior switch can cause collisions and we are interested in detecting the first time that this does not happen.

The procedure greedily explores the possible behaviors that can navigate the path. It begins by evaluating the path traversal until a collision is found. We want to avoid this collision, but at the same time, we do not want to vary too much from the original path. To do this, the system adds a 2D point-obstacle to the navigation mesh environment so that when a new path is found, this point is avoided, and thus the previous collision is avoided. The point-obstacle location is based on the collision point projected on the floor and the root position of the character. We cannot choose the collision point as a new point-obstacle, because it lies inside one already. If we use the root position the path may change too much, specially in narrow passages. We therefore compute the point-obstacle as a point between these two places. An example of this path modification procedure is shown in Figure 4.2.

When a collision cannot be prevented by path modification, the algorithm will eventually find a path that exceeds the length threshold l_t, at which point it will start to test different behaviors to overcome the existing obstacle. Once a behavior is found, it annotates that section of the path and iterates back to the frontal walking behavior in order to continue the behavior search.

An example of a solution path obtained is shown in Figure 4.3. In this image, the first time **Furthest** is called, the method returns the last point along the first blue trajectory. Then it switches to $\mathcal{B}_2$ and calls **Closest**, it then returns the last point along the green trajectory on that path and the green path section will be annotated with its corresponding behavior.

Algorithm 3 - Behavioral Path Planner Main Iterations

Input: initial position $\hat{\boldsymbol{s}}$, goal position $\boldsymbol{g}_h$, current path Π_{cur} and the number of available behaviors n (3 in our current work.)
Output: path section starting at $\hat{\boldsymbol{s}}$ with annotated behavior.

procedure BHPATH($\hat{\boldsymbol{s}}, \boldsymbol{g}_h, \Pi_{cur}, n$)
  // Evaluate if preferred behavior is feasible:
  $\boldsymbol{p}$ = TESTCOLLISION($\Pi_{cur}, \hat{\boldsymbol{s}}, \mathcal{B}_n$);
  if ($\boldsymbol{p}$ == *null*) **return;** *// Done.*
  $\boldsymbol{p}_{obs}$ = point between $\boldsymbol{p}$ and character;
  Insert $\boldsymbol{p}_{obs}$ as point-obstacle in LCT;
  Π_{new} = FINDPATH($c_1; \hat{\boldsymbol{s}}, \boldsymbol{g}_h$);
  if ($|length(\Pi_{new}) - length(\Pi_{cur})| > l_t$) **then**
    // Π_{new} is too long
    // Find the preferred navigation behaviors of Π_{cur}:
    Remove $\boldsymbol{p}_{obs}$ from LCT;
    $\boldsymbol{q}$ = FURTHEST($\Pi_{cur}, \hat{\boldsymbol{s}}, \mathcal{B}_n$);
    for ($i = n - 1$ to 1) **do**
      $\boldsymbol{r}$ = CLOSEST($\Pi_{cur}, \boldsymbol{q}, \mathcal{B}_i$);
      if ($\boldsymbol{r}$ exists) **then**
        Append FINDPATH($c_n; \hat{\boldsymbol{s}}, \boldsymbol{q}$) to Π_{sol};
        Append FINDPATH($c_i; \boldsymbol{q}, \boldsymbol{r}$) to Π_{sol};
        break; *// exit for loop*
  else
    Append Π_{new} to Π_{sol};
  return Π_{sol};

After successful completion the overall algorithm will return a path

$$\Pi = \{\pi_1(c_{j_1}; \boldsymbol{a}_1, \boldsymbol{b}_1), \ldots, \pi_n(c_{j_n}; \boldsymbol{a}_n, \boldsymbol{b}_n)\},$$

where $\boldsymbol{b}_i = \boldsymbol{a}_{i+1}$, $i \in \{1, ..., n-1\}$. The path sequence implicitly defines a solution path with behavior transitions in the following way: if $j_i = 3$ regular walking is used for that section; if $j_i = 2$, constrained walking is used; otherwise lateral walking is used.

In summary, the overall procedure will follow two directions, depending on the feasibility of the current path. One will try to find the best path by adding point obstacles and the other will switch to a different behavior before trying to add more obstacles. We can see an overview of our system on Figure 4.4.

4.5.1. Results. This proposed method was tested on three different scenarios and we have designed an interactive application that computes at interactive rates path following to arbitrarily given points in a given scenario.

4.5.1.1. ***Scenarios.*** In the first scenario the character has to find a path that crosses an open door with obstacles on the way. The door has a geometry that makes it impossible to be crossed without using a different behavior than the preferred

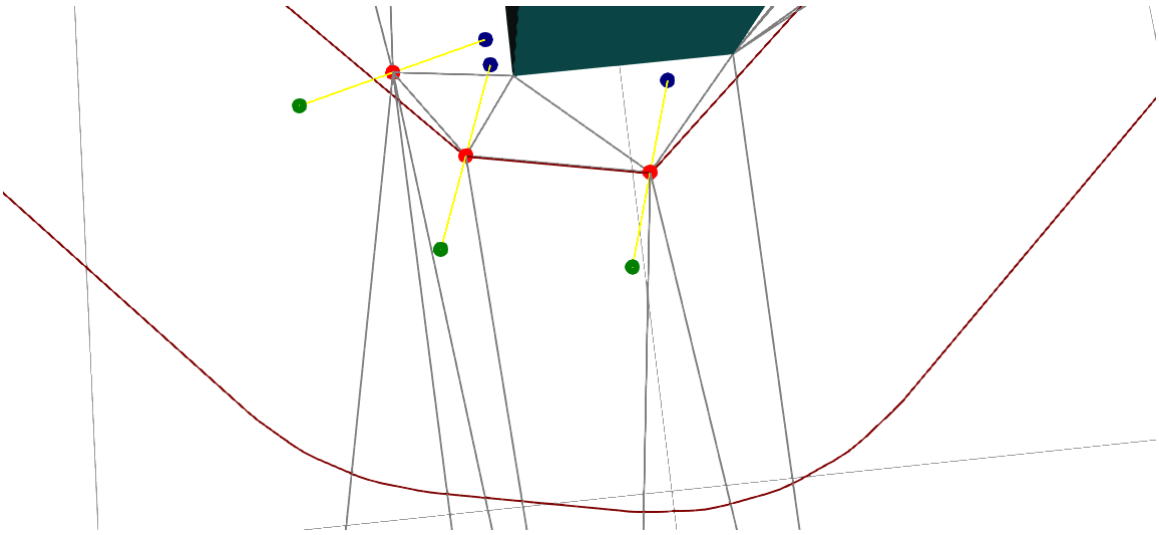

FIGURE 4.2. The blue positions represent the projected joint position of the limbs that collided with obstacles. The green positions are the projected character root positions on the floor at the moment of each collision. The red positions are the point-obstacles that were inserted in the LCT navigation mesh as additional constraints to be considered in subsequent path queries. The final path (marked by the dark red corridor) takes into account the new point-obstacles that were inserted.

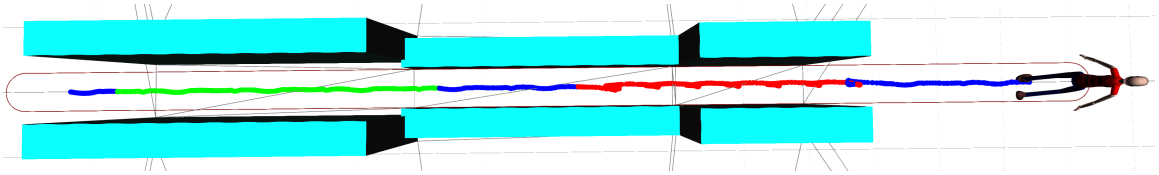

FIGURE 4.3. Example solution path found by our method. The blue path sections can be executed with the preferred behavior $\mathcal{B}_3$. The green section with $\mathcal{B}_2$ and the red sections with $\mathcal{B}_1$. The obstacles are colored cyan.

regular walking. The first area is cleared by adding point-obstacles to modify the path. However, the door obstacle could not be cleared by path modifications and so a behavior switch to the arm avoidance behavior was selected, which was sufficient to avoid collisions with the door. This example is illustrated in Figure 4.5-right.

The second scenario was created to intentionally trigger our specific behaviors. It consists of a narrow corridor with specific areas designed to be traversed only by each of the available behaviors. The first area of the corridor forces an arm avoidance behavior. The next one has less clearance for the path planner, but because the obstacles don't collide with the arms, the system allows a switch to the normal behavior. Finally, the final section has the minimum accepted clearance and lateral side-stepping is required to be employed. Figure 4.6 shows the environment and the obtained results.

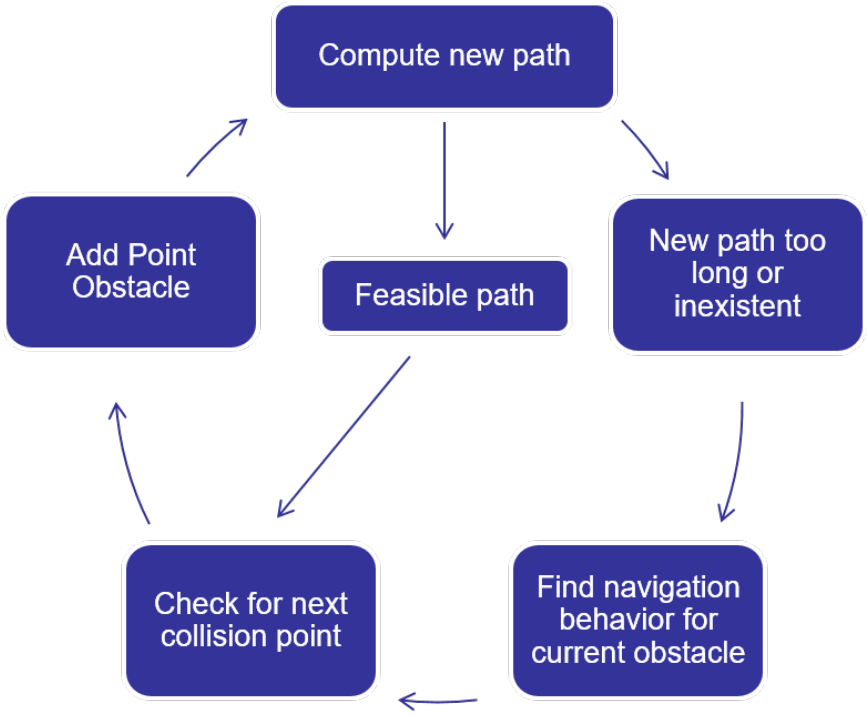

FIGURE 4.4. System overview

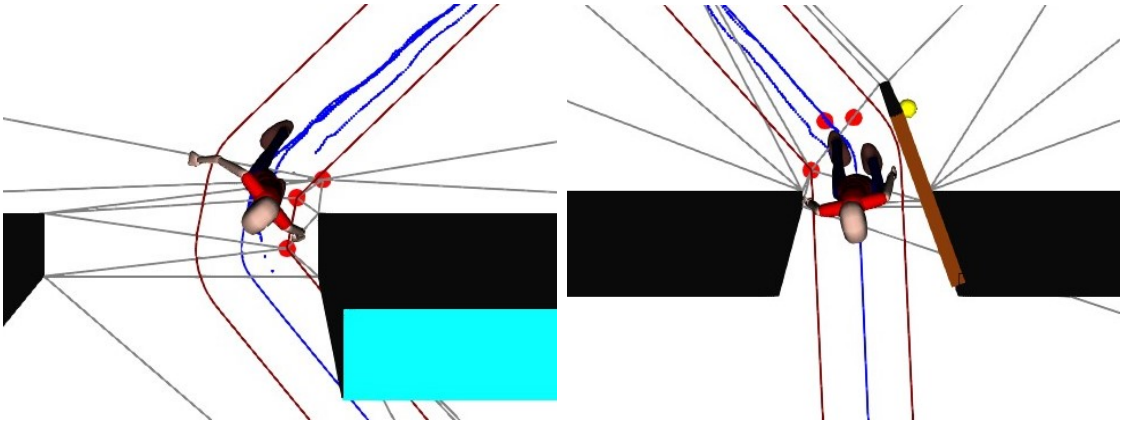

FIGURE 4.5. Left: The obstacle was avoided by adding extra point obstacles to steer away the path. Right: The corridor is narrow so an arm avoidance behavior was needed.

Finally we have implemented a randomized environment with obstacles of varied dimensions (Figure 4.7). We can control the density of the obstacles in the scene, going from easy to traverse with the normal behavior to highly cluttered and often requiring non-preferred behaviors. We use this environment for performance evaluation of our method, as described in the next sub-section.

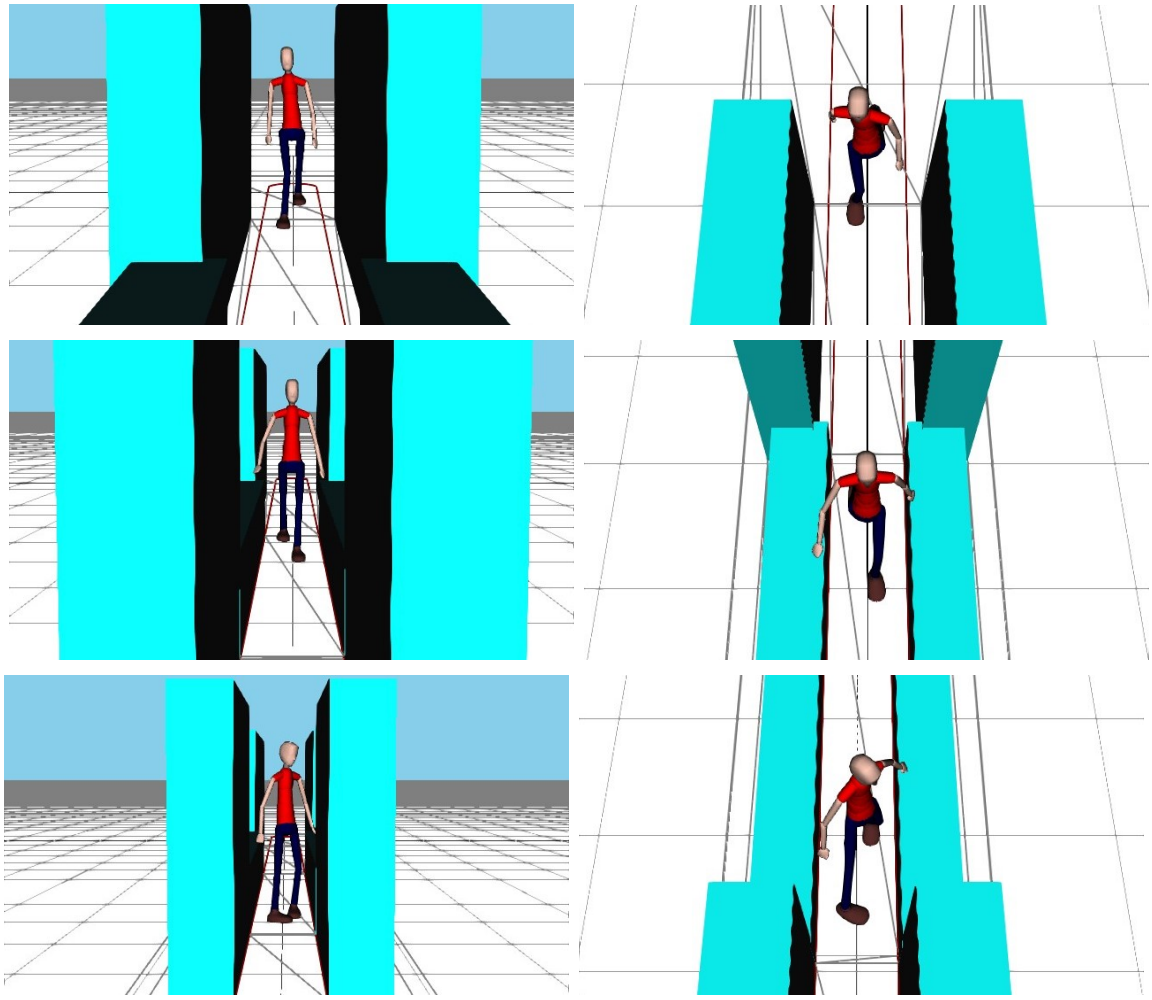

FIGURE 4.6. Behavior transition in a narrow corridor. From top to bottom: arm avoidance behavior employed in the first section. In the middle section regular walking could be used because no collisions were detected even though the path has low clearance. In the final section lateral side-stepping is required.

4.5.2. Performance Evaluation. We evaluate the performance of our planner on two different randomized environments, one with a reasonable density of obstacles considered to be typical of *normal* cases, and the other is a denser environment situation. Figure 4.7 illustrates the general appearance of these two environments.

In order to test our environments, we defined 100 different goals at the same linear distance from the starting point of the path queries. The length of the planned paths was equivalent to 6 and 10 character steps in the normal behavior. We then ran our behavioral path planner in both environments and measured the computation time taken to reach the solutions.

Our results are presented in Figure 4.8. The blue bars represent the average time of each tested type of environment, and the black line is the standard deviation of the

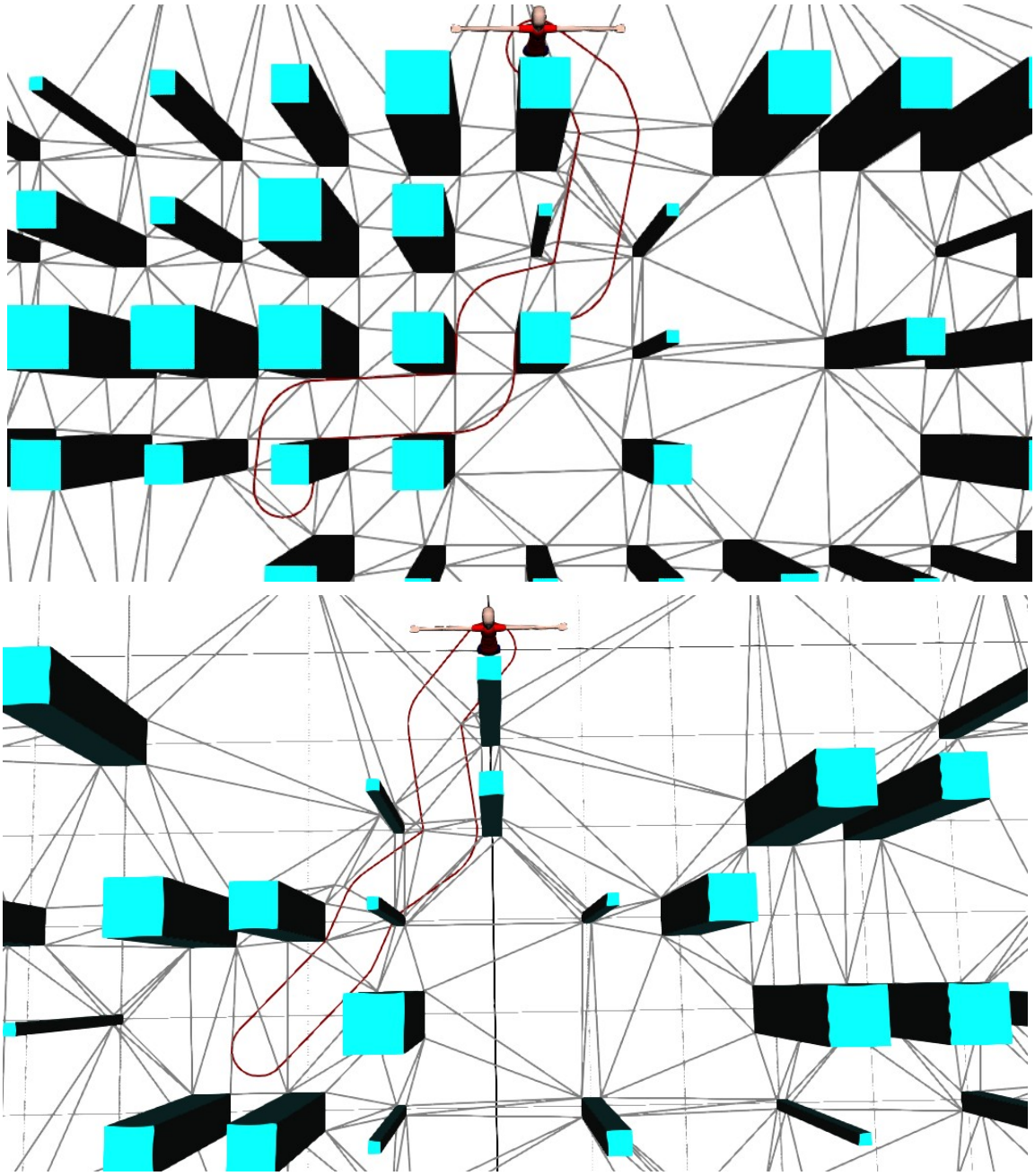

FIGURE 4.7. Top: dense environment. Bottom: normal environment. Computed solution corridors are shown in both environments for a same goal point.

results. Each bar is associated with a number and a label specifying the environment type. The number represents the linear distance between the used starting point and the goal point, and the label represents the corresponding environment. The dense environment contained several regions not allowing a direct path, generating

longer paths and requiring more computation time. Because of this reason there were some high values observed and the variance in the first evaluation in the graph is considerably higher than in the other tests.

The results show that a possible limitation on the planning horizon can generate a real-time interactive path planner, as presented on the following subsection.

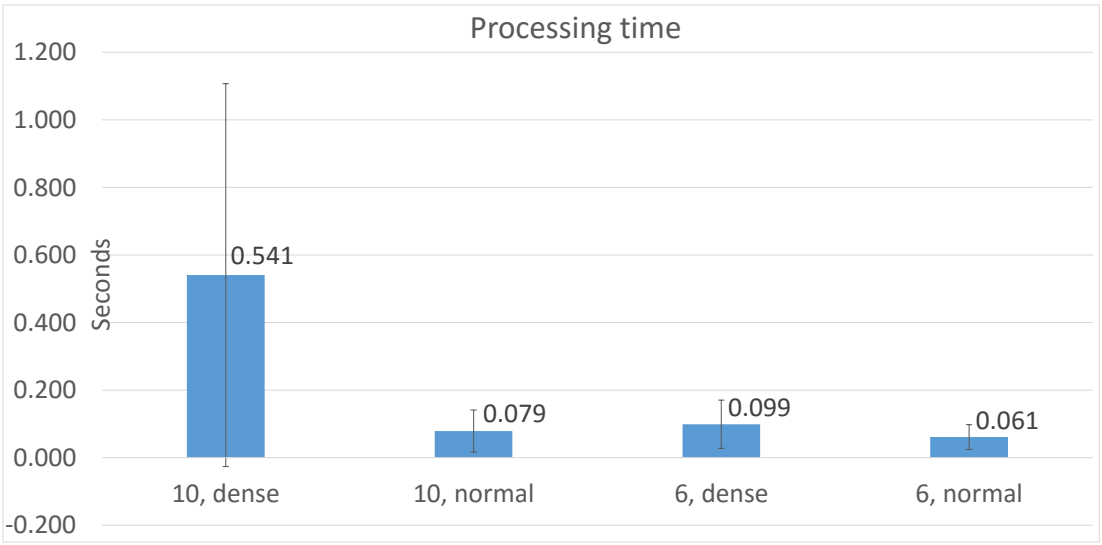

FIGURE 4.8. Performance evaluation. The graph presents the standard deviation and average times required to compute paths of different lengths on dense and normal environments.

4.5.3. Interactive Path Following. In our interactive test application every time the user selects a reachable point in the environment, the system computes the corresponding annotated path from the location at the end of the current motion clip to the newly specified target.

As our evaluation shows, if we limit the horizon distance h of the path that the algorithm is processing it is possible to run the system interactively regardless of the complexity of the environment. We are thus able to reach interactive results suitable for applications that require a mouse-guided target selection for locomotion, as is the case in several computer games.

4.5.4. Discussion and Extensions. This behavioral path planner, although fast under a limited horizon, could be further improved by implementing the path search to occur during motion synthesis, allowing later portions of the path to be upgraded to preferred behaviors while the initial path portion is executed. Our current solution is based on a computation time limit, with the current path being returned when the given time threshold is reached.

Clearly a bottleneck of our approach is the full body collision detection with the environment at the mesh level. Currently we test full collision detection at every frame of the produced motions. Optimizations at the collision detection level are possible and would significantly improve the overall performance of the planner.

It is in principle possible to only rely on clearance parameters in order to avoid inserting point-obstacles in the navigation mesh; however, additional techniques would be needed. The LCT navigation mesh has been recently extended with the concept of extra clearance Kallmann [2016], which could be used in such an approach. However, such approach is not designed to explore new corridors, it is only capable of labeling the overall clearance among the sections of the computed path. Our approach of inserting point-obstacles provides a way to solve this problem.

Our system can be easily adapted to operate with a larger set of behaviors, in particular with reactive behaviors. We have a behavior-associated measure related directly to the behavior selection mechanism, but this can be adapted to assign a priority to each behavior in order to introduce behaviors with dynamic clearances.

4.6. Behavioral Layered Path Planner

Since in the proposed approach collision detection is reduced to layered clearance tests, the locomotion behaviors taken into account are analyzed in order to determine their clearance requirements. For each frame of each motion we analyze the required clearances at several layers parallel to the ground plane. Clearances are computed as the minimum diameter completely enclosing the character in the direction orthogonal to the motion direction. A few representative layers are then selected for use during the path planning stage.

For the considered behaviors we have noticed that two layers are enough to capture the most important clearance requirements. One layer represents the clearance requirements of the legs and the other takes into account the movement of the arms. These layers are referred to as the legs and torso layers. Their respective clearances are illustrated in Figure 4.9.

A cost function will be later associated to each behavior in order to quantify the "preference" of using a behavior.

It is possible to observe that the required leg clearance is the same in all our behaviors while the torso clearance is higher in the preferred behaviors, such that $c_i^1 = c_j^1$, $\forall i, j$, and $c_i^2 < c_j^2$ when $i < j$, $i, j \in \{1, \ldots, n\}$. Our set of behaviors is therefore also ordered by increasing torso clearance requirements.

4.6.1. Layers. Our path planner relies on queries performed on navigation meshes built for each of the layers selected to capture the clearances of the locomotion behaviors. With our current set of behaviors we only need two layers to represent our environments.

The first layer (Figure 4.10-bottom) contains a navigation mesh where its 2D obstacles are the projections of all of the 3D obstacles in the environment to the ground. This layer captures all the passages of the environment and will be used to perform path queries.

The second layer (Figure 4.10-top) is used to validate, label and select alternative behaviors that can be used to execute paths with different clearances. This layer only contains as 2D obstacles the polygonal sections obtained by the intersection of the layer plane with the 3D obstacles. It therefore only represents the obstacles that can generate a conflict with the torso of our character.

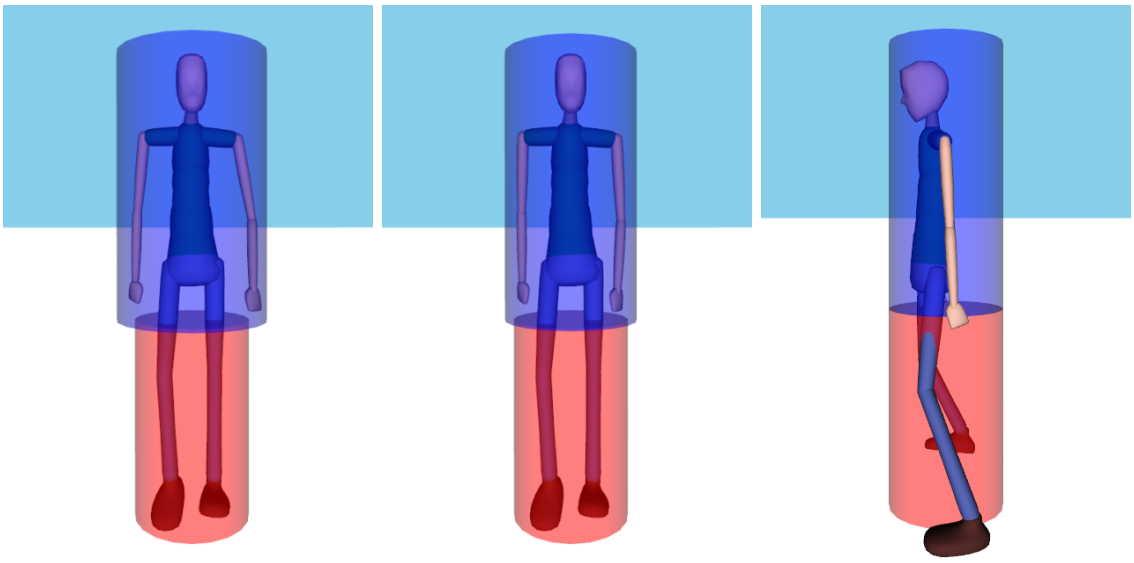

FIGURE 4.9. Left: the frontal walking behavior requires the torso clearance larger than the bottom clearance. Center: with arms constrained frontal walking requires only slightly more torso clearance than legs clearance. Right: lateral walking requires the least clearance and the torso and legs clearances are the same because arm movement occurs only along the motion direction.

Additional layers can be incorporated as needed. The overall approach is to first compute low-clearance paths on the lower layer capturing all obstacles and then to check for additional clearance on the other layers.

4.6.2. Path Planner. Our planner computes a free path from the current position of the character to a given goal position, and returns it as a sequence of concatenated sub-paths, each to be executed with a specific locomotion behavior. A null path is returned if no path exists. The overall solution is therefore a sequence of sub-path sections $\Pi = \{\pi_1, \ldots, \pi_k\}$, where each section π_i satisfies the needed clearance value for its specific behavior.

The possibility of using different behaviors leads to several possible solutions. A cost value can be associated to each behavior such that the overall cost of a solution can be defined as:

$$\textbf{COST}(\Pi) = \sum_k ||\pi_k|| \cdot cost_k\left(\mathcal{B}_k\right),$$

where $||\pi_k||$ denotes the length of the k^{th} sub-path, and $cost_k(\mathcal{B}_k)$ returns the cost of locomotion per unit length using behavior $\mathcal{B}_k$. While the goal of our planner is to find a solution that minimizes this overall cost function, our current approach is heuristic-based and does not attempt to guarantee a global optimum.

We rely on the local clearance triangulation (LCT) navigation mesh Kallmann [2014], which supports computation of paths with arbitrary clearance, clearance checks at any position, and dynamic insertion and removal of degenerate obstacles defined as

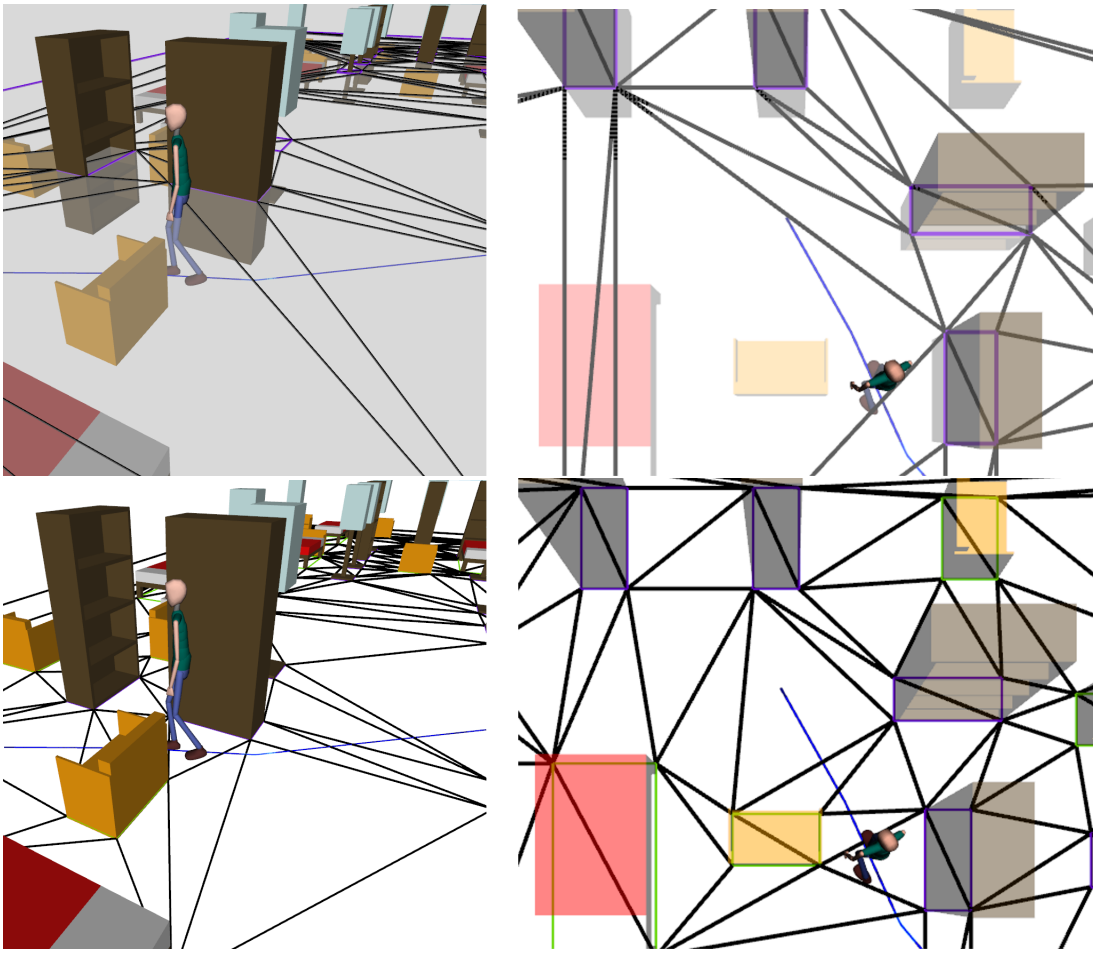

FIGURE 4.10. Two layers are used in this method. The bottom layer (bottom images) represents all obstacles while the top layer (top images) only represents the tall obstacles which are mostly important to detect torso collisions.

single points. This last feature is used to add constraints that block passages in the navigation mesh, forcing the path planner to search for alternative paths with new requirements in search for lower-cost solutions.

Algorithm 4 summarizes the main steps of the layered path planner. It starts by finding an initial annotated path in the bottom layer using the smallest clearance c_1^1, which is obtained with a call to function **FINDPATH**$(c_1^1, \boldsymbol{s}, \boldsymbol{g})$. Without additional tests this path can only be safely executed using our least preferred behavior $\mathcal{B}_1$, which is the behavior of highest cost. The algorithm will then seek to lower the overall cost of this initial solution.

Function **CONFLICTSECTIONS**() returns all points $\{\boldsymbol{p}_{obs}\}$, spaced by the minimum behavior concatenation length along the current annotated path Π_{cur}, where the path only has enough clearance for $\mathcal{B}_1$ to be executed. These points represent passages that are too narrow for other behaviors to be executed. In order to evaluate alternative

solutions, a point subset $\{\hat{\boldsymbol{p}}_{obs}\} \subset \{\boldsymbol{p}_{obs}\}$ is selected to be inserted as point-obstacles, forcing the planner to search for a different path, which may have lower overall cost. While considering all sub-sets would lead to the evaluation of all alternative paths, too many combinations would be generated and we limit our search to every single point instead of every possible subset.

Overall, Algorithm 4 greedily explores the possible behaviors that can navigate the path, using **Conflict Sections**() to detect the narrowest path sections that will likely generate collisions if the side-stepping behavior is not used. It then evaluates modifying the path in order to be able to use preferred behaviors and to lower the overall cost of the current solution. To force new paths to be evaluated, Algorithm 5 adds 2D point-obstacles to the bottom layer navigation mesh in order to block narrow passages and force new paths to be found and evaluated.

If using the least desired behavior cannot be prevented by path modification, the algorithm will eventually find a path that exceeds the cost of the current one, at which point it will finish its execution and return the current solution.

When Algorithm 4 succesfully terminates it returns a path annotated with behaviors to be executed. The annotation procedure is called for every path evaluated and it is described in Algorithm 5.

Algorithm 5 relies on **TestClearance**$(\Pi, \boldsymbol{p}, \mathcal{B})$ which verifies the possibility from point $\boldsymbol{p}$ onwards, along path Π, to employ behavior $\mathcal{B}$. The function returns the first point that does not allow using $\mathcal{B}$ due clearance requirements. This procedure relies on clearance tests performed on the upper layers as required by the requested behavior. The upper layer doesn't contain short obstacles because they won't cause collisions with the torso or arms, therefore the upper layer has more clearance in certain passages than the bottom layer. Clearance tests are efficiently performed because the navigation mesh provides an exact triangular decomposition of the free space in the layer. This allows the quick verification if a desired clearance value around a point in the path remains covered by free triangles, which is executed by only locally checking the triangles around the query point.

An example of a solution path obtained is shown in Figure 4.11. While the available leg clearance is the same inside the corridor, due to the difference in height of the obstacles, different preferred behaviors with larger required clearance are selected along the path. The solution path in this example has the minimum possible cost.

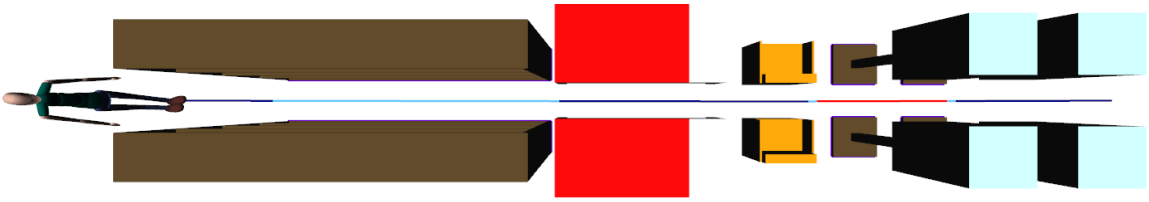

Figure 4.11. Example solution path found by our method. The blue path sections can be executed with $\mathcal{B}_3$, the cyan sections with $\mathcal{B}_2$, and the red sections with $\mathcal{B}_1$.

Algorithm 4 - Behavioral Layered Path Planner

Input: initial position $\boldsymbol{s}$, goal position $\boldsymbol{g}$ and the number of available behaviors n (3 in our current work.)
Output: path from $\boldsymbol{s}$ to $\boldsymbol{g}$ with annotated behavior, or null path if a feasible path does not exist.

1: **procedure** **BhPath**$(\boldsymbol{s}, \boldsymbol{g}, n)$
2: $cost_{old} = \infty$;
3: $\Pi_{cur} =$ **FindPath**$(c_1^1; \boldsymbol{s}, \boldsymbol{g})$;
4: $\Pi_{cur} =$ **AnnotatePath**$(\Pi_{cur}, \boldsymbol{s}, \boldsymbol{g})$;
5: $cost_{min} =$ **Cost** (Π_{curr})
6: **while** ($cost_{min} < cost_{old}$) **do**
7: $cost_{old} = cost_{min}$;
8: $\{\boldsymbol{p}_{obs}\} =$ **ConflictSections**(Π_{cur});
9: $\Pi_{min} = \emptyset$;
10: **for each** $\{\hat{\boldsymbol{p}}_{obs}\} \subset \{\boldsymbol{p}_{obs}\}$ **do**
11: Add $\{\hat{\boldsymbol{p}}_{obs}\}$ as a point obstacle in first layer;
12: $\Pi_{new} =$ **FindPath**$(c_1^1; \boldsymbol{s}, \boldsymbol{g})$;
13: $\Pi_{new} =$ **AnnotatePath**$(\Pi_{new}, \boldsymbol{s}, \boldsymbol{g})$;
14: **if** (**Cost**$(\Pi_{new}) < cost_{min}$) **then**
15: $cost_{min} =$ **Cost**(Π_{new});
16: $\Pi_{min} = \Pi_{new}$;
17: Remove $\{\hat{\boldsymbol{p}}_{obs}\}$ from the environment;
18: **if** ($\Pi_{min} \neq \emptyset$) **then**
19: $\Pi_{cur} = \Pi_{min}$;
20: **return** Π_{curr};

Algorithm 5 - Path Annotation

Input: Path Π, traversing the environment from $\boldsymbol{s}$ to $\boldsymbol{g}$
Output: Annotated path.

1: **procedure** **AnnotatePath**$(\Pi, \boldsymbol{s}, \boldsymbol{g})$
2: $\boldsymbol{p} =$ **TestClearance**$(\Pi, \boldsymbol{s}, \mathcal{B}_n)$;
3: $\Pi_{cur} =$ **FindPath**$(c_n; \boldsymbol{s}, \boldsymbol{p})$;
4: **while** ($\boldsymbol{p} \neq \boldsymbol{g}$) **do**
5: **for** $(i = n$ to 1) **do**
6: $\boldsymbol{q} =$ **TestClearance**$(\Pi, \boldsymbol{p}, \mathcal{B}_i)$;
7: **if** ($\boldsymbol{p} \neq \boldsymbol{q}$) **then**
8: Append **FindPath**$(c_i^2; \boldsymbol{p}, \boldsymbol{q})$ to Π_{curr};
9: **break;** *//exit for loop*
10: $\boldsymbol{p} = \boldsymbol{q}$;
11: **return** Π_{curr};

After successful completion the overall algorithm returns a path

$$\Pi = \{\pi_1(c_{j_1}; \boldsymbol{a}_1, \boldsymbol{b}_1), \ldots, \pi_n(c_{j_n}; \boldsymbol{a}_n, \boldsymbol{b}_n)\},$$

where $\boldsymbol{b}_i = \boldsymbol{a}_{i+1}$, $i \in \{1, ..., n-1\}$. The path sequence implicitly defines a solution path with behavior transition in the following way: if $j_i = 3$ regular walking is used for that section; if $j_i = 2$, constrained walking is used; otherwise lateral walking is used.

In summary, the overall procedure starts by finding the minimum clearance path, which can be executed with lateral stepping, however since this behavior is slow and often unnecessary, path section replacements accommodating preferred behaviors are performed whenever the overall path cost can be lowered.

4.6.3. Results. We have evaluated the performance of our method and we demonstrate results obtained in three different scenarios.

4.6.3.1. ***Scenarios***. In the first scenario the character has to find a path that traverses an apartment cluttered with furniture. For example, the door is in a position that makes it impossible to pass without using a different behavior than the preferred regular walking. This example is illustrated in Figure 4.12.

The second scenario is illustrated in Figure 4.13 and was created to intentionally trigger all our behaviors. It consists of a narrow corridor with specific areas designed to be traversed only by each of the available behaviors. The first area of the corridor requires the arms to be constrained. The second area has less clearance but because the obstacles do not collide with the arms the system allows a switch to the regular front walking behavior. The final section has the minimum accepted clearance and lateral side-stepping is required.

The third scenario is a randomized environment with obstacles of varied dimensions (Figure 4.14). We can control the density of the obstacles in the scene, going from easy to traverse with the normal behavior to highly cluttered and often requiring non-preferred behaviors. Every time the user selects a reachable point in the environment, the system computes and concatenates a new annotated path to be executed in order to reach the new point. Smooth and responsive locomotion control is achieved and performed in real-time. We use this environment for performance evaluation of our method, as described next.

4.6.4. Performance Evaluation. We evaluate the performance of our planner on two different randomized environments, one with a reasonable density of obstacles considered to be typical of *normal* cases, and the other is a denser environment situation. Figure 4.14 illustrates the general appearance of these two environments.

In order to test our environments, we defined 100 different goals at the same linear distance from the starting point of the path queries. The length of the planned paths was equivalent to 5, 10 and 20 character steps using the normal behavior. We then ran our layered locomotion planner and compared the results against using full 3D collision detection against the environment geometry as performed in previous work Juarez-Perez and Kallmann [2016a]. The results are presented in Table 4.1 and Figure 4.15.

4.6.5. Discussion. The proposed layered approach clearly overcomes the common bottleneck of collision detection, significantly speeding up computation times and allowing us to plan considerably long paths in real-time.

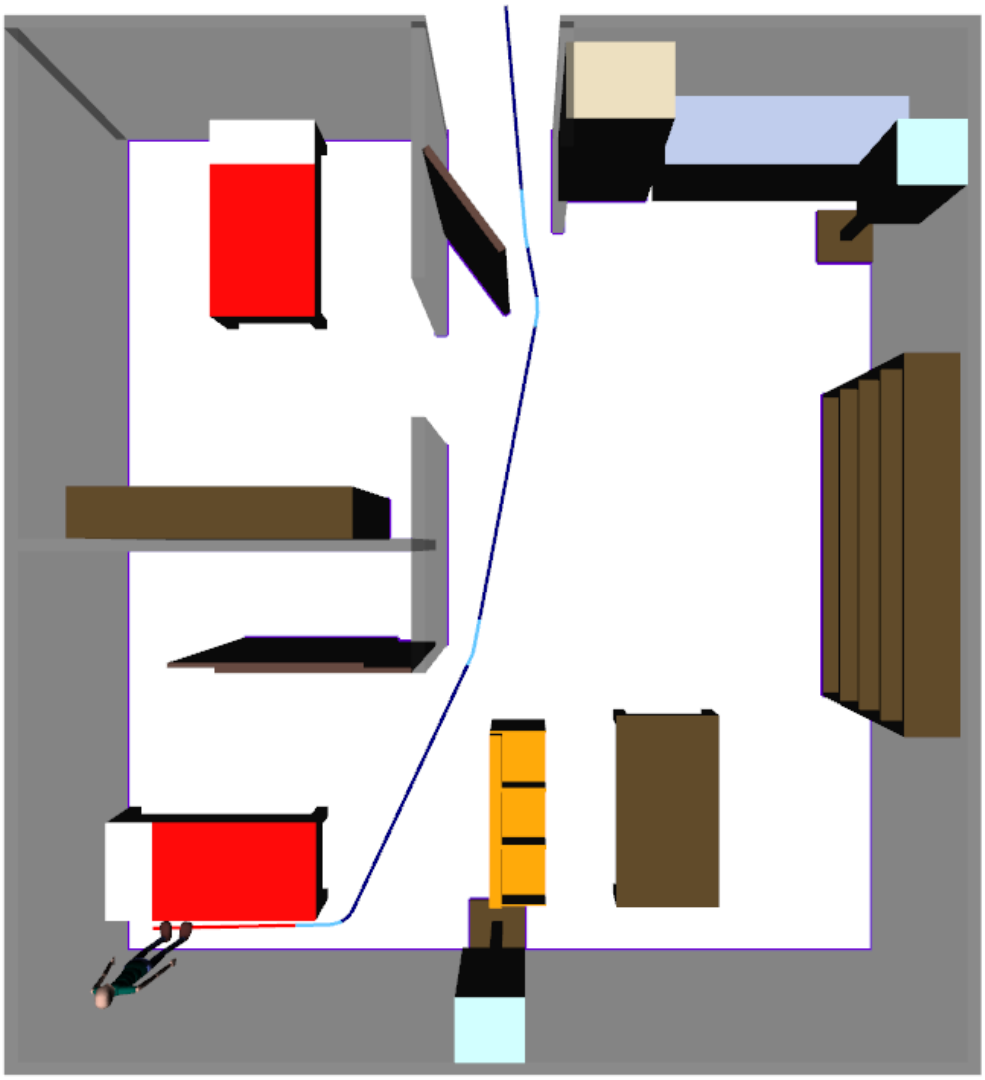

FIGURE 4.12. Apartment scenario.

TABLE 4.1. Average times in milliseconds to compute paths with our method based on layered queries versus using full 3D collision checking.

	Dense environment		Normal environment	
	Layered	3D Collision	Layered	3D Collision
20 Steps	9.70	189.40	6.14	110.65
10 Steps	10.62	105.26	6.48	61.15
5 Steps	9.67	28.21	6.16	22.23

Our system can be easily adapted to operate with a larger set of behaviors and correspondent layers. While currently our approach of projecting all obstacles to the bottom layer cannot capture more complex movements such as crawling under a table, it is certainly possible to design specific uses of layers in order to capture

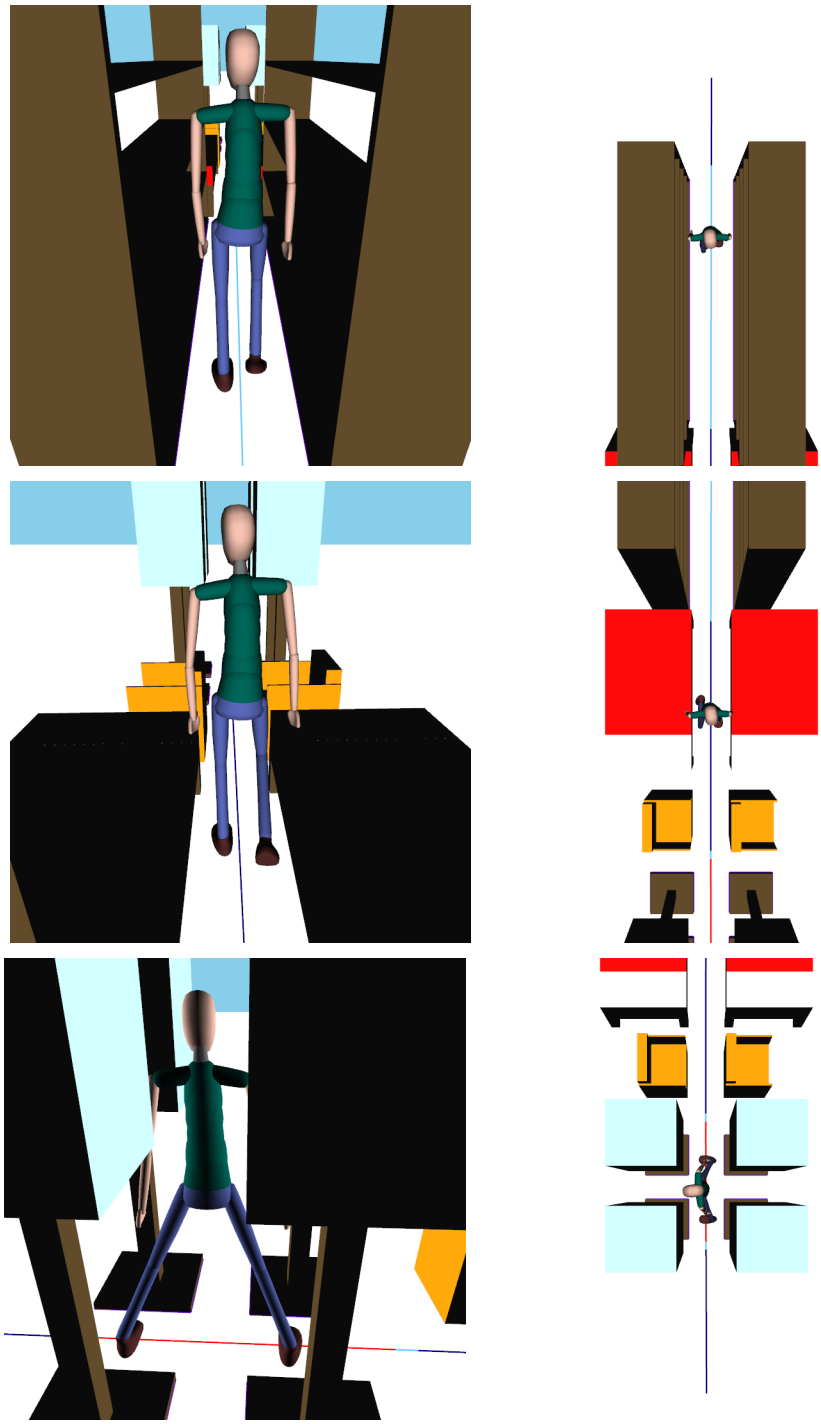

FIGURE 4.13. Narrow corridor behavior selection. From top to bottom: arm avoidance is employed in the first section, regular walking is correctly employed in the middle section given the high clearance at the torso layer, and lateral side-stepping is required in the last, narrowest, section.

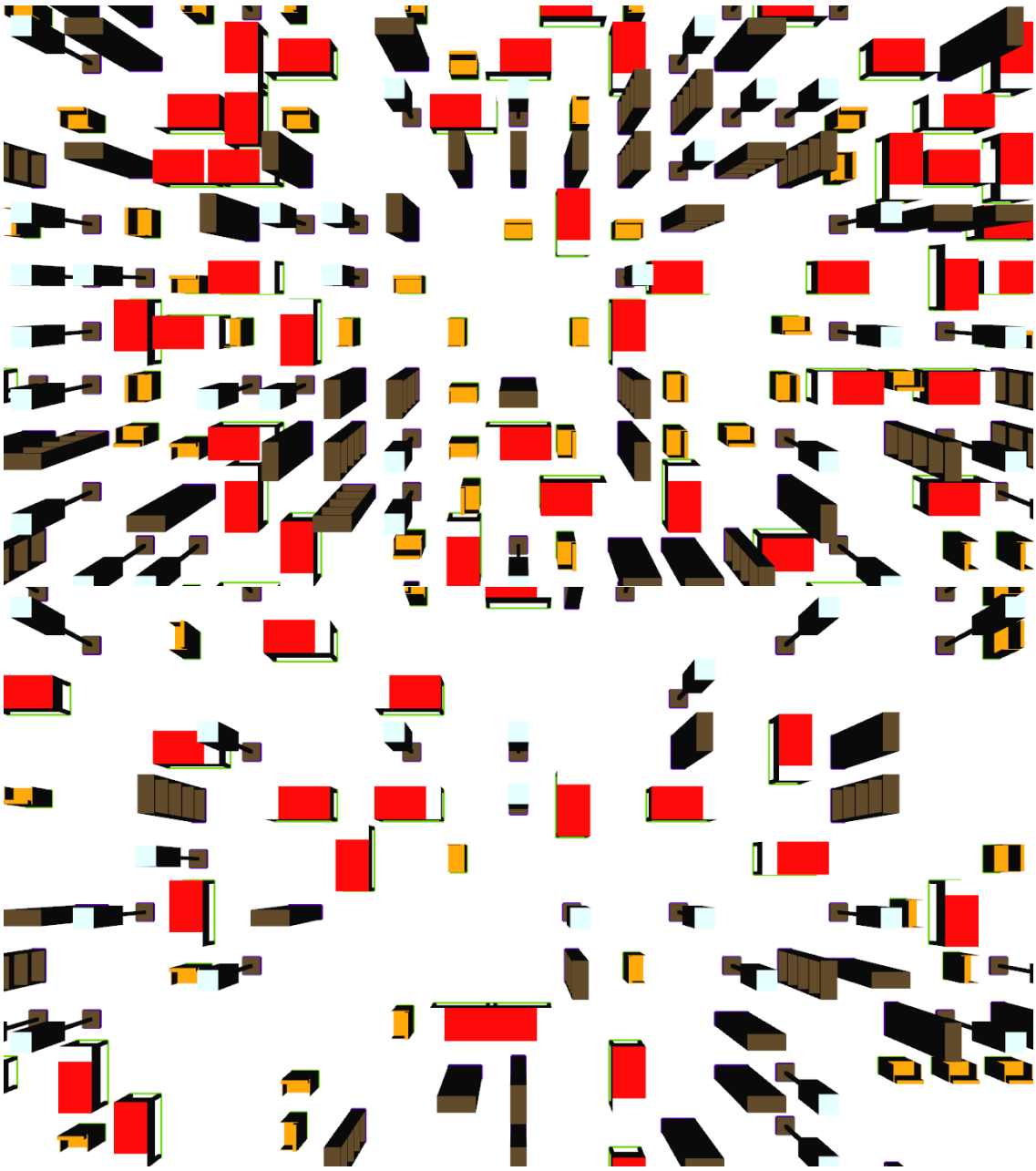

FIGURE 4.14. Dense (top) and Normal (bottom) environments.

such behaviors. As long as the potential behaviors have a way to be ordered, the overall computation should be maintained: we can compute the clearance on the current layer and do a binary search to find the best behavior available for such clearance, this strategy would not increase the overall computational time of the motion planner. Even if our layer mechanism is used for only capturing clearance along horizontal planes, which is appropriate for most locomotion behaviors, applying

layers in generic orientations for other types of movement planning is an interesting future work direction.

Another possible future work direction is to update our method to find globally-optimal solutions. Our current greedy approach based on blocking passages could be extended to evaluate all relevant passages and the underlying graph search could be extended to generate and evaluate all relevant paths Eppstein [1999]. However a combinatorial approach may generate too many candidates given that the global optimum of our problem lies on the Euclidean plane and not on the used adjacency graph representing the free space.

4.7. Discussion

In this section I analyze the reported results and algorithms against competing strategies for behavioral path planning. A summary of these results can be seen on Table 4.2. The analyzed methods are *Precomputed search trees: planning for interactive goal-driven animation* (**PST**)[Lau and Kuffner, 2006], *Deformable Motion: Squeezing into Cluttered Environments*(**DM**) [Choi et al., 2011] and *Precomputed Motion Maps for Unstructured Motion Capture* (**PMM**)[Mahmudi and Kallmann, 2012].

I start by evaluating how each of the strategies perform behavior selection. PMM and PST rely on a precomputed search tree, although they are using different behaviors, their system is incapable of returning a solution that lies outside of their search tree.

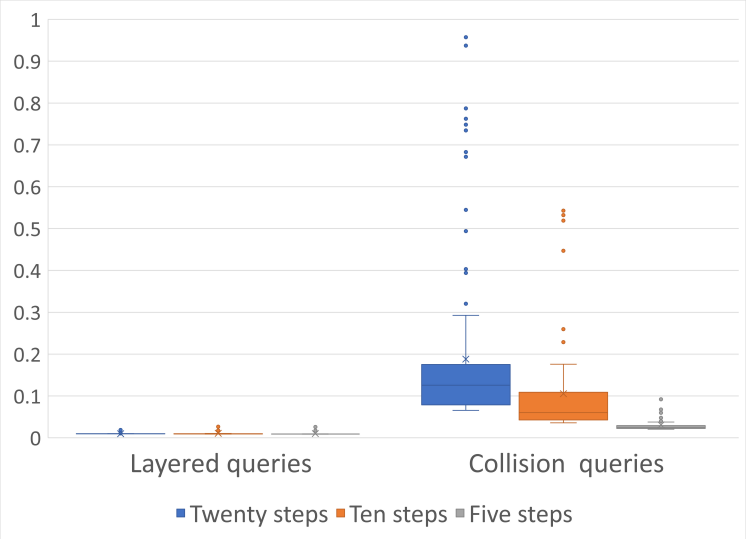

FIGURE 4.15. Average times, deviation and outliers when computing paths of different lengths on a dense environment.

PST and DM strategies require to compute the whole plan if there is any environment change, in contrast this work and PMM can quickly adapt to environment changes.

On challenging environments, DM works only if it is guided by a precomputed Probabilistic Roadmap (PRM) while PST relies on having the solution on the tree after pruning, which is not considered by any of its computations.

Finally, each of the methods had a different way to report the time it required to produce a path. I decided to convert all of the results into planned distance per millisecond of computation. Some of the reported results do not specify the type of environment used or the type of computer that ran them, which limits the reliability of this specific comparison. However, it is clear that some of the results vary by more than 3 orders of magnitude. From this we can conclude that PST and this work are considerable faster than its competitors.

TABLE 4.2. Comparison table with behavioral planning methods.

	Behavior selection	Deformable Paths	Cluttered Environments	Planner Used	Distance computed per ms	Motion Capture Quality
Lau and Kuffner [2006]	**O**	✗	**O**	Bitmap planner	45 m	✓
Choi et al. [2011]	✓	✗	**O**	RRT and PRM	4.5 cm	✗
Mahmudi and Kallmann [2012]	**O**	✓	✓	LCT	9.7 cm	✓
This work	✓	✓	✓	LCT	150 m	✗

4.8. Conclusion

We have presented two different approaches for path planning that incorporates the automatic selection of behaviors of different costs in order to achieve interactive multi-behavior navigation in cluttered environments. These methods compute paths which can be precisely followed by locomotion behaviors that adapt to the given environment in order to achieve collision-free results. The computed solutions adapt to dense environments with narrow passages and we have shown that our system can run at interactive rates producing solutions that represent natural behavioral choices.

Even though the method from section 4.5 shows an overall improvement of section 4.6 and both solve the same problems by design, there could be some specific scenarios in which the former gives a better result. The full body collision detection can allow a path to be followed by a behavior with greater clearance if the obstacle is irregular and allows the upper body to fit through it. Meanwhile the layered approach requires

a simplification of the obstacles and character that could block this path during the planning phase. However in practice, we rarely find this specific scenarios.

Our second method introduces the concept of applying layered clearance tests which eliminate the need for full 3D collision checking, providing a significant speed-up in computation time.

CHAPTER 5

Coordination

The locomotion controller explored in the previous chapter presents the opportunity of characters capable of full scenario exploration. However, the planner is incapable of dealing with moving scenarios or interactive obstacles. A possible approach is to run the planner every time there is a change on the environment, but this introduces a great computational cost and it does not address the problem of interactivity. This chapter explores how a character can interact with mobile obstacles as well as performing actions while on motion.

5.1. Introduction

Achieving autonomous virtual characters able to perform full-body interactions with the environment represents a challenging computational problem. Given the large number of factors which influence the way humans carry out actions Rosenbaum et al. [2013] purely algorithmic approaches are difficult to be developed. At the same time, data-based approaches also face a number of difficulties when generalizing to varied targets, environments, and character dimensions.

As a result we obtain a parametric object interaction controller which is able to synthesize complex environment manipulations while walking, addressing an important class of actions given that mobility is often crucial for supporting object interaction. Our model produces continuous motions with correct velocity profiles, is suitable for real-time applications, and relies on a simple and fast learning phase as pre-computation. The work of this chapter was first introduced on [Juarez-Perez and Kallmann, 2017] and at the time of this dissertation submission, a full submission [Juarez-Perez and Kallmann, 2018a] is being prepared.

5.2. Related Work

A number of strategies have been explored for synthesizing full-body animations of virtual characters interacting with the environment. One important approach is to rely on characters simulated under physics laws. One main advantage of the approach is that physically-valid postures and reactions are naturally produced, and a number of methods have been proposed for achieving full-body controllers able to control characters in physics-based simulations [Faloutsos et al., 2001; Yin et al., 2007; Coros et al., 2010; Liu et al., 2016]. Overall, work in this area has largely focused on locomotion control given that full-body object interaction under physics remains a difficult task to be addressed.

Object interaction under physics has been explored [Liu, 2009; Jain and Liu, 2009,0; Bai et al., 2012]. However, the focus is on body coordination and not on

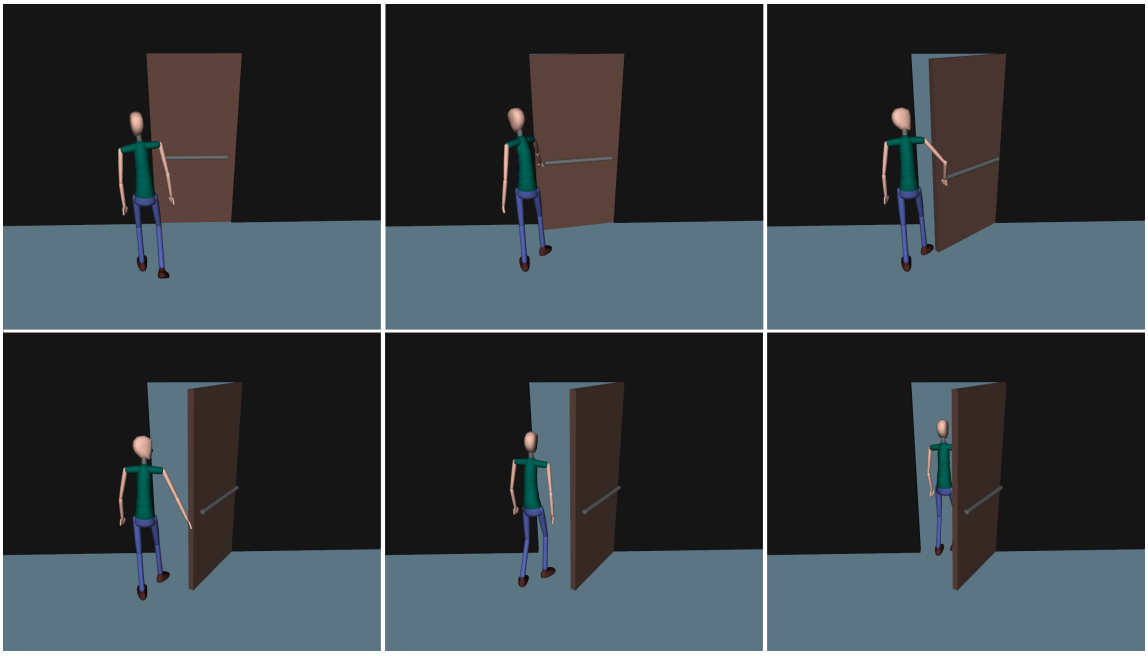

FIGURE 5.1. This sequence of opening and passing through a door synthesizes the upper-body motion in coordination with a generic walking controller. This result demonstrates complex anticipatory spine and arm movements which are key for successfully performing the action without collisions.

coordination with the environment during the interactions. In contrast, our data-based approach focuses on learning spatial relationships with the environment and is able to achieve detailed motions clearly showing full-body anticipation and adaptation to the environment. Our approach can be applied to incorporate these features in physics-based controllers.

In order to address complex environments, data-based approaches often incorporate some sort of planning process [Choi et al., 2002; Yamane et al., 2004; Shapiro et al., 2007; Zhang et al., 2009; Pan et al., 2010; Mahmudi and Kallmann, 2015]. These approaches however have not been extended to address manipulations synchronized with walking, which represents a challenging scenario that is the focus of our work and do not address coordination between locomotion, the upper-body action, and the environment.

It is possible to achieve complex results using motion blending [Rose et al., 1998; Kovar and Gleicher, 2004]. While blending example motions represents a powerful approach for achieving realistic and parameterized motion controllers, each controller has to be carefully implemented from example motions addressing all the needed variations that may be encountered. Furthermore, the number of variations significantly grows for mobile manipulations.

Our approach overcomes these difficulties by relying on a generic walking controller and on a small set of representative example motions, and then on applying a regression

technique that successfully generalizes motion strategies in the example motions. While a pure data-based interpolation approach is heavily depended on the used example motions, our approach is designed to be practical, fast, and to improve the autonomy of controllable autonomous characters.

Another important approach is to employ reinforcement learning methods for achieving data-based motion controllers [Treuille et al., 2007; Levine et al., 2011; Lo and Zwicker, 2008; Shum et al., 2014]. The approach is effective; however, it is time-consuming to learn control policies and controllers have to remain in a low dimension space. Instead of learning a controller, our approach introduces the concept of learning motion coordination features which can be applied to optimize postures in generic settings. In doing so our learning phase is very fast and is not limited to a low dimension control space.

Our coordination features encode spatial relationships between the character and the environment. Spatial relationships have been used in previous work for instance to solve motion retargeting [Al-Asqhar et al., 2013] and here we demonstrate their importance for achieving complex manipulations with the environment. Additional related techniques have been proposed for motion editing [Lee and Kim, 1999; Kim et al., 2009], for deforming motions in order to address a number of constraints including narrow passages [Choi et al., 2011; Kim et al., 2012], and for composing full-body motions from motions computed for body parts [Jang et al., 2008; Heck et al., 2006]. Many of these techniques could enhance our employed posture optimization module; however, in general these approaches do not address the automatic determination of the constraints which are needed for a successful complex interaction with the environment. Local avoidance and reactive methods are basically not sufficient to achieve the spatial and temporal anticipatory movements which are required in the addressed examples.

In summary, our work introduces the concept of learning motion coordination models that can be applied on top of generic walking controllers, leading to a practical and powerful approach able to achieve complex full-body environment interactions for autonomous characters in real-time applications. Our approach automatically detects the important constraints to be addressed, and includes an integrated selection mechanism to trigger the most suitable interaction model to be employed when a character faces diverse actions and situations to be performed in an environment.

5.3. Overview

Our overall method is composed of two phases. First, coordination data is extracted from motion capture examples as an offline phase, and then during the real time phase our coordination model uses the pre-computed data in order to synthesize successful interactions in new situations. Figure 5.2 illustrates the main modules and operations in these two phases.

Our system requires a locomotion controller to be available. First a path is computed passing by the interaction points of interest. Then, when the character is detected to be approaching an interaction, the respective coordination model is

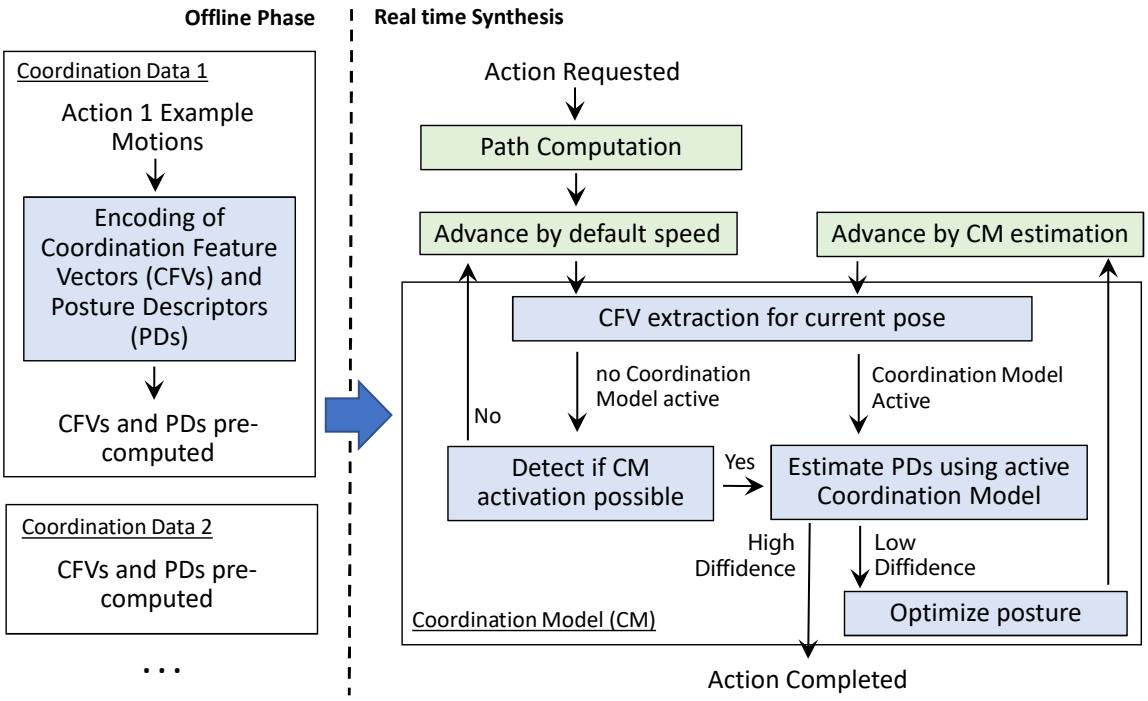

FIGURE 5.2. Overall method overview. Green boxes correspond to locomotion operations and blue boxes correspond to the main modules of the coordination model.

employed to optimize each locomotion posture in order to ensure that the target action can be successfully performed.

During the offline phase Coordination Feature vectors (CFV) and Posture Descriptors (PDs) are extracted from the poses of the example motions. CFVs describe body-environment spatial relationships and PDs encode postural information that is associated to each CFV and that will be later used to optimize postures. Additional higher-level action information is also required, such as hand target positions and information defining if actions are to be considered as multi-phase or single-phase actions. The complete method is described in subsequent sections.

During the online phase the locomotion controller is used to follow the path computed for the given interaction. In most cases a clearance-based path planner is used to generate a path passing near the target interaction points, such that the path conforms to the action requirements.

While the locomotion controller starts to follow the available path, all available coordination models are considered for taking control over the character. When a particular coordination model shows low diffidence with the encountered spatial configuration, it is then activated to optimize the postures of the character and to control the speed of the locomotion in order to perform the target action. Control of the locomotion velocity is essential for achieving the complex coordinations required in our scenarios.

The overall process is depicted in Figure 5.2. Overall, the coordination model is based on performing a regression over learned coordination feature vectors in order to estimate posture descriptors which are then applied to meaningfully synthesize the target interaction.

5.4. Encoding Coordination Features

In order for our method to be successful we need to define Coordination Feature Vectors (CFVs) capable of describing the overall relationships between the nearby environment and the body poses used to perform the given action. Our CFV is based on minimum distances between the environment and a selection of body parts, and it also includes normalized timing information.

CFVs are computed both in the offline phase and in the online phase. During the offline phase CFVs are extracted for every frame of the input example motions, while during the online phase CFVs are computed for the poses of the character being animated.

During the offline phase we are given motion-captured examples of the environment interactions of interest. Each captured motion example M_j consists of a collection of N poses described by joint data: $M_j = \{p_1, \ldots, p_N\}$. Each pose is described by the position and orientation of the skeleton root, plus all the joint values of the remaining degrees of freedom of our character. Each pose p_i is associated with CFV $\boldsymbol{Y}_i$.

5.4.1. Coordination Feature Vector. The CFV is defined as $\boldsymbol{Y}_i = \{\boldsymbol{d}^i, \boldsymbol{\sigma}^i\}$, where $\boldsymbol{d}^i$ encodes environment coordination parameters and $\boldsymbol{\sigma}^i$ encodes time coordination parameters.

The environment coordination parameters $\boldsymbol{d}^i$ encode the vectors of minimum distance from each selected body part to the environment. This requires computing the closest pair of points between each selected body part geometry, posed at pose p_i, and the environment. If the environment is too large only the nearby objects to the action have to be considered, and this can be detect by considering all objects which intersect the bounding box of all poses in M_j.

Since CFVs will also be computed during the online phase it is important to rely on an efficient method. We have developed a fast method based on bounding volume hierarchies (BVHs) of swept sphere volumes Larsen et al. [2000] using the PQP toolkit. We represent the environment and each body part B of interest as independent BVHs. We then transform the BVH of B according to pose p_i and compute the closest pair of points $\boldsymbol{p}_i$ and $\boldsymbol{q}_i$ between B_i and the environment using the respective BVHs. Vector $\boldsymbol{d}_{\min}(B_i) = \boldsymbol{q}_i - \boldsymbol{p}_i$ then encodes the vector of minimum distance between B_i and the environment.

The set of minimum distances for all k body parts of the character in a given pose i, $\{\boldsymbol{d}_{\min}(B_{ik})\}$, is illustrated in Figure 5.3-left. While this set already gives us the basic information needed, we further optimize it in order to reduce the number of elements and to focus on encoding upper-body relationships with the environment.

A smaller representative set is defined by subdividing the character into sections of interests. In this chapter these sections comprise the torso, the left arm, and the right arm. Each section of interest $\{S_1, S_2, S_3\}$ is therefore composed of multiple geometries.

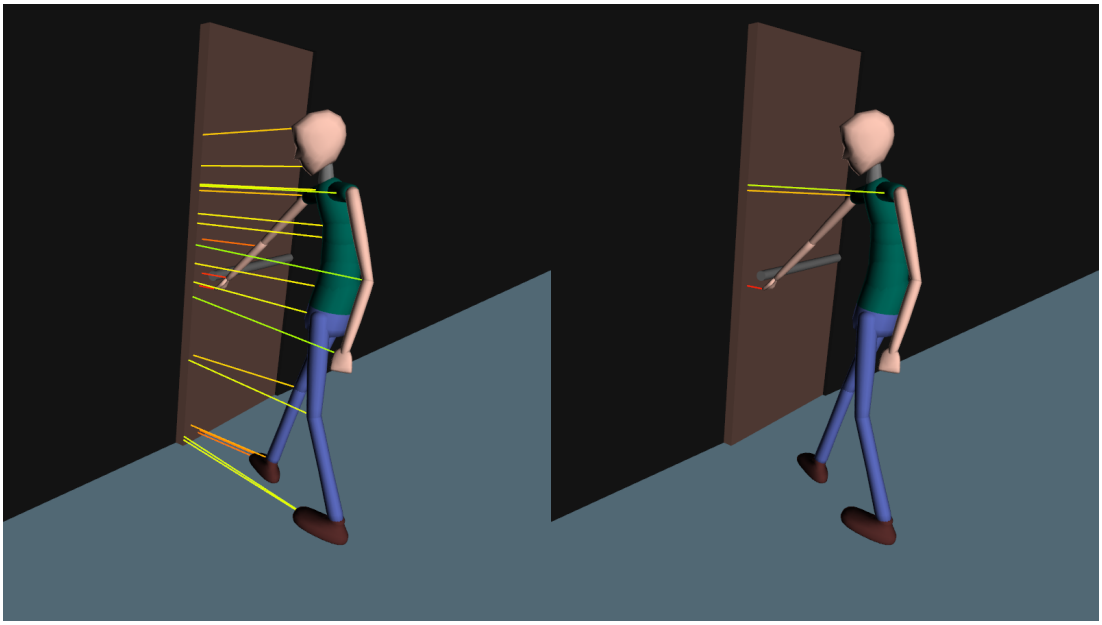

FIGURE 5.3. *Left:* Set of minimum vectors to the environment considering all body parts independently. *Right:* Set of minimum vectors with respect to only the body sections of interest.

The left arm is composed of the geometries associated with the left hand, left forearm and left upper arm. The right arm is equivalently defined, and the torso contains the spine and head geometries. For each section, only one vector $\boldsymbol{d}'_{\text{min}}(S_i)$ is used to represent the section, which is the vector of minimum length among the vectors computed for each body part in the section. The computation of $\boldsymbol{d}'_{\text{min}}(S_i)$ still requires traversing all involved BVHs, but can be optimized by using the length of the $\boldsymbol{d}'_{\text{min}}$ computed so far to prune subsequent BVH traversals.

Our environment coordination parameters are thus encoded as

$$\boldsymbol{d}^i = (d'_{min}(S_1), d'_{min}(S_2), d'_{min}(S_3)),$$

and are illustrated in Figure 5.3-right. Every pose p_i is therefore associated with a $\boldsymbol{d}^i$ encoding vectors of minimum distances between each selected body section and the environment.

The other part of the CFV encodes time coordination parameters $\boldsymbol{\sigma}^i$, which consist of a normalized time parameterization of each phase of the action being represented. A phase is identified as a sub-segment of an action which requires different coordination strategies than its neighboring phases. In such cases, $\boldsymbol{\sigma}^i$ is used to inform our regression module in which phase the character currently is during motion synthesis, such that coordination strategies from different phases are not mixed.

For example, our door opening motion examples clearly have three phases: approach, interaction, and release, while our drawing motion examples are modeled without a release phase. The time coordination parameters for an example motion have 0 for the first frame of the motion, and 1 for the last frame of the first phase

in the motion. Frames in subsequent phases are again annotated with values from $n-1$ to n for the nth phase. The specification of phases is task-dependent and will be described in Section 5.6 specifically for the presented results.

5.4.2. Posture Descriptors. Posture Descriptors (PDs) are also associated to the frames of the example motions. A PD for pose p_i is defined as $\boldsymbol{a}_i = \{\boldsymbol{e}^i_1, \ldots, \boldsymbol{e}^i_m, \boldsymbol{\omega}^i\}$, where $\boldsymbol{e}^i_j$, encodes the position and/or orientation of the end-effector $j \in \{1, \ldots, m\}$ at frame i, and $\boldsymbol{\omega}^i$ encodes the rotations of spine joints. These parameters represent key coordination information which will later guide our online posture optimizer.

In the presented examples we consider the hands and head of the character to be end-effectors and we encode the hand positions and the head orientation in our $\boldsymbol{e}^i_j$ vectors. These end-effectors are the ones corresponding to each body section of interest $\{S_1, S_2, S_3\}$. The head is considered an end-effector in order to integrate head orientation as part of the coordination model, encoding important gazing movement during action execution.

Parameter $\boldsymbol{\omega}^i$ encodes spine joint rotations. It consists of a quaternion describing the accumulated rotation of the spine joints. The spine rotations at pose p_i are extracted from the example motions as a set of quaternions $\{\mathbf{q}^k_{p_i}\}$, where $k \in \{1, \ldots, N_k\}$ is the index of the quaternion values from the lumbar joint to the up-most cervical vertebrae joint. We then define $\boldsymbol{\omega}^i = \prod_{j=1}^{N_k} \mathbf{q}^j_{p_i}$ in order to encode the rotations as a single one. This definition allows us to work with a single quaternion value to encode the overall spine movement during an action. As shown in the presented results, spine rotations encode essential movements for avoiding collisions and for approaching manipulation targets.

5.4.3. Validation of Proposed Coordination Feature Vector. The need to encode relationships with the nearby environment during action execution is clear because it allows the recovery of posture parameters every time similar relationships are found in new situations. Our encoding is in local coordinates with respect to the skeleton's root such that equivalent relationships can be found independently of the location of the character in the scene.

Finding the right parameters to encode is however not straightforward. Our approach based on closest points to body sections leads to dynamic connections between the important character parts and the environment, instead of static ones linking to every body part of the character.

We confirm the validity of our approach not only with the presented results but also visually using dimensionality reduction. We have encoded three candidate CFVs and we show the results of reducing the dimensionality of each feature vector to 2 by using GPLVM Lawrence [2005] for one of our door opening motion examples. GPLVM has been proven to be a succesful representation of human motion modeling [Fan et al., 2011]. The three cases are discussed below in reference to the cases shown in Fig. 5.4.

Case A. In this case we consider our features to be the joint angle values of each pose, since learning in joint space is a common approach in some works. We can observe that the structure of consecutive poses contain several jumps and undesirable clusters, creating a problematic space for our time-dependent regression problem. In

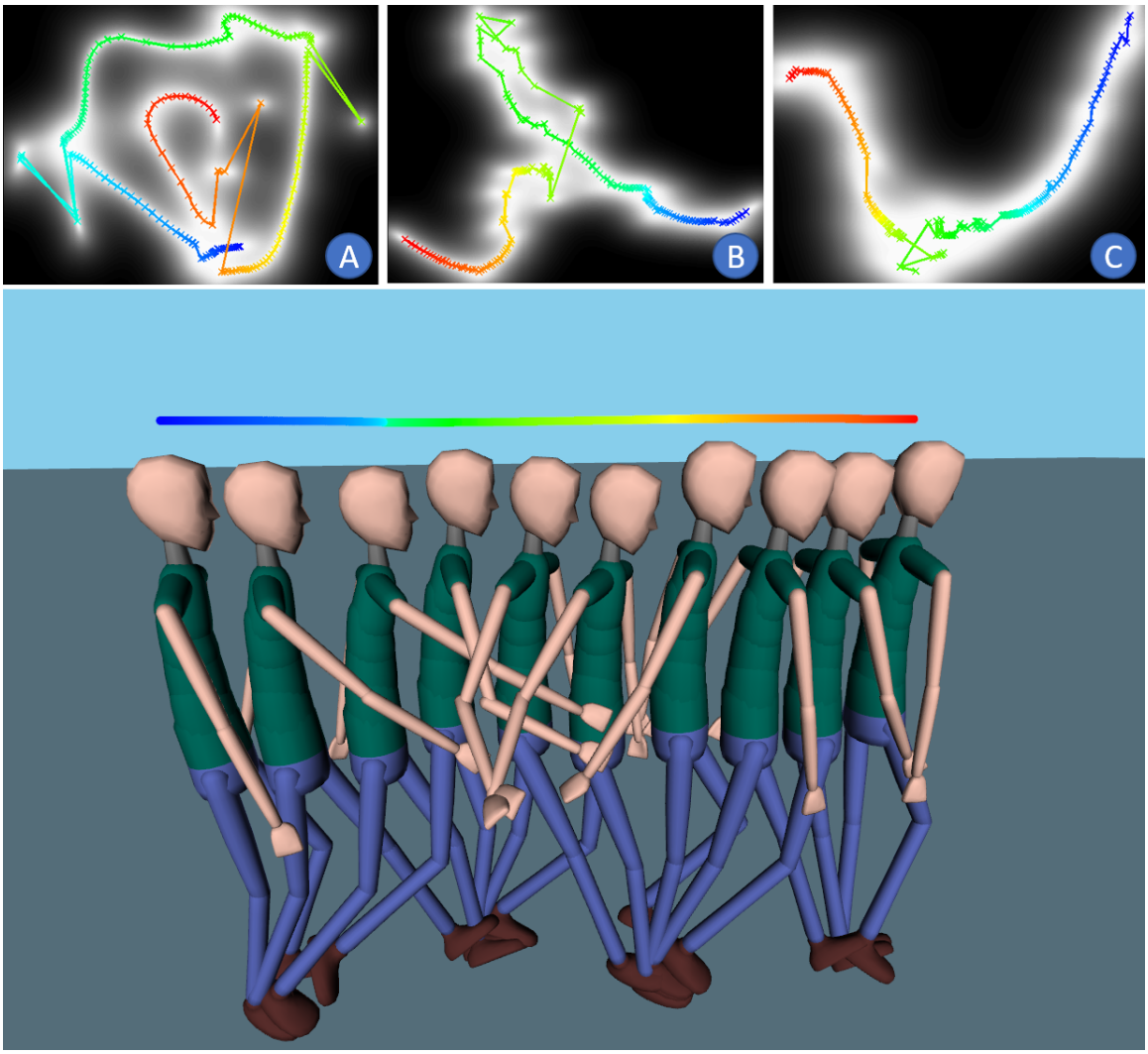

FIGURE 5.4. GPLVM visualization of different CFV encodings. **A:** joint angles, **B:** full feature vector; **C:** proposed feature vector. The motion used in this validation example is a door opening motion (bottom image). The color bar on top of the motion is used to label each of the poses by vertically projecting on the bar the skeleton's root position at each pose. These colors identify the poses corresponding to each curve point in images A, B and C.

addition, the lack of an environment-related representation prevents this description from being used to correspond learned poses with new environments.

Case B. In this case we have used our minimum distance vectors for all body parts of the character (Figure 5.3-left) instead of only using the proposed selected body sections. A continuous trajectory becomes noticeable, however, with a noisy section which corresponds to the part of the motion that maintains contact with the door. The noise comes from the constant changes of reference points which are

generated given the many body parts that are considered. Essentially the relevant body coordination gets diluted in the motion of the multiple moving body parts. This increases the diffidence of our regression method during synthesis, often retrieving inconsistent posture descriptors.

Case C. Here we have the proposed representation based on our minimal distance vectors. The obtained trajectory is much improved and the interaction area of the motion is mapped to points which appear close to each other, leading to lower diffidence values for our predictions.

5.4.4. Addressing Multiple Example Motions per Coordination Model. For simplicity of notation our description has considered that one coordination model is generated per example motion. However, as indicated in Figure 5.2, we can also build a coordination model for an action described by multiple example motions. Multiple example motions can be considered as long as the example motions encode similar body strategies, such that posture descriptors from corresponding frames in different example motions are not structurally different, since they will be considered by a single regression involving all motions.

The CFV relative to multiple example motions can be considered to be the concatenation of all the individual CFVs of each motion. This set of CFVs would not take into account the time of the original motion and its order would not affect the overall computation. During regression, PDs will be retrieved from the example motion best matching the CFV of the current pose being synthesized. Selected example motions may switch during the coordination, improving the number of situations that can be covered.

As it will be later described, in our drawing examples we consider multiple example motions in a single coordination model while in our door opening examples we use a single example motion per coordination model. The door examples still rely on multiple example motions, but each one is encoded as an independent coordination model given that the motions are structurally different from each other. In this case we let our method automatically select the best coordination model to use according to each encountered situation.

5.5. Real-Time Motion Synthesis with the Coordination Model

After processing example motions we obtain, for each target coordination model, a set of CFVs $\{\boldsymbol{Y}_i\} \subseteq \mathbb{Y}$ and a set of PDs $\{\boldsymbol{a}_i\} \subseteq \mathbb{A}$.

The goal of a coordination model is to learn a ***coordination function*** $\mathcal{C} : \mathbb{Y} \rightarrow \mathbb{A} \times \mathbb{R}^+$ such that, given any feature vector $\boldsymbol{Y} \in \mathbb{Y}$ extracted from a synthesized body pose, it returns an estimation of a PD $\boldsymbol{a} \in \mathbb{A}$ and a diffidence value $v \in \mathbb{R}^+$. Our diffidence value tells us that larger values of v mean low confidence, and small values of v mean high confidence of the prediction. In this way, our target becomes a diffidence value of 0, which indicates an exact match with the training data. The retrieved PDs will then be used to optimize the current pose being synthesized such that a successful coordination with the environment is achieved when executing the task at hand.

We compute $\mathcal{C}$ such that $\mathcal{C}(\boldsymbol{Y}_i) \simeq (\boldsymbol{a}_i, 0)\ \forall i \in \{1, \ldots, N\}$, that is, each captured pose should be associated with the original correspondent PDs and with lowest

diffidence value of the prediction. At the same time, our coordination function needs to work well on generically synthesized poses such that if the feature vector of a new given pose lies *close* to a previously learned feature vector, we will obtain PDs that will modify the pose meaningfully. We have explored different learning models including based on the GPLVM method used in the analysis of Fig. 5.4, and the selected proposed model is based on a local linear regression based on k-nearest neighbors. The proposed model achieved comparable or better results than the other tested methods, and is significantly faster.

5.5.1. Activation of the Coordination Function. The goal is to have a prediction lying close to the input data and we need the diffidence value to represent a good estimation of the feasibility of the action. The diffidence value of the prediction is used to select and decide when a coordination model is selected, and from that point forward the respective coordination function will be used to apply PDs to optimize the poses of the character. See Figure 5.2.

We begin with a walking controller that navigates the environment by following a path passing by the interaction points of the action. The path computation is dependent on the action and is described in Section 5.6 for each of our examples. Basically a suitable path can be computed once the target points of the action are identified. The available walking controller is only able to follow the path, as it does not have any information on how to control the upper-body or the speed of the motion for an interaction with the environment.

Multiple coordination functions may be available, one for each coordination data previously processed. While the character is far away from the first interaction point all the available coordination functions will have high diffidence indicating that the interaction has not yet started. When a diffidence value is low enough the system will then start to apply the respective coordination function in order to optimize the upper-body postures generated by the walking controller.

Given a CFV $\boldsymbol{Y_c}$ computed for the current walking posture, we compute distances to the training data $v_i = d(\boldsymbol{Y_c}, \boldsymbol{Y}_i)$, and we then select the k smallest distances in order to compute the diffidence value as:

$$v = \sum_{r=1}^{k} v_r,$$

such that $\{v_1, \ldots, v_k\}$ are the smallest k distances and r is the index of the r-th smallest distance.

When a CFV $\boldsymbol{Y_c}$ is computed for the current walking posture a time coordination parameter σ^c has also to be specified. While no coordination model is active the value taken is 0. Then, when a coordination model is activated we are able to predict time values from the training data and use them in the computation of the CFV $\boldsymbol{Y_c}$ of the subsequent frame. In this way the active coordination model starts to predict and control the speed of the walking generated by the walking controller (see Figure 5.2.)

While a coordination model is active the diffidence values are constantly checked and the interaction terminates when high diffidence is detected.

5.5.2. The Coordination Function. The coordination function implements a diffidence-based k-nearest neighbors regression model. Let v be the diffidence value computed for the current frame from the values $\{v_1, \ldots, v_k\}$ corresponding to the k smallest distances to the CFVs of the active coordination model. We similarly associate each of the CFVs and PDs with the same indexing scheme, such that the smallest value v_1 corresponds to the CFV $\mathbf{Y}_1$ and the PD $\mathbf{a}_1$. Weight values are then computed such that their sum is 1 and they correspond inversely to each diffidence value:

$$w_i = \frac{v - v_i}{(k-1)v} \quad ,$$

which will give us the following PD corresponding to $\boldsymbol{Y_c}$:

$$\boldsymbol{a} = \sum_{i=1}^{k} w_i \boldsymbol{a}_i.$$

The computation of $\mathcal{C}(\boldsymbol{Y_c}) = (\boldsymbol{a}, v)$ is then obtained. At the beginning v will have high values and only when v becomes smaller than a pre-defined threshold indicating low diffidence, the corresponding coordination model is selected and activated.

Figure 5.5 shows the behavior of our coordination function. At the beginning the distances of the k-nearest neighbors are too high, giving us high diffidence and forcing our method to wait until lower diffidence is reached in order to begin the interaction. This is represented by the warmer colors at the image top-left corner. In the bottom right corner of Figure 5.5 we can see that after the action is completed the coordination function begins to gain diffidence which terminates the effects of the coordination model over the action.

5.5.3. Applying the Coordination Function during Motion Synthesis. Once the coordination function $\mathcal{C}$ is ready it can then be used to synthesize new variations of the same type of object interaction. We have designed our method to be applied on top of generic walking controllers [Park et al., 2004; Lockwood and Singh, 2011], such that successful action coordination can be achieved with regular walking controllers. We used the walking controller that we described on Chapter 3.

First, a path for the new interaction to be executed is defined. This path can in most cases be planned by any clearance-based path planning method Kallmann [2014], by computing a path passing near the action points in the given interaction. Paths can also be computed towards generic target points given by the user, without considering any target action. In this case, if a coordination model is activated during path following, the upper-body motion will automatically be controlled in order to execute any needed interactions while following the path. This is the typical case for opening door actions performed in an environment with multiple doors. Paths are planned without considering the doors, and the coordination model will open doors when needed during path following.

Once a path is obtained the character starts to follow it and $\mathcal{C}(\boldsymbol{Y})$ is computed for every frame. When diffidence is low the activated coordination module will start to

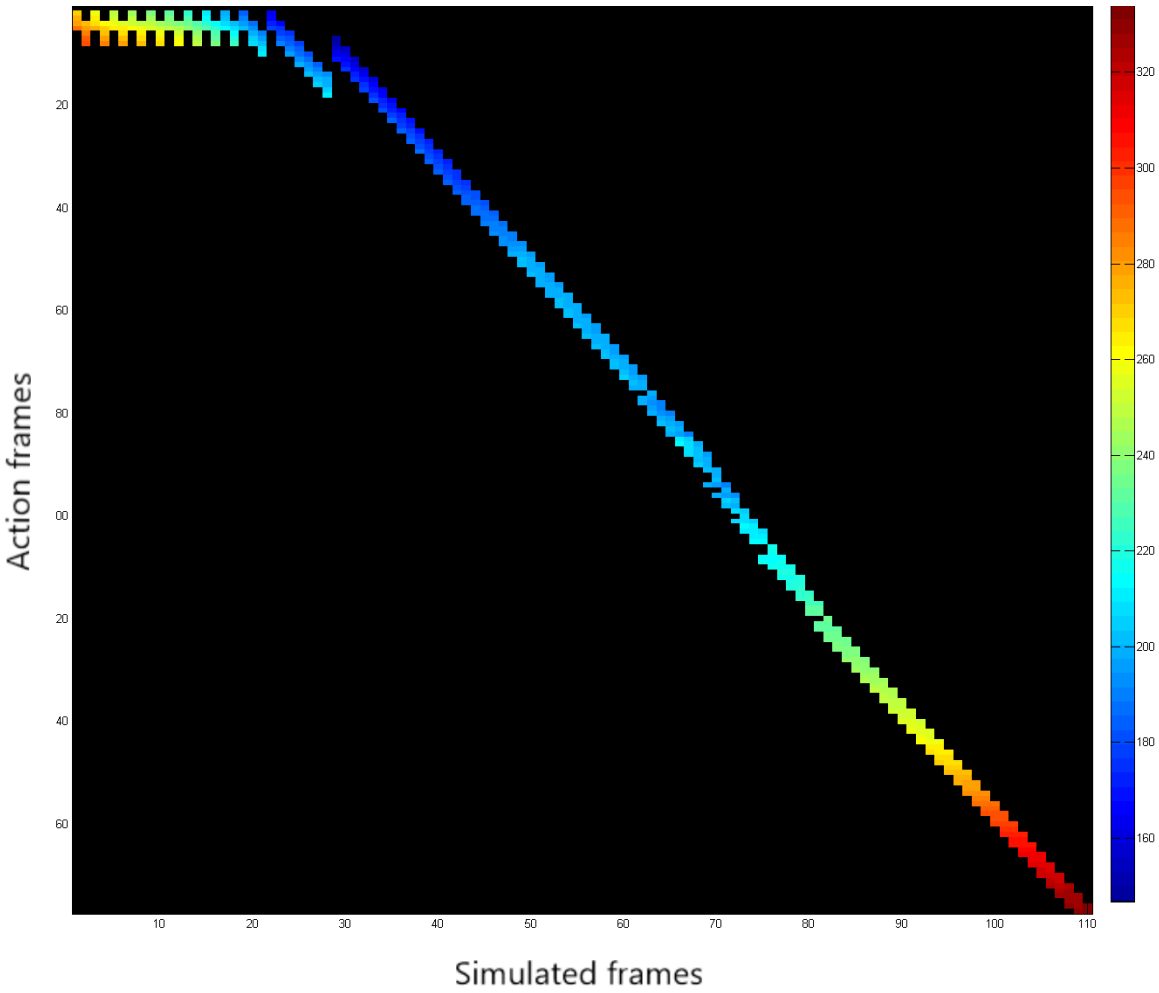

FIGURE 5.5. The image represents for every simulated frame the distance value v_j of each element in the set of k-nearest neighbors. Every non-black pixel in a column of the image corresponds to one of the selected k-nearest neighbors for the corresponding simulated real-time frame. The image shows that the k-nearest neighbors are most often composed of adjacent frames in the training data, indicating high correlation between the training data and the action being synthesized.

estimate and apply the PD retrieved by $\mathcal{C}(\boldsymbol{Y}) = (\boldsymbol{a}, v)$ to optimize the upper-body postures produced by the walking controller.

We perform the body optimization procedurally. Given our encoding of the PDs, they can be efficiently enforced in a given pose by IK. We select fast procedural and analytical joint optimization procedures instead of a global Jacobian-based IK approach. The PDs include target rotations for the spine joints, and target positions and/or rotations for the end-effectors. In order to deform the current posture towards the estimated targets, we first apply incremental rotations to the spine joints until the

estimated target is achieved. Then, we solve the IK for the end-effectors analytically, without moving the optimized spine joints.

The estimated time coordination parameter is used to estimate how far we are from the next interaction point, such that the current end effector target can be gradually interpolated towards it. In this way, as the current pose gets close to the next object contact point, IK will gradually enforce reaching object contact at the right time. IK enforcement is also used during a contact phase for preserving the contact during the phase. This case happens during the opening phase of a door, to maintain the hand in contact with the door while the door is open.

In addition, our synthesized actions require a dynamic time warping operation in order to obtain correct speed profiles. In order to achieve this we compare the previous environment coordination parameter σ_{t-1} and the current one σ_t, where t represents the frame number in a constant frame rate. If the distance between the two values surpasses a scalar factor of the expected distance of the two adjacent predictions, we then reduce the speed of the locomotion controller, compute a new feature vector, and restart the posture optimization process. Equivalently, if the distance is detected to be below, the speed of the locomotion controller is naturally increased.

This procedure basically replicates the speed profiles of the example motions. In most of our presented results it is possible to observe that the character speed naturally decreases during an interaction, then gradually increases as the interaction ends, until returning to the default walking speed of the locomotion controller.

5.6. Results and Discussion

Our method has been successfully applied to solve diverse full-body object interactions in varied situations. We have performed several experiments in two main scenarios: opening doors, and drawing in a wide whiteboard. All presented results were generated in real-time.

5.6.1. Scenario 1: Door Opening. In this scenario we learned coordination feature vectors from three example motions encoding different ways of opening a door according to three different approach angles. The approach angles were 0°, 90° and 180° degrees, considering the door handle as the center of a 2D coordinate system with the X axis parallel to the door wall and the Y axis perpendicular the wall as shown in Figure 5.6.

In this scenario the input example motions are structurally different. There is wide variance in the positions and velocities of the end-effectors, the timing and amplitude of the spine and arm movements vary significantly, and the trajectories of the end-effectors are considerably different. Clearly the right arm movements in the approach angles of 0° and 180° are quite different because each move in a completely different direction with respect to the body. We therefore choose to implement an independent coordination model for each example motion.

We divide each of our examples in 3 phases: approach, contact, and release. The division points are defined based on the handle first contact and release frames. Let c be the frame with the first handle contact and r the frame first releasing from handle contact, such that $0 < c < r < N$. The time coordination parameter σ^i is defined such

FIGURE 5.6. In left-right order: the example motions had approach angles of: 0°, 90° and 180° degrees. The postures in this image were synthesized with our coordination method given that differences in the character size and door dimensions lead to collisions with the considered environment when displaying the original data.

that it parameterizes each phase with unit intervals starting at 0, 1, and 2; as follows:

$$\sigma^i = \begin{cases} \frac{i}{c}, & 0 < i \leq c; \\ 1 + \frac{i-c}{r-c}, & c < i \leq r; \\ 2 + \frac{i-r}{N-r}, & r < i \leq N. \end{cases}$$

Given a door being approached, the coordination model will therefore evaluate which coordination model to activate based on the one with lowest diffidence, triggering the best model for the current situation. In order to make a robust selection, the lowest-diffidence model is selected as the first model to accumulate enough continuous frames of low diffidence. In our experiments, we found that using half a second presented good results. At long distances, all the coordination functions will return very high diffidence values and a selection will only happen at the right time.

In this scenario we generate paths with a path planner considering all doors open. Obstacles are defined as projected walls and open doors to the floor plane and our character is considered as a disk traversing the environment such that a clearance-based path planner can be applied Kallmann [2014]. Given a goal point selected interactively by the user the planner computes a feasible path towards the goal and possibly passing through a door. The planner ignores the time at which the door opens or the motion of the door while opening. The returned path does not have any annotation on speed or distance to the door. The character speed and the door opening are completely controlled by the activated coordination model. The door angle is simply estimated according to the time coordination parameter of the first phase, since at the end of the first phase the door should be completely open.

Overall we achieve a controller which is able to handle any approach direction and which is robust to obstacles perturbing the approaching trajectory (see Figure 5.7). We developed an application that demonstrates a real-time character able to open doors in any direction and while avoiding obstacles in the way.

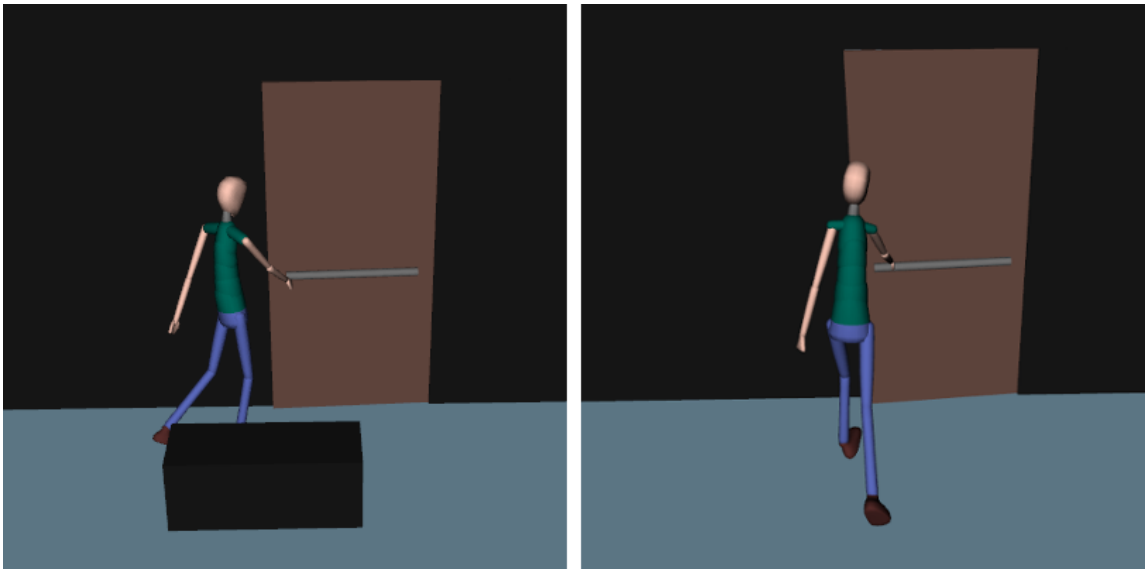

FIGURE 5.7. By allowing obstacles to be placed anywhere in the scene the approach direction to open a door can be arbitrary (left) instead of always front-facing (right). Our examples motions cover different approach directions and the most suitable coordination model is automatically selected by the system.

5.6.2. Scenario 2: Drawing on a Wide Whiteboard. The motion examples for this scenario were 18 different motions of a same person approaching and drawing straight lines in a wide whiteboard. The actor was in particular instructed to draw lines connecting extreme points in the whiteboard in an attentive way, for example, from the left-top corner of the board to the right-bottom corner, such that locomotion was required while drawing the lines. The attentive style led to example motions where, when possible, the spine remained visibly close to the end-effector.

The example motions were divided in 2 different phases: approaching and drawing. Using similar notation as in the door example, let c be the frame that starts hand contact with the whiteboard, $0 < c < N$. Our time coordination parameter becomes:

$$\sigma^i = \begin{cases} \frac{i}{c}, & 0 < i \leq c; \\ 1 + \frac{i-c}{N-c}, & c < i \leq N. \end{cases}$$

In this scenario the example motions were structurally similar. As long as postures are compared in a same phase, poses from different example motions can be used as they all encode appropriate poses from a current position to reach a given point in the whiteboard. We have therefore combined all example motions in a single coordination model.

For this scenario we implemented an example application where full-body motions can be executed to perform any given drawing on the whiteboard. We allow the user to perform a freehand drawing on the whiteboard model with the mouse. Once the drawing is complete, our system generates a walking path based on the drawing

structure, which is basically a horizontal path parallel to the whiteboard and covering the entire extent of the drawing in the whiteboard. When instructed to execute the drawing the character approaches the whiteboard and then follows the path with the walking controller performing side-stepping motions. The speed of the side-stepping is automatically controlled by our coordination model, which reduces the locomotion speed as needed according to the evolution of the drawing. In this scenario horizontal lines require faster side-stepping while vertical lines may stop the character from walking.

Figure 5.8 illustrates the obtained result. An infinity symbol is given as input to the system and the coordination model controls the upper-body on top of side-stepping motions in order to correctly accomplish the full-body drawing action.

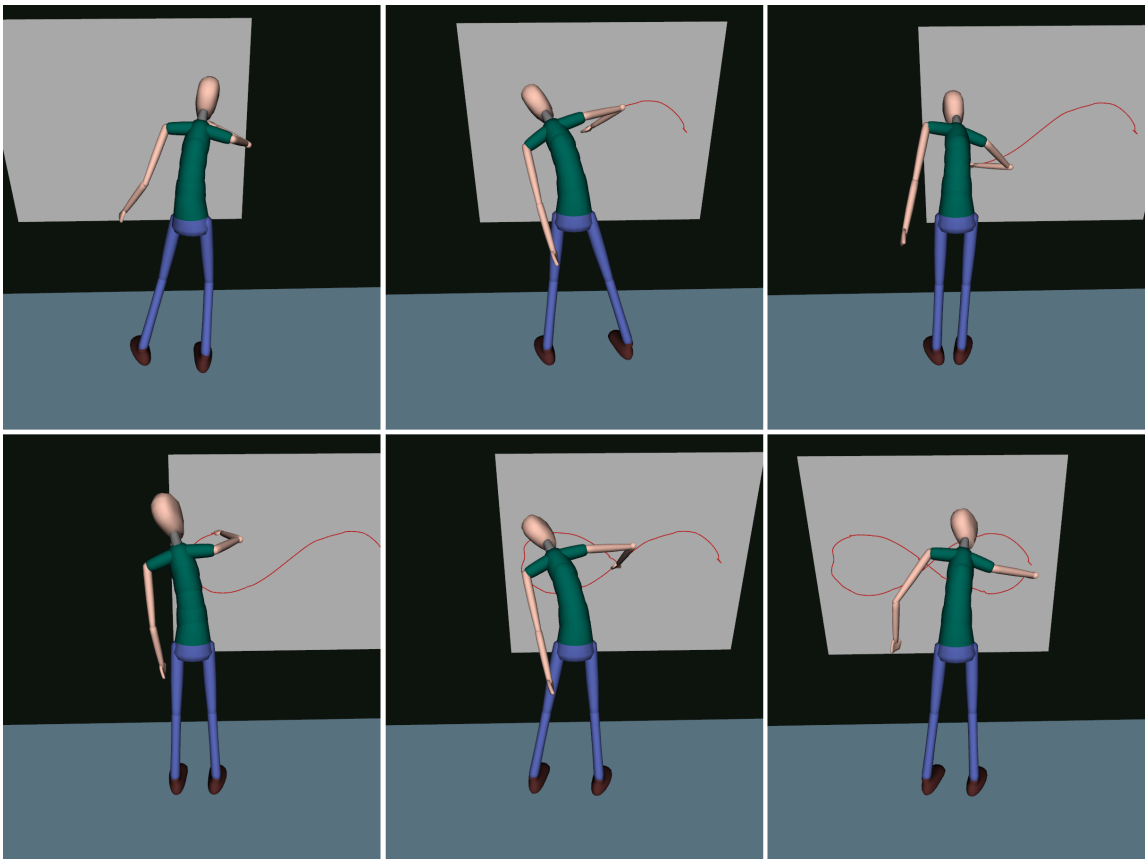

FIGURE 5.8. Drawing sequence for a generic shape. This example was synthesized with a coordination model built from straight line drawing examples in attentive style.

5.6.3. Performance Evaluation. Table 5.1 shows average computation times of the main modules of our system. The most expensive module is the distance computation to nearby obstacles, which is required for extracting feature vectors during motion synthesis. Still, distance computation only requires a few milliseconds per frame, and the other modules are executed under a millisecond. The feature

TABLE 5.1. Average computation times, in miliseconds, taken by the main modules of our method.

Example	Locomotion	Feature extraction	Coordination
Door	0.04	7.43	0.09
Board	0.05	2.79	0.2

extraction in the whiteboard scenario is faster because the environment has less triangles being considered. However, the regression computation time is higher because the whiteboard scenario relies on a larger set of examples for the computation of the k-nearest neighbors.

5.6.4. Discussion. Our method has shown to successfully replicate indispensable body coordination strategies needed to accomplish complex full-body actions in coordination with locomotion.

Fig. 5.9 illustrates the importance of the speed control and spine optimization modules of the coordination model. The figure shows obtained postures at equivalent frames when key system modules are turned off. The speed control regulates the necessary time for the character to walk while moving the arm to reach the handle of the door. If the character goes too fast it will collide with the door, while if it goes too slow the arm will not reach the door handle at the right time. Spine optimization is critical to improve realism and to allow the character to avoid obstacles and to reach targets during the interaction. In the right-most images it is also possible to notice the unrealistic straight spine that is obtained without spine control.

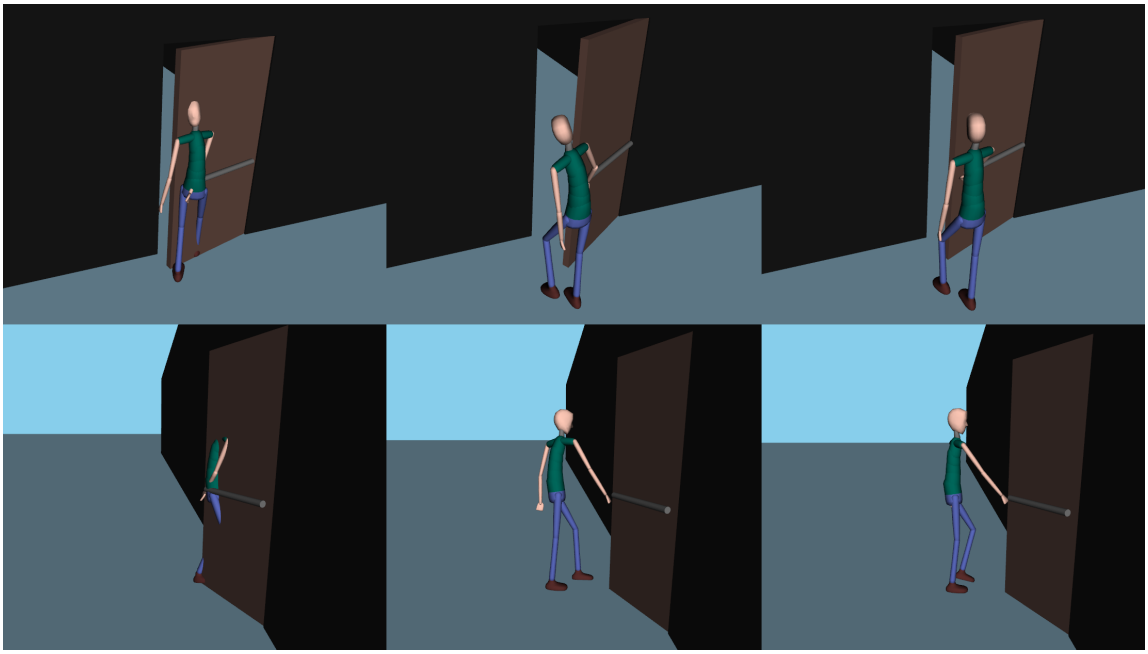

FIGURE 5.9. Left: Speed control turned off. Center: All modules turned on. Right: Spine deformations turned off.

Our results also include a gaze model which relies on the estimated time coordination parameter in order to control the gaze of the character towards the manipulation target. Additional posture descriptors could be easily integrated in the proposed methodology in order to capture other types of coordination strategies. In summary, our overall methodology is able to successfully produce a number of full-body interactions coordinated with generic controllable walking, and is also able to automatically select the most suitable coordination model in scenarios with multiple possible actions in diverse situations.

Using a combination of a lower body and independent upper body controllers has been done before. Heck et al. [2006] and van Basten et al. [2011] presented splicing methods, making the argument that time and space constraints are essential in this types of strategies. Meanwhile Zhang et al. [2009] proposed that subdividing the posture optimizer is a fast and reliable approach when there are several constrains to meet during motion synthesis. Al-Asqhar et al. [2013] demonstrated that using environment distance relationships for posture descriptions can create a generalized model for synthesis on environment variations. Finally, Bai et al. [2012] and Agrawal and van de Panne [2016] presented the design of motion planning around motion tasks. Our work borrows on some of these ideas and improves over the autonomy of the presented previous works by allowing the synthesis of complex tasks without requiring large databases.

As limitations, our method has to be applied to solve tasks which are structurally similar to the example motions. The overall approach is however suitable for generating a collection of full-body controllers able to address all the variations possible to be encountered in a given application. Another limitation is that by relying on a generic walking controller it is possible to observe a mismatch between the style of the locomotion and the style of the example motions, possibly degrading the overall realism of the synthesized actions. If high-quality results are required attention to capturing meaningful example motions and relying on suitable walking controllers may be needed. Example motions and data-based walking controllers can also be built from data of the same actor.

In our presented results the walking motions come from a different actor than the action examples, demonstrating that successful motions can be achieved with generic walking controllers, without the need to rely on large sets of collected motions or on extensive data fine-tuning. Overall, our approach focuses on achieving autonomous characters able to perform complete full-body interactions with the environment only from given high level commands, and relying on minimal learning requirements. The presented results demonstrate the capabilities of the approach.

5.7. Conclusion

We introduce in this chapter a new methodology for learning complex environment-dependent body coordination models which can be applied on top of generic walking controllers. Our approach only requires a few example motions during the learning phase, which consists of a simple and fast feature precomputation procedure. Our overall approach is suitable for real-time applications. As a result our method achieves

virtual characters demonstrating high levels of autonomy when solving complex full-body object interactions.

CHAPTER 6

Conclusions

Creating an autonomous character capable of navigating and interacting with its environment is an interesting challenge. There is an important trade-off between controller flexibility and overall quality which makes this problem particularly complex to solve effectively given that both goals are desired.

In this dissertation I present independent but easily integratable approaches for character navigation and environment interaction. With the proposed framework I am able to find annotated paths in complex environments at real time rates and our methods are general enough that integrating extra features into the motion planner, layers or coordination is a straightforward task.

The initial work on locomotion presented in chapter 3 allowed the design of the methods of the subsequent chapters. In my experience this showed that an easily controllable, flexible and simple locomotion controller is essential in the development of animation tools. Specifically tools that require a high level control for the locomotion. My framework also takes advantage of the lack of restrictions over the locomotion controller.

My motion planning approach takes advantage of the ability of following any 2D path in the environment by exploring secondary objectives on the path that are usually addressed during run time. The proposed planner can guarantee collision free paths while working on dense environments. The generated paths are designed to minimize a cost function that considers comfort and distance, which is not usually explored. Prior approaches require a reactive phase that can alter significantly the path selection, limiting the capability to know the character location and narrowing possible applications that require direct predictions on the character behavior.

My feature description of the environment interaction done by the characters provides a novel approach to motion characterization for environment interaction. The proposed featured vectors are capable of describing complex interactions that achieve multiple objectives in a natural approach. A task as simple as opening a door requires a lot of considerations that are hard to characterize on a general approach. In this dissertation I showed how this task can be approached using a flexible framework capable of solving general mobile interactions in real time.

6.1. Limitations

As described on Section 1.3, data acquisition is time consuming and some of the motion capture sessions were needed to be discarded due to actors performing exaggerated motions. In my experience, continuous work with this type of data reduces the capacity to perceive low quality results, increasing the potential of developing

methods with unwanted results. In professional pipelines, the capture, cleaning and processing is performed by different people and there is a lot of interaction between the artistic and technical sides of the work. This reduces considerably the previously discussed limitation, however it is not a common practice followed by research approach to motion synthesis, which limits the potential quality of the generated results.

By design, the choice of feature vector can generate a discontinuous space, which is not optimal for a k-nearest neighbors regression approach, however in practice I didn't find any motion inconsistency. Figure 5.5 shows that the k-nearest neighbors found follow a linear time similar to the original motion providing a desired behavior. This can be greatly enforced by selecting only small windows around the previous prediction such that the desired time-relationship is obtained. Similarly the coordination function is only activated when the k-nearest neighbors returns a good diffidence value. This restricted the amount of potential errors that could have been encountered.

Working on integrating independent upper body and lower body controllers is challenging. It introduces potential errors on the physics of the original motions and ignores possible twists that propagate among the whole which can generate erroneous poses during synthesis. The speed control limits this possible errors, however it doesn't discard the possibility of creating unwanted results.

6.2. Future Work

In chapter 3 I explored a series of validity measures that achieved adequate results for this dissertation purposes. A number of additional directions can be explored in terms of defining and measuring synthesized motion quality. The ability to visualize spatial coverage and quality not only informs building a collection of mobility controllers achieving high-quality coverage, but also introduces a generic methodology for analyzing quality metrics spatially. I intend to include both perceptual and ergonomic metrics to the quality evaluation of the produced motions. Perceptual metrics can be specified with the help of human studies, where bounds on each deformation operation can be specified perceptually in order to guarantee perceptually-validated high-quality motions. Ergonomic metrics can be incorporated with the inclusion of movement energy expenditure metrics and as well stability or body balance metrics. In addition, similar metrics computed over the upper-body motions generated by IK during manipulation can also be incorporated in order to extend the coverage-quality analysis to address manipulation motions.

The behavioral layered path planner deals with a generalization of navigation meshes. A future avenue on this work can be directed into generate a formal geometric description of the layered problem as well as to adapt and solve a mesh that integrates the needs of the planner directly. With a proper analysis on this method I could properly define an optimal solution for this problem as well as provide an optimization algorithm than could converge to it. In order to increase the strength of my results, it is necessary to present a formal comparison showing benchmarks in the same environments done by other approaches. Unfortunately, most well-known approaches do not address directly cluttered environments and multiple behaviors so designing

appropriate benchmarks will require special attention to the specific limitations of different approaches.

The methods presented in section 4 and 5 are flexible and could be integrated to design a planner capable of environment interaction. During the planing phase, the environment interaction can be treated as an specific behavior with an specific weight attached to it, that way the planner will select the most appropriate way to navigate the environment considering when and if an obstacle should be contemplated for interaction. A system like this has the potential to solve generic everyday navigation actions using high level instructions.

A limitation on the environment interaction solution presented is the need to compute potentially expensive proximity vectors. It might be possible to accelerate its computation using a layered approach in the same way as it was presented on the motion planner of section 4.6. Another setback appears on the selection of the Coordination Feature Vector, a deeper analysis exploring different encodings could provide more consistent results in the motion synthesis.

References

Abe, Y. and Popović, J. (2006). Interactive animation of dynamic manipulation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '06, pages 195–204, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Agrawal, S. and van de Panne, M. (2016). Task-based locomotion. *ACM Trans. Graph.*, 35(4):82:1–82:11.

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*, SCA '13, pages 45–53, New York, NY, USA. ACM.

Arikan, O. and Forsyth, D. A. (2002). Synthesizing constrained motions from examples. *Proceedings of SIGGRAPH*, 21(3):483–490.

Arikan, O., Forsyth, D. A., and O'Brien, J. F. (2003). Motion synthesis from annotations. *Proceedings of SIGGRAPH*, 22(3):402–408.

Bai, Y., Siu, K., and Liu, C. K. (2012). Synthesis of concurrent object manipulation tasks. *ACM Trans. Graph.*, 31(6):156:1–156:9.

Choi, M. G., Kim, M., Hyun, K. L., and Lee, J. (2011). Deformable motion: Squeezing into cluttered environments. *Computer Graphics Forum*, 30(2):445–453.

Choi, M. G., Lee, J., and Shin, S. Y. (2002). Planning biped locomotion using motion capture data and probabilistic roadmaps. *Proceedings of SIGGRAPH*, 22(2):182–203.

Choi, M. G., Lee, J., and Shin, S. Y. (2003). Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Transactions on Graphics*, 22(2):182–203.

Coros, S., Beaudoin, P., and van de Panne, M. (2010). Generalized biped walking control. *ACM Trans. Graph.*, 29:130:1–130:9.

Eppstein, D. (1999). Finding the k shortest paths. *SIAM J. Comput.*, 28(2):652–673.

Esteves, C., Arechavaleta, G., Pettré, J., and Laumond, J.-P. (2006). Animation planning for virtual characters cooperation. *ACM Transaction on Graphics*, 25(2):319–339.

Faloutsos, P., van de Panne, M., and Terzopoulos, D. (2001). Composable controllers for physics-based character animation. In *Proceedings of SIGGRAPH*, pages 251–260, New York, NY, USA. ACM Press.

Fan, G., Zhang, X., and Ding, M. (2011). Gaussian process for human motion modeling: A comparative study. In *Machine Learning for Signal Processing (MLSP), 2011 IEEE International Workshop on*, pages 1–6. IEEE.

Gleicher, M., Shin, H. J., Kovar, L., and Jepsen, A. (2003). Snap-together motion: assembling run-time animations. In *Proceedings of the symposium on Interactive*

3D graphics and Games (I3D), pages 181–188, NY, USA.

Hauser, K. K., Ng-Thowhing, V., Gonzalez-Baos, Mukai, H., and Kuriyama, S. (2007). Multi-modal motion planning for a humanoid robot manipulation task. In *International Symposium on Robotics Research (ISRR)*.

Heck, R. and Gleicher, M. (2007). Parametric motion graphs. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, I3D '07, pages 129–136, New York, NY, USA. ACM.

Heck, R., Kovar, L., and Gleicher, M. (2006). Splicing upper-body actions with locomotion. In *Proceedings of Eurographics 2006*.

Ho, E. S. L., Komura, T., and Tai, C.-L. (2010). Spatial relationship preserving character motion adaptation. *ACM Trans. Graph.*, 29(4):33:1–33:8.

Hodgins, J. K., Wooten, W. L., Brogan, D. C., and O'Brien, J. F. (1995). Animating human athletics. In *Proceedings of SIGGRAPH*, pages 71–78.

Huang, Y. and Kallmann, M. (2016). Planning motions and placements for virtual demonstrators. *IEEE Transactions on Visualization and Computer Graphics*, 22(5):1568–1579.

Huang, Y., Mahmudi, M., and Kallmann, M. (2011). Planning humanlike actions in blending spaces. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Jain, S. and Liu, C. K. (2009). Interactive synthesis of human-object interaction. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, pages 47–53, New York, NY, USA. ACM.

Jain, S. and Liu, C. K. (2011). Modal-space control for articulated characters. *ACM Trans. Graph.*, 30(5):118:1–118:12.

Jang, W.-S., Lee, W.-K., Lee, I.-K., and Lee, J. (2008). Enriching a motion database by analogous combination of partial human motions. *Vis. Comput.*, 24(4):271–280.

Johansen, R. S. (2009). *Automated semi-procedural animation for character locomotion*. PhD thesis, Citeseer.

Juarez-Perez, A., Feng, A., Kallmann, M., and Shapiro, A. (2014). Deformation, parameterization and analysis of a single locomotion cycle. In *Proceedings of the Seventh International Conference on Motion in Games*, MIG '14, pages 182–182, New York, NY, USA. ACM.

Juarez-Perez, A. and Kallmann, M. (2016a). Full-body behavioral path planning in cluttered environments. In *Proceedings of the 9th International Conference on Motion in Games*, MIG '16. ACM.

Juarez-Perez, A. and Kallmann, M. (2016b). Modeling data-based mobility controllers with known coverage and quality properties. In *The 4th International Digital Human Modeling Symposium (DHM '16)*.

Juarez-Perez, A. and Kallmann, M. (2017). Coordinating full-body interactions with the environment. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '17, pages 21:1–21:2. ACM.

Juarez-Perez, A. and Kallmann, M. (2018a). A coordination model for full-body interactions while walking. In *Preparation*.

Juarez-Perez, A. and Kallmann, M. (2018b). Fast behavioral locomotion with layered navigation meshes. In *Proceedings of the 21st ACM SIGGRAPH Symposium on*

Interactive 3D Graphics and Games, I3D '18. ACM.

Kallmann, M. (2014). Dynamic and robust local clearance triangulations. *ACM Trans. Graph.*, 33(5):161:1–161:17.

Kallmann, M. (2016). Flexible and efficient navigation meshes for virtual worlds. In Pelechano, N., Allbeck, J., Kapadia, M., and Badler, N., editors, *Simulating Heterogeneous Crowds with Interactive Behaviors*. Taylor & Francis.

Kallmann, M., Huang, Y., and Backman, R. (2010). A skill-based motion planning framework for humanoids. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2507–2514. IEEE.

Kallmann, M. and Kapadia, M. (2016). *Geometric and Discrete Path Planning for Interactive Virtual Worlds*. Morgan and Claypool Publishers.

Kim, M., Hwang, Y., Hyun, K., and Lee, J. (2012). Tiling motion patches. In *Proceedings of the 11th ACM SIGGRAPH / Eurographics Conference on Computer Animation*, EUROSCA'12, pages 117–126, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Kim, M., Hyun, K., Kim, J., and Lee, J. (2009). Synchronized multi-character motion editing. *ACM Trans. Graph.*, 28(3):1–9.

Kovar, L. and Gleicher, M. (2004). Automated extraction and parameterization of motions in large data sets. *ACM Transaction on Graphics (Proceedings of SIGGRAPH)*, 23(3):559–568.

Kovar, L., Gleicher, M., and Pighin, F. (2002). Motion graphs. *ACM Trans. Graph.*, 21(3):473–482.

Land, W. M., Rosenbaum, D. A., Seegelke, C., and Schack, T. (2013). Whole-body posture planning in anticipation of a manual prehension task: Prospective and retrospective effects. *Acta Psychologica*, 144(2).

Larsen, E., Gottschalk, S., Lin, M. C., and Manocha, D. (2000). Fast distance queries with rectangular swept sphere volumes. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 4, pages 3719–3726. IEEE.

Lasseter, J. (1987). Principles of traditional animation applied to 3d computer animation. *SIGGRAPH Comput. Graph.*, 21(4):35–44.

Lau, M. and Kuffner, J. J. (2006). Precomputed search trees: planning for interactive goal-driven animation. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA)*, pages 299–308.

Lawrence, N. (2005). Probabilistic non-linear principal component analysis with gaussian process latent variable models. *J. Mach. Learn. Res.*, 6:1783–1816.

Lee, H.-K. and Kim, J. H. (1999). An HMM-based threshold model approach for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):961–973.

Lee, J., Chai, J., Reitsma, P., Hodgins, J. K., and Pollard, N. (2002). Interactive control of avatars animated with human motion data. *Proceedings of SIGGRAPH*, 21(3):491–500.

Lee, K. H., Choi, M. G., and Lee, J. (2006). Motion patches: Building blocks for virtual environments annotated with motion data. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 898–906, New York, NY, USA. ACM.

Lee, Y., Lee, S. J., and Popović, Z. (2009). Compact character controllers. In *ACM SIGGRAPH Asia 2009 Papers*, SIGGRAPH Asia '09, pages 169:1–169:8, New York, NY, USA. ACM.

Lee, Y., Wampler, K., Bernstein, G., Popović, J., and Popović, Z. (2010). Motion fields for interactive character animation. *ACM Trans. Graph.*, 29(6):138:1–138:8.

Levine, S., Lee, Y., Koltun, V., and Popović, Z. (2011). Space-time planning with parameterized locomotion controllers. *ACM Trans. Graph.*, 30(3):23:1–23:11.

Levine, S., Wang, J. M., Haraux, A., Popović, Z., and Koltun, V. (2012). Continuous character control with low-dimensional embeddings. *ACM Trans. Graph.*, 31(4):28:1–28:10.

Liu, C. K. (2009). Dextrous manipulation from a grasping pose. *ACM Trans. Graph.*, 28(3):59:1–59:6.

Liu, L., Panne, M. V. D., and Yin, K. (2016). Guided learning of control graphs for physics-based characters. *ACM Trans. Graph.*, 35(3):29:1–29:14.

Lo, W.-Y. and Zwicker, M. (2008). Real-time planning for parameterized human motion. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '08, pages 29–38, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Lockwood, N. and Singh, K. (2011). Biomechanically-inspired motion path editing. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '11, pages 267–276, New York, NY, USA. ACM.

Mahmudi, M. and Kallmann, M. (2011). Feature-based locomotion with inverse branch kinematics. In *Proceedings of the 4th International Conference on Motion In Games (MIG)*.

Mahmudi, M. and Kallmann, M. (2012). Precomputed motion maps for unstructured motion capture. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '12, pages 127–136.

Mahmudi, M. and Kallmann, M. (2013). Analyzing locomotion synthesis with feature-based motion graphs. *Visualization and Computer Graphics, IEEE Transactions on*, 19(5):774–786.

Mahmudi, M. and Kallmann, M. (2015). Multi-modal data-driven motion planning and synthesis. In *Proceedings of Motion in Games*.

Min, J. and Chai, J. (2012). Motion graphs++: a compact generative model for semantic motion analysis and synthesis. *ACM Trans. Graph.*, 31(6):153:1–153:12.

Mukai, T. and Kuriyama, S. (2005). Geostatistical motion interpolation. In *ACM SIGGRAPH*, pages 1062–1070, New York, NY, USA. ACM.

Pan, J., Zhang, L., Lin, M., and Manocha, D. (2010). A hybrid approach for synthesizing human motion in constrained environments. In *Conference on Computer Animation and Social Agents (CASA)*.

Park, S. I., hoon Kim, T., and Shin, H. J. (2004). Online motion blending for real-time locomotion generation. computer animation and virtual worlds 15. *Comput. Animat. Virtual Worlds*, 15:3–4.

Peng, X. B. and van de Panne, M. (2017). Learning locomotion skills using deeprl: Does the choice of action space matter? In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '17, pages 12:1–12:13, New

York, NY, USA. ACM.

Pražák, M., Hoyet, L., and O'Sullivan, C. (2011). Perceptual evaluation of footskate cleanup. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '11, pages 287–294, New York, NY, USA. ACM.

Pražák, M., McDonnell, R., and O'Sullivan, C. (2010). Perceptual evaluation of human animation timewarping. In *ACM SIGGRAPH ASIA 2010 Sketches*, SA '10, pages 30:1–30:2, New York, NY, USA. ACM.

Raibert, M. H. and Hodgins, J. K. (1991). Animation of dynamic legged locomotion. In *Computer Graphics / Proceedings of SIGGRAPH*, volume 25, pages 349–358.

Rose, C., Bodenheimer, B., and Cohen, M. F. (1998). Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18:32–40.

Rosenbaum, D. A., Chapman, K. M., Coelho, C. J., Gong, L., and Studenka, B. E. (2013). Choosing actions. *Frontiers in Psychology*, 4(273).

Ryall, K., Hoyet, L., Hodgins, J., and O'Sullivan, C. (2012). Exploring sensitivity to time-warped biological motion. *Perception*, 41:149.

Safonova, A. and Hodgins, J. K. (2007). Construction and optimal search of interpolated motion graphs. *ACM Transactions on Graphics (Proceedings. of SIGGRAPH)*, 26(3).

Shapiro, A., Kallmann, M., and Faloutsos, P. (2007). Interactive motion correction and object manipulation. In *ACM SIGGRAPH Symposium on Interactive 3D graphics and Games (I3D)*, Seattle.

Shum, H. P., Hoyet, L., Ho, E. S., Komura, T., and Multon, F. (2014). Natural preparation behavior synthesis. *Comput. Animat. Virtual Worlds*, 25(5-6):531–542.

Treuille, A., Lee, Y., and Popović, Z. (2007). Near-optimal character animation with continuous control. In *Proceedings of ACM SIGGRAPH*. ACM Press.

van Basten, B. J., Egges, A., and Geraerts, R. (2011). Combinining path planners and motion graphs. *Computer Animation and Virtual Worlds*, 21:1–22.

van Basten, B. J. H. and Egges, A. (2011). Flexible splicing of upper-body motion spaces on locomotion. *Comput. Graph. Forum*, 30(7):1963–1971.

van Basten, B. J. H., Peeters, P. W. A. M., and Egges, A. (2010). The step space: Example-based footprint-driven motion synthesis. *Comput. Animat. Virtual Worlds*.

van de Panne, M., Kim, R., and Fiume, E. (1994). Synthesizing parameterized motions. In *Fifth Eurographics Workshop on Animation and Simulation, at Oslo*.

Yamane, K., Kuffner, J. J., and Hodgins, J. K. (2004). Synthesizing animations of human manipulation tasks. *ACM Trans. Graph.*, 23(3):532–539.

Yin, K., Loken, K., and van de Panne, M. (2007). Simbicon: Simple biped locomotion control. In *Proceedings of ACM SIGGRAPH*, San Diego, CA.

Zhang, L., Pan, J., and Manocha, D. (2009). Motion planning of human-like robots using constrained coordination. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, pages 188–195. IEEE.

Zhao, L., Normoyle, A., Khanna, S., and Safonova, A. (2009). Automatic construction of a minimum size motion graph. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

Zhao, L. and Safonova, A. (2008). Achieving good connectivity in motion graphs. In *Proceedings of the 2008 ACM/Eurographics Symposium on Computer Animation (SCA)*, pages 127–136.

Zimmermann, D., Coros, S., Ye, Y., Sumner, R. W., and Gross, M. (2015). Hierarchical planning and control for complex motor tasks. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, SCA '15, pages 73–81, New York, NY, USA. ACM.