UNIVERSITY OF CALIFORNIA, MERCED

Motion Capture Based Animation for Virtual Human Demonstrators: Modeling, Parameterization and Planning

A dissertation submitted in partial satisfaction of the requirements
for the degree Doctor of Philosophy

in

Electrical Engineering & Computer Science

by

Yazhou Huang

Committee in charge:

Professor Marcelo Kallmannn, Committee Chair
Professor Stefano Carpin
Professor Teenie Matlock
Professor Michael Neff

2012

ABSTRACT OF THE DISSERTATION

Motion Capture Based Animation for Virtual Human Demonstrators: Modeling, Parameterization and Planning

by

Yazhou Huang
Doctor of Philosophy in Electrical Engineering & Computer Science
University of California, Merced, 2012
Professor Marcelo Kallmannn, Chair

A huge collection of character animation techniques has been developed to date and impressive results have been achieved in the recent years. The main pursued approaches can be categorized as physics-based, algorithmic-based or data-based. High-quality animation today is still largely data-based and achieved through motion capture technologies. While great realism is achieved, current solutions still suffer from limited character control, limited ability to address cluttered environments, and disconnection from higher-level constraints and task-oriented specifications. This dissertation addresses these limitations and achieves an autonomous character that is able to demonstrate, instruct and deliver information to observers in a realistic and human-like way.

The first part of this thesis addresses motion synthesis with a simple example-based motion parameterization algorithm for satisfying generic spatial constraints at interactive frame rates. The approach directly optimizes blending weights for a consistent set of example motions, until the specified constraints are best met. An in-depth analysis is presented to compare the proposed approach with three other popular blending techniques, and the pros and cons of each method are uncovered. The algorithm has also been integrated in an immersive motion modeling platform, which enables programming of generic actions by direct demonstration of example motions.

In order to address actions in cluttered environments and maintain the realism of motion capture examples, the concept of exploring the blending space of example motions in then introduced. A bidirectional time-synchronized sampling-based planner with lazy collision evaluation is proposed for planning motion variations around obstacles while maintaining the original quality of the example motions. Coupled with a locomotion planner, it generates realistic whole-body motion in cluttered environments.

Finally, high-level specifications for demonstrative actions are addressed with the proposed whole-body PLACE planner. It is based on coordination models extracted from behavioral studies, where participants performed demonstrations involving locomotion and pointing in varied conditions. The planner achieves coordinated body positioning, locomotion, action execution and gaze synthesis, in order to engage observers in demonstrative scenarios.

The dissertation of Yazhou Huang is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Professor Stefano Carpin

Professor Teenie Matlock

Professor Michael Neff

Professor Marcelo Kallmannn, Committee Chair

University of California, Merced

2012

To my parents . . .
I would not be where I am today without your love and support.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

First and foremost, I would like to sincerely thank my academic advisor, Professor Marcelo Kallmann, for his outlook and guidance on my research topics over the past years. He has helped me gain perspective, confidence and problem-solving skills. This work could never be done without him.

I would like to thank all my committee members, Professor Teenie Matlock, Professor Stefano Carpin, and Professor Michael Neff (UC Davis), for the valuable inputs and suggestions they have given me on my research projects.

My research is partially supported by the National Science Foundation (NSF) Awards IIS-0915665 and CNS-0723281, the Center for Information Technology Research in the Interest of Society (CITRIS) seed funding grant, and the UC Merced Graduate & Research Council (GRC) fellowship grants.

I have had the pleasure to work with my colleagues and co-authors from UC Merced, who have inspired me on both professional and personal levels. This includes Carlo Camporesi, Robert Backman, Mentar Mahmudi and Justin L. Matthews. Some of the researches and chapters were made possible through our collaborative work.

I am also very grateful for the SmartBody team at USC Institute for Creative Technologies (ICT): Doctor Ari Shapiro, Doctor Wei-Wen Feng and Yuyu Xu, for having me as a summer intern, and for teaching me the approach to develop solutions that are practical. Chapter 4 and 8 are based on the research we have conducted together during my internship.

Many thanks to all the student workers and motion capture participants (specifically Chenté Cervantes) for their valuable assistance in the collection and annotation process in building the motion databases.

Last but not least, I would like to thank everyone that helped me in any way during my stay at UC Merced. I could not have done it without you all!

VITA

2002-2006	B.Sci. (Automation), University of Science and Technology Beijing (USTB), Beijing, China.
Jan–July 2007	Electrical Engineer, Tsinghua Tongfang Co., Ltd., Beijing, China.
2007-2012	Ph.D. (expected), Electrical Engineering & Computer Science Department, UC Merced, Merced, CA.
2007–2012	Research Assistant (advised by Prof Marcelo Kallmann), UC Merced.
Fall 2010	Teaching Assistant, CSE171 Game Programming, UC Merced
Fall 2007	Teaching Assistant, CSE140 Computer Architecture, UC Merced
Fall 2011	Application Developer Intern, EON Reality Inc., Irvine, CA
Summer 2012	Visiting Research Assistant, USC ICT, Los Angeles, CA.

PUBLICATIONS

Yazhou Huang and Marcelo Kallmann. *Motion Parameterization with Inverse Blending.* In Proceedings of the 3rd International Conference on Motion In Games (MIG), Netherlands, 2010.

Andrew (Wei-Wen) Feng, Yazhou Huang, Marcelo Kallmann and Ari Shapiro. *An Analysis of Motion Blending Techniques.* In Proceedings of the 5th International Conference on Motion In Games (MIG), Rennes, France, 2012.

Carlo Camporesi, Yazhou Huang and Marcelo Kallmann. *Interactive Motion Modeling and Parameterization by Direct Demonstration.* In Proceedings of the International Conference on Intelligent Virtual Agent (IVA), Philadelphia, Pennsylvania, 2010.

Yazhou Huang, Carlo Camporesi and Marcelo Kallmann. *Immersive Interfaces for Building Parameterized Motion Databases (poster).* In Proceedings of the International Conference on Intelligent Virtual Agent (IVA), Santa Cruz, CA, 2012.

Yazhou Huang, Justin L. Matthews, Teenie Matlock and Marcelo Kallmann. *Modeling Gaze Behavior for Virtual Demonstrators.* In Proceedings of the International Conference on Intelligent Virtual Agent (IVA), Reykjavik, Iceland, 2011.

Stephanie Huette, Yazhou Huang, Marcelo Kallmann, Teenie Matlock and Justin L. Matthews. *Gesture Variants and Cognitive Constraints for Interactive Virtual Reality Training Systems.* In Proceedings of the International Conference on Intelligent User Interfaces (IUI), Palo Alto, CA, 2011.

Yazhou Huang, Mentar Mahmudi and Marcelo Kallmann. *Planning Humanlike Actions in Blending Spaces.* In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), San Francisco, CA, 2011.

Andrew (Wei-Wen) Feng, Yazhou Huang, Yuyu Xu and Ari Shapiro. *Automating the Transfer of a Generic Set of Behaviors Onto a Virtual Character.* In Proceedings of the 5th International Conference on Motion In Games (MIG), Rennes, France, 2012.

Yazhou Huang and Marcelo Kallmann. *Interactive Demonstration of Pointing Gestures for Virtual Trainers.* In Proceedings of 13th International Conference on Human-Computer Interaction (HCII), San Diego, CA, 2009.

Marcelo Kallmann, Yazhou Huang and Robert Backman. *A Skill-Based Motion Planning Framework for Humanoids.* In Proceedings of the International Conference on Robotics and Automation (ICRA), Anchorage, Alaska, 2010.

Yazhou Huang, Robert Backman and Marcelo Kallmann. *Walking Gait Generation with Foot Placement Control for the HOAP-3 Humanoid Robot.* University of California Merced School of Engineering Technical Report 2011-02, 2011.

CHAPTER 1

Introduction

1.1 Background

The notion of animation, or specifically keyframing, dates back to more than a thousand years ago when ancient Chinese zoetrope-type device was first invented to produce the illusion of motion from a rapid succession of static pictures. We are all familiar with the various forms of animation, from the traditional hand-drawn picture frames in the form of a flip book, to the technology of stop motion or claymation. It has not changed much over all this time, until the introduction of Computer Graphics came along.

Computer Graphics in general refers to the modeling and rendering of 3D objects, including the sub-field of methods for digitally synthesizing the animation of objects. There has been a huge increase in the interest and development in computer graphics, which had, and is still having, profound impact on many types of media around us. It has truly revolutionized the traditional movie, animation and the video game industry by bringing virtual objects to life. It has evolved from the simple 2D scenes in the early arcade games, to the complex 3D structures which redefine our perception for photorealism. A major part of the animation we have seen is, not surprisingly, about ourselves, known as the character animation. It has already been widely applied in many fields, such as computer-aided design (CAD) with ergonomic human factors [Sie80], social networking [Lin03], virtual training in sports [Mot98], oil & gas industry [EON09], medical applications [For98], and so on.

The notion of character animation can be largely divided into skeletal animation (particularly for vertebrates) and geometry deformation animation (blend shape, Free-Form Deformation (FFD) and so on). Skeletal animation relies on the representation of character's skin mesh binded onto a bone hierarchy underneath, known as skeleton or rig. Movement of the skeleton translates into the movement of skin mesh vertices. For blend shape animation, a neutral mesh shape is first created, based on which a series of target blend shapes are pre-deformed. The interpolation between neutral and target shapes generates new shapes. This technique has been widely used for cloth and facial expression. And for the FFD modifier, the geometry is surrounded with a lattice, and by adjusting the control points of the lattice, the enclosed geometry is deformed accordingly. This dissertation is focused only on skeletal animation.

To date, there are mainly three popular categories of character animation. First category is digital key-framing and hand driven, a process similar to the original animation creation but with the help of 3D modeling tools. It is still a very labor-intensive task

that requires artistic talent and training.

The second category is data driven with motion capture. The technology of motion capture emerged in the 1980s. It provides the ability to record the performance of a human actor/actress in 3D space, and then map onto virtual characters without losing the fidelity. In the past decades, motion capture has gone from magnetic field sensing [Asc91] and exoskeleton [Ani97], to optical marker tracking [Vic95] and inertia sensing [Xse07], and to marker-less tracking that are either video camera based [AST08, Org] or depth camera based [Mic10] designed for game consoles, making this technology more popular and readily available than ever before.

The third category is algorithmic and model-driven approaches. One example is the dynamic physics-based model, like reverse pendulum, contact frictions and joint torque control, to list a few. Great achievements has been made, such as rag doll simulation [Nat05], however the complexity of human structures is a huge obstacle in the development of fully dynamic motion synthesis.

1.2 Problem Statement

People in computer graphics and robotics are familiar with the uncanny valley phenomenon. There is an important view [Goe03] on the "uncanny valley" [Mor70] I very much agree with, which says that in essence what triggers the brain to react is when the human-like appearance of the android and its robotic motion "mismatch". For example, cumbersome movement from ASIMO [Hon00] and BigDog [Bos05] are well perceived by the public because what's behind such movement are their matching robotic appearances. However, this is not the case for computer graphics. The biggest challenge is that the substantial advances in modeling and rendering, with the improvement on modern graphics hardware at exponential rate [Moo65], has created a "mismatch" between the physical appearance of a simulated character (skin, hair, etc) verses the animation aspects (body, facial expression, lip syncing, etc), evidently causing perceptual conflicts and pushing the industry further into the uncanny valley.

Motion capture has been widely used in the movie and game industry to achieve maximum realism and fidelity and is seen as the Holy Grail in modern character animation, even though it usually requires intensive man power for clean up and post-processing. Recent research and development has targeted automating and reducing the humans factors from the loop. However this process is not as straightforward as it sounds. The problem, I believe, is that the characteristics of what is perceived as human-like can not be easily translated into computer algorithms and program codes. As an example, different styles of locomotion exist from one person to another, yet they are all perceived as human-like. Clearly the criterion for naturalness of character motion synthesis is not a closed-form solution like the minimal energy consumption (otherwise only one optimal solution exists), but rather involves the aspects of aesthetics, elegance and individual styles, all of which are constituents of the "realism". Even though this work is focused on skeletal animation, the realism in this context refers to

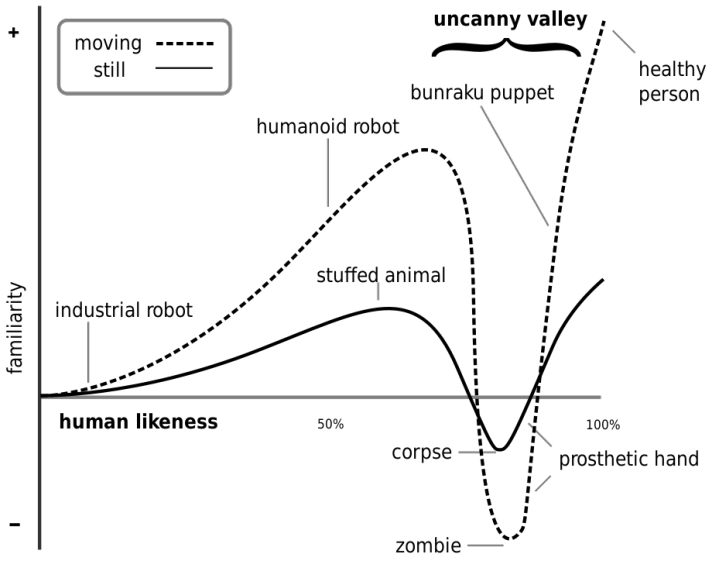


Figure 1.1: According to Mori, the uncanny valley is the region of negative emotional response towards robots that seem "almost human". Image taken from a recent reprint [MMK12] of his original work [Mor70]. Also according to Goetz [Goe03], movement amplifies this emotional response, so does the "mismatch".

the naturalness and humanlikeness of the overall synthesis result including aspects of skinning, rendering, motion dataset which may came from motion retargeting, locomotion and action synthesis, whole-body coordination, behavior modeling, etc.

My experience working on the animation pipeline with a small art team at ICT makes me believe that motion capture is certainly not going away anytime soon. However, this is not to say that high quality animations must rely on motion capture and animation artists. In fact, over the past years many advances based on motion capture have been made, concerning slight but precise modifications of an original motion or the parameterization of large motion databases. And I agree with Michael Gleicher, creator of the original motion graph, that the future of technology for animated characters in interactive systems lies as a hybrid of synthesis-by-example and algorithmic approaches [Gle08]. However it still poses great challenges in achieving realistic synthesis with interactive motion control.

So the problem being addressed in this dissertation can be summarized as how to achieve a character animation system that combines realism, flexibility, precise control and ability to adapt to the cluttered environments. Additionally, the system needs to be connected to higher-level constraints with task-oriented specifications, and needs to produce real-time synthesis within interactive applications.

1.3 Objectives and Approaches

This dissertation aims at creating an autonomous human-like character that is able to demonstrate, instruct and deliver training information in a virtual environment. The motion synthesis and interaction models are carefully developed according to the following objectives:

- precise parametrization for specified constraints;
- rely on motion capture data for high-fidelity synthesis;
- employ real-time algorithms for fast responsiveness;
- target at training applications on an interactive platform;
- employ randomized planner for adaptability to unknown environments;
- behavior modeling, use real observations as ground truth;
- easy motion retargeting for arbitrary character models.

The approaches presented in this dissertation to achieve these objectives are briefly summarized as follows. The first part of this thesis addresses motion synthesis with a simple example-based motion parameterization algorithm for satisfying generic spatial constraints at interactive frame rates.

The proposed motion parametrization approach is to blend a set of consistent time-aligned example motion sequences. To achieve precise motion control, blending weights are directly optimized until the specified constraints are best met. For example, to synthesize a pointing gesture for the character to pin-point at given target object, the algorithm finds the best set of bending weights so that the end-effector (finger tip) precisely touches the object at certain stroke times. This process in general takes a few milliseconds depending on the dataset and the complexity of the character's joint hierarchy, suitable for real-time applications. Also since motion blending is known for preserving the humanlikeness aspects from the example motions, it meets the objective for high-fidelity.

This motion parametrization approach has been integrated on an interactive motion modeling platform. The experts could easily build the training motion dataset via direct demonstration inside the virtual environment, based on which the new motions could be synthesized for teaching and training purposes for novel users, effectively facilitates the virtual training applications. A wearable motion capture vest is also described as a low-cost motion recording device and also part of the immersive training platform.

In order to generate collision-free actions motions in a cluttered environment, the motion parametrization approach is integrated with randomized planning. Humanlikeness is hard to achieve with the traditional sampling-based approaches inside the configuration space. For this reason, a bi-directional time-synchronized sampling-based

planner is proposed that explores the weight-time blending space of example motions. This allows the final synthesis to have the flexibility going around obstacles while maintaining the original quality of the example motions. Coupled with a locomotion planner, it generates whole-body motions in cluttered environments, with the realism from motion capture examples and flexibilities from sampling-based planning.

Further more, based on the data collected from human participants, we proposed a whole-body planner that attempts to address high-level coordinations for demonstrative tasks. The planner breaks down the overall planning problem into coordinated body positioning, locomotion, action execution and gaze synthesis, each solved with a sub-module. Certain parameters are modeled using data collected from human participants. The final goal is to achieve autonomous virtual agents engage and demonstrate to observers in various scenarios.

Finally to enable easy transfer of motion datasets onto arbitrary character models with different joint names and hierarchies, we describe a motion retargeting solution that uses heuristics to find symmetries and key joint names in character skeletons, achieving automatic joint name matching that requires little to none human input.

1.4 Structure of dissertation

This dissertation presents several motion synthesis approaches in the topic of skeletal character animation, organized as four parts.

Part I is dedicated to motion capture based motion parametrization. Chapter 3 presents a real-time motion synthesis algorithm *Inverse Blending* designed for meeting generic spatial constraints at certain frames or stroke points. Chapter 4 is a detailed analysis over 4 motion synthesis algorithms, which measures their performance to better understand the advantage of each method. Chapter 5 presents an immersive 3D motion modeling platform designed to easily program generic gestures and actions for virtual agents via direct demonstration.

In an attempt to achieve human-like motion planning, Part II (Chapter 6) introduces a novel concept of *Blending Space*, and presents a planner solution that combines *Inverse Blending* with bi-directional randomized search to quickly explore collision-free natural-looking postures in the workspace.

Part III (Chapter 7) focuses on simulating interactions between a demonstrator and observers inside the virtual demonstrative scenarios. A behavior-level planner *PLACE* is proposed that takes into account relevant aspects including body-positioning, locomotion, action, coordination and gaze in an unified way.

To complete the character animation pipeline, a wearable motion capture setup *GestureVest* for data acquisition is presented in Chapter 9 as an addition to the on-line motion modeling platform. And to facilitate the motion data transfer, a fully automated motion retargeting approach is described in Chapter 8, which handles various commonly seen skeleton joint hierarchies and naming conventions.

CHAPTER 2

Literature Review

This chapter reviews the state of the art in skeletal character animation related researches. Topics include kinematic and physics models, motion blending and parametrization, motion planning, and behavior modeling.

2.1 Kinematic-based Models

In general, kinematics is defined as the branch of classical mechanics that describes the motion of points and groups of objects. In computer graphics, it refers to simulating the movement of the skeleton and rigid bodies that attached to it, without using dynamic models.

2.1.1 General Kinematic-based Approaches

There have been many published works using kinematic models to generate upper-body motions or locomotion sequences based on a set of controllers primarily driven through various procedurally-based algorithms. For example, Kallmann [Kal08] presented a whole-body analytical inverse kinematics (IK) method integrating collision avoidance and customizable body control for animating reaching tasks, with a new simple search method for achieving postures avoiding joint limits and collisions. Sun et al. [SM01] addressed the problem of gait generation with three modular components, including a low-level gait generator based on sagittal elevation angle control, an inverse motion synthesis method based on barycentric interpolation to handle uneven terrains, and a path following module. While these methods are fast, accurate, simple to implement, and in many cases hard to find other alternatives, they may appear robotic as certain human-like characteristics are missing.

On the other hand, kinematic-based models play an important role in post processing motion capture data to impose minor modifications and constraints. One large class of related work involves retargeting motion capture data onto characters with different sizes. [HRE08] introduces a novel way to record animations in a morphology independent form, preserving both structural relationships and its stylistic information. At runtime the system uses a robust IK solver to animate characters with highly varying skeleton morphologies completely unknown when the animation was created. [HKT10] introduced the interaction mesh, a structure to represent implicit spatial relationships between body parts and surrounding objects. Inter-penetrations were reduced by mini-

mizing the local mesh deformations within animation frames, and the algorithm can be applied in real-time to motions with various kinds of close interactions.

Other popular approaches include using IK to match end-effector goals specified from the raw capture. [LS99] combines a hierarchical multilevel B-spline fitting with analytical IK that aims at solving the animation problems formulated as space-time constraints. The fitted curve is used to interpolate the joint displacements that are smoothly propagated to neighboring frames, then IK adjustments are done to meet the constraints in each frame. To fix the foot-skating problem commonly seen in motion retargeting, a simple but efficient foot-skate cleanup method [KSG02] was proposed using a modified single-limb IK algorithm. A combination of knee-damping and ankle-to-hip length adjustment is applied to fix the "knee pop" and generate smooth result without toe-floor penetration.

2.1.2 Motion Blending

As the technology for motion capture becomes more readily available, a great amount of research has been conducted on how to re-use the captured data, without introducing too much changes or modifications, to generate new motion sequences. Kovar et al. [KGP02] pioneered the work of motion graph, which is an auto-constructed interconnected graph consisting both the pieces of original motion and automatically generated transitions. New motion sequences, such as certain styles of locomotion along arbitrary paths, are generated piece-by-piece by simply "walk" along the graph nodes and select those with minimal errors. Improvements were made in [HG07] by introducing the new data structure of parametric motion graph, which is capable of dynamically generating the transitions in real-time. The graph is structured according to particular motion types. Accurate streams of high-fidelity controllable sequences were generated through blending-based parametric synthesis.

To address the problem of spatial constraints with motion graph, [SH07] introduced the interpolation of two time-scaled paths in the graph. It was coupled with A^* search to find global near-optimal solutions consisting a variety of concatenated behaviors with more flexibilities, however the search space is still non-continuous. The Inverse Branch Kinematics (IBK) method introduced in [MK11] aims at deforming the 2D graph branch with Cyclic Coordinate Descent (CCD) solver, so that the branch end reaches the goal with much better precision, offering more flexibility and parameterization to motion graphs without degrading synthesis quality. However, even with the IBK correction, it remains difficult for Motion Graph based methods to enforce higher DOF constraints, for example, to have the character arrive precise at certain 2D position with certain exact body orientation as specified (a total of 3-DOFs constraints).

Motion graphs could easily become bigger by including variations of motion types and styles. In order to handle large motion capture datasets with redundancies, [KG04] provided a "match web" metric for identifying similarities in large datasets, and then use them to build a continuous parameterization space through dense sampling and

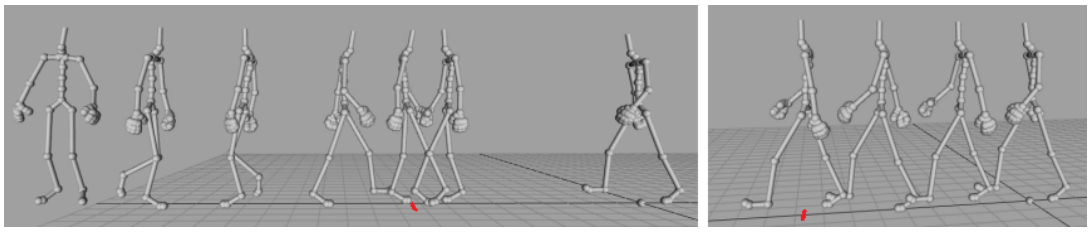


Figure 2.1: An example showing a typical problematic solution from traditional motion graph on left where the character could not arrive exactly at target position (marked red) and had to do a zig-zag. IBK method [MK11] on right better solves this problem.

k-nearest-neighbors (KNN) interpolations. [ZNK09] proposed an iterative sub-graph algorithm to better build motion graphs that are small, fast, and with good connectivity as well as smooth transitions. The algorithm efficiently finds optimal solutions by searching for a minimal-sized subgraph within the larger graph. While these motion graph related methods excel in maintaining the originality by imposing less modification to the original motion, they lack the flexibility of parametrization, especially towards constraints with higher Degrees of Freedom (DOFs).

2.1.3 Motion Parametrization

This section is focused on the topic of continuous motion parametrization, which is the first main topic of this dissertation. The reviews include motion capture based methods that incorporate higher-level characteristics with user defined parametrizations.

A number of earlier studies have found it beneficial to look into the frequency space. Unuma et al. [UAT95] uses Fourier analysis to interpolates, extrapolates motion data as well as to alter their styles. Bruderlin and Williams [BW95] apply a number of different signal processing techniques to motion data, including motion filtering, multitarget motion interpolation and waveshaping (set "soft" joint limit), to facilitate the reuse and adaptation of existing motion data on various parametrization levels. [WH97] proposed stochastic re-sampling inside the parameter space, providing an efficient way to expand the range of possible motions by "mixing" them to the exact specifications. Lee and Shin [LS01] developed a multi-resolution analysis method for motion editing. The data is represented as a hierarchy of coefficients that store the global pattern all the way to details at finer resolutions, which helps coordinating the invariance for motion smoothing, blending, and stitching. Based on the motion correlations of articulated figures, [PB02] allows the creation of animations through texturing, i.e. sketching a small number of keyframes on a low-DOF space, which is then predicted and enhanced with motion capture data. The method relies on frequency analysis to segment the data, searches for paths that maximize the use of consecutive fragments, then joins them with smoothing. The limitation of these approaches is that they do not aim at creating a completely automatic method, but rather as an additional tool for animators to incorporate the motion capture data into their own creations.

As a different approach aside from signal processing techniques, Rose et al. [RBC98] proposed the motion interpolation scheme combining the Radial Basis Function (RBF) with polynomial terms, and used it to generate new synthesis with examples characterized by emotional expressiveness or control behaviors. The result tends to degrade when the desired input gets further from the original data because the synthesis would rely more on the linear approximation terms.

More generically, spatial properties such as end-effector placement and feet sliding are better addressed by Mukai et al. [MK05] with a geostatistic model. It optimizes the interpolation kernels to estimate the correlations inside the parametric space, thus predicts motions with more accuracy and less undesirable artifacts, alleviating the problem of spatial inconsistencies. Another approach for improving the maintenance of spatial constraints is to adaptively add pseudo-examples, as shown in [RSC01, KG04]. As an extension to automatically correct the non-linearities [RSC01], pseudo-examples were derived from the artist-designed initial interpolation space and then used to produce smooth and more accurate results. However the error between the end-effector and specified target still exists.

Below is an illustration of (a) the original RBF interpolation, (b) improved RBF with pseudo-examples, and (c) geostatistic interpolation. Although geostatistic model achieves better precision at the character's end-effector compared to RBF interpolation, errors still cannot be eliminated.

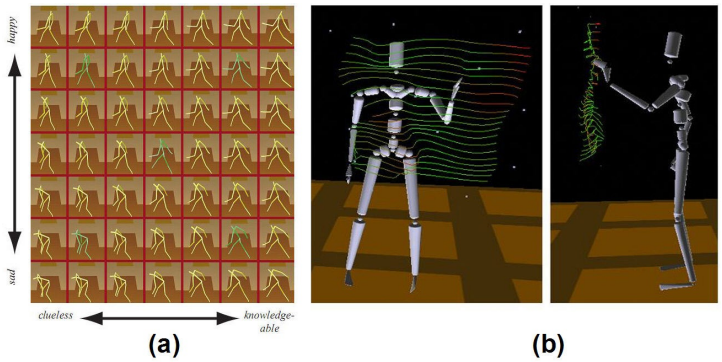


Figure 2.2: (a): the original RBF interpolation [RBC98]; (b): improved RBF with pseudo-examples [RSC01].

Although potentially the pseudo-examples could better cover the continuous space of the constraint, these random sampling approaches however requires significant computation and storage in order to meet constraints accurately, and it would be difficult to handling several constraints at one time. In these previous methods, spatial constraints are only handled as part of the motion parametrization approach employed. Typically, examples motion that are close to the desired constraints are chosen, then used in an abstract interpolation space to obtain motion variations as well as to satisfy the constraints. Another possible technique sometimes employed is to apply IK solvers in addition to blending [RSC01, CHP07], however risking to penalize the obtained realism.

Other statistical methods were also applied in the animation researches. For example, [BH00] applied Hidden Markov Models (HMM) in learning motion patterns from highly varied motion sequences. After the common choreographic elements were identified, new choreography can be synthesized in many styles, such as converting a novice ballet into graceful modern dancing. Principal Component Analysis (PCA) is another popular statistical method that can be found in several works. In [BLC02], a PCA-based technique was presented to track the motion from traditionally animated cartoons and retarget it onto 3-D models and 2-D figurers, producing expressive and exaggerated new sequences. [EMM04] created new idle motions using a PCA-based animation approach to combine layers of posture variations and balance changes, effectively avoiding unnatural repetitions in the results. [SHP04] uses PCA to find a low-dimensional space that captures the properties of the desired behavior, and then solves optimization problems within this subspace. The challenge is that postures mapped from the low-dimensional space consists only the joint angles (translations are usually discarded for PCA analysis) and thus may not handle very well the positional aspects of the animations. Also the jitters in-between postures are hard to eliminates after remapping onto the original space, due to the loss of high frequency information in the dimension reduction process.

Since the introduction of Gaussian process latent variable model (GPLVM) by Lawrence et al. [Law03], it has gained popularity in the animation community and has been used in several research works. This is a probabilistic model for principal component analysis (PCA) that could map from a latent space to the observed data-space using a particular Gaussian process. [GMH04] extends this original model by introducing the Scaled-GPLVM model that is trained using human poses with various styles. This model was used to build an IK system capable of producing the most likely poses satisfying given constraints, but prefers the poses that are most similar to the space of poses in the training data. Similarly, [LWH12] presents a technique that interactively animates characters performing user-specified tasks with constraints on a low-dimensional latent space.

Due to the nature of the Gaussian process, these statistical models could generate smooth postures when mapping back into the original data space. It is also a plus to be able to better extrapolate the data with less requirements on the dataset consistency. However, the higher computational cost in the training process imposes limitations on applications that involve on-line appending or refining of the dataset. Also it remains challenging to address constraints with higher DOFs using these methods, especially considering the computational complexity of solving optimization problems formulated based on IK.

2.1.4 Motion Planning

Motion planning is a well studied topic in the robotics domain. It provides an effective way to explore the unknown environment and solve complex tasks. Manipulation planning was applied to computer animation as early as [KKK94] to computes

the collision-free arm trajectories for manipulating movable objects and re-grasping them. Traditional motion planning methods [Lat90, Lau98, LaV06] are based on the systematic search of configuration spaces. Among the several techniques, sampling-based methods [KSL96, LaV98, KL00b] have become extremely popular for planning in continuous configuration spaces. Such methods are also popular for planning humanoid motions, in particular for planning footsteps [KNK03, CLC05, CNK07] and reaching motions [KKK94, KKN03, BKD06, DN06, DK08].

[YKH04] combines path planning with the domain knowledge in synthesizing full-body motions. A path planner searches for a motion with the corresponding poses satisfy geometric, kinematic, and posture constraints. A constraint-based IK solver resolves the redundancies and generates solutions toward natural-looking poses extracted from captures. The path is then converted to motion trajectories using velocity profile, to create the object manipulation sequences. Generally speaking, the complexity of these kinematic models of the characters or humanoid robots used in the planning algorithms may lead to longer planning times. Additionally the results generally look robotic and lack of naturalness, even after the smoothing process.

In contrast, methods originated from the computer animation domain focus on achieving humanlike results using motion capture data, without much importance given to searching for collision-free motions in complex environments. Probably the most popular approach for computing realistic full-body motions is to extract motions from a motion graph structure. Motion graphs are built by connecting the frames of high similarity in a database of motion capture examples [KGP02, AF02, LCR02, PB02, LWS02, Saf06]. More reviews covering this topic is presented in the previous section. The main drawback of motion graphs is that, as the motion variations needed to satisfy many constraints go up, such as around obstacles and addressing precise placements of end-effectors, the size of the graph structure would grow exponentially, which poses great challenges in the storage as well as maintenance.

Planning methods have been integrated with motion capture data in many ways. For instance, Lau and Kuffner [LK05] plan over a behavior-based finite state machine (FSM) of motions [LK06], Choi et al. [CL03] combines probabilistic path planning with hierarchical displacement mapping. A roadmap is constructed from randomly sampled valid footholds, with graph edges being motion clips. The roadmap guides the locomotion and generates motion sequences, and finally adapts the sequence to constraints specified by the footprints. Aiming at planning natural-looking locomotion, many other planning methods have been proposed for synthesizing full-body motions among obstacles [KAA03, EAP06a, KL00a, LK06, PZL10]. However, less attention has been given to planning generic upper-body actions in a continuous space in coordination with locomotion. The continuous space is specifically mentioned here for its ability to search and compute more complex solutions.

Traditional planning algorithms have also been coupled with dynamic models for more natural-looking simulations. As an example, [KNK03] adapt existing randomized path planner by imposing balance constraints on incremental search motions to main-

tain dynamic stability of the final path. This was done by constraining the zero moment point (ZMP) trajectory and transforming statically stable paths into dynamically stable ones. More review of the physics-based models and their limitations are presented in the next section.

To better addresses the whole-body motion planning problem, it was a natural solution to model the upper-body gesture and action motions independent of the stepping motions, which are needed for achieving precise locomotion around the target objects being demonstrated or manipulated. This is known as the concept of planning with motor primitives, an important topic in motion planning. The basis of this approach is that complex motions can be achieved through a combination of simpler primitive motion skills [Arb81], which is supported by evidence found in cognitive studies [TS00]. Many works in the robotic domain have studied the problem of planning with motion primitives, such as [HBH08, LaV06, KHB10]. In recent years it also gained popularity in character animation [PKL08]. To investigate the problem of humanoid agents reaching to arbitrary targets in an environments with obstacles, [DKM04] tests the popular algorithms for single-arm reaching, i.e. randomized motion planning or constructing uninformed trajectories in operational space. The efficiency of the reaching algorithms are analyzed in both static and dynamic environments.

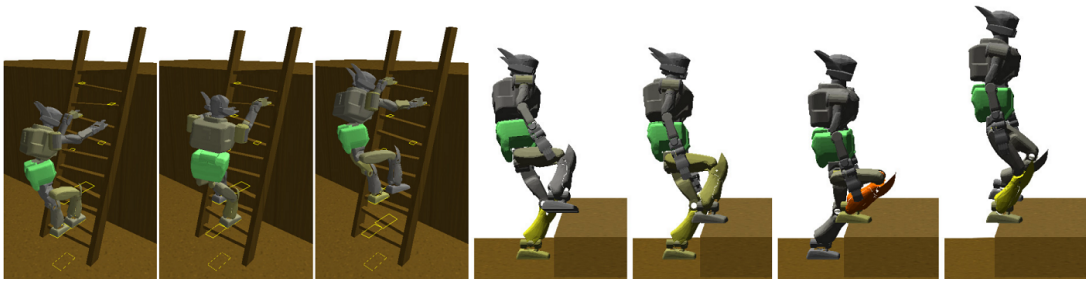


Figure 2.3: An illustration of motion primitives developed for a humanoid robot [HBH08]: climb ladder and stairs.

To alleviate the bottle-neck of randomized planning algorithms, [LK06] proposed to pre-compute search trees of motion clips for arbitrary environments. Then, solution paths and motion sequences are efficiently extracted using a lookup table. For distant goals, rough path of intermediate sub-goals are generated to guide further lookups. Similarly [JK07] proposed an attractor guided planner that extracts significant attractor points from successful paths, and reuse them as guiding landmarks that can effectively avoid planning from scratch and unnecessary re-planning. In an attempt to speed-up the planning with primitives, [CCL10] proposed using basic (small) motion primitives as atomic actions, and coupled with fast-to-compute heuristics and an Anytime Repairing A^* (ARA^*) search. It was shown in the experimental analysis that the planner could generate consistent, low-cost motions faster with guarantees on completeness and bounds on sub-optimality. However the downside of planning with motion primitive it that, without an overall coordination of the primitives, the synthesis results may still appear robotic and less humanlike.

2.2 Physics-based Models

We all live in this physical world and all of our movements are subject to the physics laws. Hence it is a natural strategy to apply these physics laws onto animated characters.

Shapiro and Faloutsos [SPF03] proposed the DANCE framework for animating interactive characters by combining kinematic animation with physical simulation. Transition methods between kinematic control and dynamic control were developed to leverage the advantages of both techniques in a unified framework. In [YLP07], control strategies SIMBICON were created from motion capture with a small number of parameters, and the system is capable of generating various gaits and styles in real-time including walking, running, skipping and hopping. However, fully dynamic control of an articulated character still remains a very difficult problem.

Figure 2.4: Kinematically controlled kick and dynamically controlled reaction and interaction [SPF03].

More recently, motion capture data was coupled with physics-based approaches to achieve better realism. Zordan et al. [ZH02] proposed synthesizing reactive human motions by combining dynamic simulation and motion capture data. The system tracks the trajectories to follow motion capture data, maintains character's balance, while modifies sequences from a motion library to accomplish specified tasks like punching or racket swinging. Similar approaches have been used to help determine the best plausible re-entry into motion library playback following unexpected impacts and changes towards the character [ZMC05], or to track multiple trajectories in parallel and blend or transition between different path controllers at arbitrary times for locomotion synthesis [MPP11].

Physics models are also used for gesture synthesis because dynamics has a lot to offer in facilitating the subtleties found in the gesture motions. Neff et al. [NF02] introduced tension and relaxation into the posture based animation by separating stiffness control from position control, thus achieving better control in posture interpolation and dynamics efficiently.

In recent years, more attentions have been given to optimizing complex physics-based models and control loops. As an example, [MZS09] presented a hybrid inverse/forward dynamics algorithm that determines joint torques to track a reference motion while responding to external perturbations through an optimization routine. Also in [JYL09], a physics-based approach is presented to solves a constrained optimization problem formulated at every time step. Active control strategies are specified using intuitive kinematic goals, enabling the synthesis of virtual characters interacting with

dynamic environments.

To summarize, in general it still remains difficult to generate a wide variety of realistic motor behaviors using purely physics-based methods alone. Good results were achieved by incorporating captured datasets as references. However, without motion capture in the loop, the result of dynamic simulation still has a hint of rigidness and robot-like.

2.3 Behavior Modeling

In general, behaviors refer to the range of actions made as the response of the system or organism toward various stimuli or inputs in conjunction with the environment, such as conversation, social reactions with other parties, emotions, changes in position, orientation or relations of persons, etc. In my opinion, similar to what was proposed in [Dil98], the behavior modeling can be largely divided into two major categories: implicit modeling and explicit modeling. Implicit modeling is primarily a black-box with rules that are less intuitive, and output is an inductive process of the stimulations taken in, "doing what you know"; whereas explicit modeling is essentially a deductive process, patterns or perceptions are perceived and are then put into practice, "know what you are doing". Both processes could generate effective simulations, the difference is that the perceived patterns or perceptions need to be explicitly modeled in the later case.

There have been a great amount of attention on the gesture modeling researches. For implicit behavior models, [LKT10] presents an optimal-policy gesture controller that schedules gesture animations in real time based on acoustic features in the user's speech. This process is done through an inference layer that learn the hidden associations between body language style and speech acoustic features, and also a control layer that uses reinforcement learning to select the optimal motion accordingly. To animate an avatar in real-time with a large repertoire of behaviors, [LCR02] presents a framework that first construct a graph structure identifying plausible transitions between motion segments, then allows the user to control the avatar with a low-dimensional input device.

For explicit behavior modeling, [KKM06] describes Behavior Markup Language (BML) within a framework to unify a multi-modal behavior generation. As an efficient way for motion synthesis, BML stores sync points, conditions, behavior elements and entries for gesture, face, gaze and speech elements. SmartBody [TMM08] is the open-source modular framework referred here for animating embodied conversational agents, through a flexible combination of real-time animation approaches.

[KNK07] presents an algorithm that exploits the concept of gesture units to make gestures produced by the nonverbal behavior a continuous flow of movements. A data-driven approach synthesizes the hand and arm gestures by recreating gestural behavior in the style of a human performer. Similarly, [LK05] explores a behavior planning approach for automatic motion generations. Motion clips are abstracted as high-level

Behavior	Example	BML Command
Gaze	look at the object called 'table'	`<gaze target="table"/>`
Locomotion	move to location (10, 75)	`<locomotion target="10 75"/>`
Head Movement	nod your head	`<head type="NOD"/>`
Idle	assume an idle posture called 'idling_motion1'	`<body posture="idling_motion1'/>`
Animation	play an animation called 'dosomething'	`<animation name="dosomething"/>`
Gesture	point at character1	`<gesture type="POINT" target="character1"/>`
Reach	Grab the object 'cup'	`<sbm:reach target="cup"/>`
Constraint	Constraint your hand to 'ball'	`<sbm:constraint target="ball"/>`

Figure 2.5: A quick BML reference table for the modular animation framework SmartBody.

behaviors and associated with a behavior FSM that defines its capabilities. At runtime, motion is generated by planning on the FSM with global search of behavior sequences with heuristic cost functions, to reach user-designated goal positions. [NKA08] produces full-body gesture animation for given input text in the style of a particular performer. A video of a person's particular gesturing style is first annotated and tagged with theme, rheme and focus. Then a statistical model creates the gesture script specifying stream of continuous gestures coordinated with speech, enhancing the gesture descriptions with additional details.

Apart from gesture oriented behavior modeling, crowd simulation represents an important aspect of group behavior modeling. Thalmann et al. [TMK99] identifies the essential mechanisms for implementing virtual crowd, and introduces the autonomy distribution among agents, groups and objects. An abstraction for specific behavior were used to simulate large crowds in complex environment involving human agents, groups of agents, and interactive objects. [ST05] integrates motor, perceptual, behavioral, and cognitive components into a model of pedestrians as individuals, and demonstrates scenarios in a large urban environment. Inside the agents could plan their actions either on local scales or global ones. While various behaviors are simulated in crowd simulation such as following and fleeing, detailed agent-agent interactions are typically not modeled.

Throughout the literature of behavior modeling, much less attentions have been given to the territorial modeling, i.e. the science of territoriality. Interestingly, the concept of human territories has been proposed much before the formation of modern computer graphics, by Scheflen and Ashcraft [SA76], Adam Kendon [KHK75], and Allan Pease [PK81]. The book of Scheflen et al. [SA76] defines territoriality as the a way of looking at human behavior, and the study of human territoriality is the study

of human behavior. It gives us a thorough illustration of human territories, the many units of space-time people form and use. Two important concepts were introduced in the book, 1) orientational field, an inter-dependent configuration in space and time consisting of all of the simultaneous and sequential orientation relations that occur within all the relationships that make up that human activity or event. For example, orientational field between a group of people conducting a conversation refers to the mutual body orientation: between hands, between speech and listening, and between gazes. 2) territorial field, a customary, ordered pattern of positions, relationships, and clusters. including element (a row or arc persons holding a common orientation), face formation (the spatial patterns formed during face-to-face interactions), gathering (cluster of multiple elements and face formations), hub (two or three concentric rows of people who face a common nucleus of people facing them) and intersected formation (an array of people subdivided into separate formations).

Although these territoriality concept were proposed several decades ago, close-up behavior modeling between smaller social groups such as scenarios in [Ken76, Ken90a] has been given less attentions in the literature of character animation. The following work were specifically designed to simulate these scenarios with explicit behavior models. Pedica and Vilhjálmsson incorporated artificial intelligence (AI) rules [PV08] and behavior trees for modeling the human territorial behaviors within social interactions, such as conversation, gatherings, standing in line, etc. The model monitors the surrounding and signals social and spatial reactions. Combined with steering, the system was able to simulate group dynamics during social interactions with high efficiency. As an extension, [PV09] addressed the human territorial behaviors during social interactions with AI models, bringing the module of territoriality which often gets separated from simulation under the social setting. [CGV11] examined the case where people select seating places inside café and restaurant. A model for specific behavioral pattern was implemented, and notable effectiveness was observed in the simulated behavior of individuals and groups. However, without detailed motion planning, coordination and synthesizing, these systems still cannot simulate close interactions between agents with desired fidelity and capability.

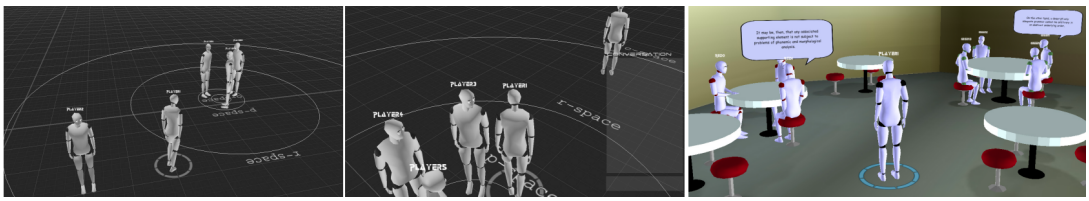


Figure 2.6: Modeling the Territorial behaviors. Left [PV09]: agent joins a conversation group then leaves. Right [CGV11]: seating behavior in public places.

As for modeling the close encounters between two agents during locomotion, Pettré et al. [POO09] investigated the interactions between two humans walking along converging paths and avoid colliding into one another. A model elaborated from experimental data handles such cases for virtual agents, which was calibrated with real data.

Even though this work is only focused on collision avoidance for agent locomotion, it could be seen as one step ahead in better simulating agent-agent reactions in close proximity.

In face-to-face communications such as conversations, the speaker continuously checks if the listener is engaged by monitoring the eye-gaze behaviors. As a critical embodiment for conversational engagement [NI10], gaze behavior has been intensively studied through a large body of research. Some of this neurological research focuses on the nature of eye movements, including saccades (ballistic eye movements in a visual scene) [PPU88, VSC08]. Some studies [LBR92] closely examine vestibulo-ocular reflex (VOR) in saccadic and slow phase components of gaze shifts. Additional studies [CHB04, GG92] involve fine-grained analysis of small and large gaze shifts where classic feedback loops are used to model the coupling and dynamics of eye and head-orienting movements. Gaze has been used in computer graphics from gaze-contingent real-time level of detail (LOD) rendering [MDT09] to the modeling of movement for eyes balls, eye lids and related facial expressions [DLN05]. Gaze direction in particular is known to help with basic two-way communication because it can help a speaker direct attention and disambiguate for a listener [Ken90b]. Gaze direction has also been shown to help human listeners better memorize and recall information in interactions, including robot storytellers [MHF06] or narrative virtual agents [BWA10]. In [TLM09, LM07], emotion models were introduced with body posture control to make synthesized gaze emotionally expressive. These systems typically use pre-recorded voice coupled with simulated gaze to interact with the listener. The simulated agent in general remains at the same spot facing the audience, without the need for locomotion. Less attention has been given in modeling the higher-level gaze and gaze related behaviors such as body positioning, coordination with body gestures, etc, with the purpose of delivering information to a human observer at various locations.

Finally, to revisit the "mismatch" aspects that caused the uncanny valley effect mentioned in the introduction section, research [EMO10] shows that humans are more sensitive to visual desynchronization of body motions, than to mismatches between the characters' gestures and their voices. Furthermore, the interactions modeled between conversational agents (talker and listener) contributes more to the realism of the synthesis, surpassing the body motion desynchronization or mismatched audio. This again proves that in order to generate more natural and humanlike character motions, many aspects of the synthesis must be taken into account, including motion retargeting, locomotion and action synthesis, whole-body coordination, behavior modeling, etc. And by incorporating the advantages of various algorithms and approaches covered in this literature review, this dissertation aims to achieve a character animation system that combines realism, flexibility, precise control and environment adaptability with higher-level constraints and task-oriented specifications.

CHAPTER 3

Motion Parametrization

Motion blending is a popular motion synthesis technique which interpolates similar motion examples according to blending weighs parameterizing high-level characteristics of interest. We present in this chapter an optimization framework for determining blending weights that are able to produce motions precisely satisfying multiple given spatial constraints. Our proposed method is simpler than previous approaches, and yet it can quickly achieve locally optimal solutions without pre-processing of basis functions. The effectiveness of our method is demonstrated in solving two classes of problems: 1) for precise control of end-effectors during the execution of diverse upper-body actions, and 2) for walk animation synthesis with precise feet placements, demonstrating the ability to simultaneously meet multiple constraints and at different frames. Our several experimental results demonstrate that the proposed optimization approach is simple to implement and effectively achieves realistic results with precise motion control.

3.1 Motion Parametrization with Inverse Blending

Keyframe animation and motion capture represent popular approaches for achieving high-quality character animation in interactive applications such as in 3D computer games and virtual reality systems. In particular, motion blending techniques [RBC98, RSC01, MK05] provide powerful interpolation approaches for the parameterization of pre-defined example animations according to high-level characteristics. While direct blending of motions is able to produce fast and realistic motions, it remains difficult to achieve blendings and parameterizations able to precisely satisfy generic spatial constraints. This chapter shows that with an optimization approach, the algorithm is able to always solve spatial constraints when possible, and usually less example motions are required to cover the spatial variations of interest.

Our method models each spatial constraint as an objective function whose error is to be minimized. The overall multi-objective inverse blending problem is solved by optimizing the blending weights until a locally optimal solution is reached. Solutions can be found in few milliseconds and no pre-computation of basis functions is needed. The method is therefore suitable for interactive applications and several results running in real-time are presented.

While previous work has addressed the maintenance of spatial properties in a single motion interpolation step [RSC01, MK05, KG04], the proposed algorithm is focused on optimizing blending weights until best meeting multiple spatial constraints. This

approach has the additional flexibility of modeling spatial constraints with objective functions which are independent of the abstract space used by the motion blending. Generic spatial constraints can be handled and Inverse Kinematics problems can also be solved based on motion blending. While other work has explored the problem in solving blending weight [YKH04], our contribution is its evaluation for the skeletal animation problem. This approach has recently been integrated in particular to an interactive virtual reality training platform [CHK10, HCK12], and the obtained results were very effective.

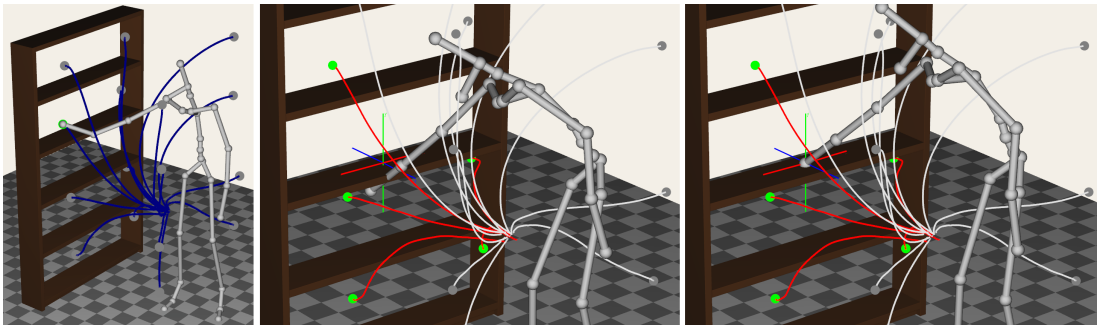


Figure 3.1: The images show hand trajectories of 16 reaching motions to different locations around the character. The center image shows the result of a standard K-Nearest Neighbors (KNN) interpolation of the highlighted 5 closest motions, with blending weights computed in respect to the distances to the target location (center of tri-axes manipulator) using RBF kernel functions. The right-most image shows the accurate target being reached using Inverse Blending.

This chapter demonstrates our methods in three scenarios: pointing to objects, pouring water and character locomotion. The spatial constraints of inverse blending are modeled differently for each scenario. As a result, our interactive motion modeling framework allows animators to easily build a repertoire of realistic parametrized human-like motions (gestures, actions, locomotion, etc) from examples which can be designed by hand or collected with motion capture devices.

3.2 Related Work in Motion Parametrization

Several approaches to motion interpolation have been proposed involving different techniques such as: parameterization using Fourier coefficients [UAT95], hierarchical filtering [BW95], stochastic sampling [WH97], and interpolation based on radial basis functions (RBFs) [RBC98]. Our motion blending framework is closely related to an extension of the verbs and adverbs system which performs RBF interpolation to solve the Inverse Kinematics (IK) problem [RSC01]. RBFs can smoothly interpolate given motion examples and the types and shapes of the basis functions are optimized in order to better satisfy the constraint of reaching a given position with the hand.

More generically, spatial properties such as feet sliding or hand placements are well

addressed by the geostatistical interpolation method [MK05], which computes optimal interpolation kernels in accordance with statistical observations correlating the control parameters and the motion samples. Another approach for improving the maintenance of spatial constraints is to adaptively add pseudo-examples [RSC01, KG04] in order to better cover the continuous space of the constraint. This random sampling approach however requires significant computation and storage in order to meet constraints accurately and is not suited for handling several constraints.

In all these previous methods spatial constraints are only handled as part of the motion blending technique employed, i.e., by choosing sample motions which are close to the desired constraints and then using the abstract interpolation space to obtain motion variations which should then satisfy the constraints. Another possible technique sometimes employed is to apply Inverse Kinematics solvers in addition to blending [RSC01, CHP07], however risking to penalize the obtained realism.

The work on mesh-based IK [SZG05] does address the optimization of blending weights for the problem of blending example meshes. However, although our approach is simple and intuitive, no specific previous work has specifically analyzed and reported results applied to skeletal motion, and in particular also simultaneously solving multiple constraints and at different frames.

The Scaled Gaussian Process Latent Variable Model (SGPLVM)[GMH04] provides a more specific framework targeting the IK problem which optimizes interpolation kernels specifically for generating plausible poses from constrained curves such as positional trajectories of end-effectors. The approach however focuses on maintaining constraints described by the optimized latent spaces. Although good results are obtained, constraints cannot be guaranteed to be precisely met.

The presented approach can be seen as a post-processing step for optimizing a given set of blending weights, which can be initially computed by any motion blending technique. This chapter demonstrates that such optimization framework is able to address any kind of constraints without even the need of specifying an abstract parameterization space explicitly. Only error functions for the spatial constraints are necessary in order to optimize the blending weights using a given motion interpolation scheme.

This method has also been applied for locomotion parameterization, which is a key problem in character animation. Many methods have been previously proposed for finding optimal solutions for path following [KS05, KGP02], for reaching specified locomotion targets [SH07, HG07], or also for allowing interactive user control [TLP07, KS05, ALP04]. Most of these works combine motion blending techniques with motion graphs [KGP02, AF02, AFO03, GSK03, SH07] and can then generate different styles of locomotion and actions. Different than these methods, specific attentions were given to precisely meet specified foot placements. The geostatistical interpolation method [MK05] reduces foot sliding problems but still cannot guarantee to eliminate them. Other precise feet placement techniques [KSG02, CBY08] are available, however not based on a generic motion blending approach.

In conclusion, diverse techniques based on motion blending are available and sev-

eral of these methods are already extensively used in commercial animation pipelines for different purposes. Our work presents valuable experimental results demonstrating the flexibility and efficiency of a simple optimization framework for solving inverse blending problems involving multiple spatial constraints. Our approach is effective and easy to implement, and was first described in [CHK10], where it was applied to an interactive virtual reality training application. The formulation is again described here for completeness purposes, and then several extensions and new results are presented to effectively model diverse parametrized upper-body actions, and also to model parametrized walking animations with precise footstep placements.

3.3 Inverse Blending

Given a set of similar and time-aligned motion examples, a traditional RBF motion interpolation scheme was first employed to compute an initial set of blending weights. These weights are then optimized to meet a given constraint C.

Each motion M being interpolated is represented as a sequence of poses with a discrete time (or frame) parameterization t. A pose is a vector which encodes the root joint position and the rotations of all the joints of the character. Rotations are encoded with exponential maps but other representations for rotations (as quaternions) can also be used. Each constraint C is modeled with a function $e = f(\boldsymbol{M})$, which returns the error evaluation e quantifying how far away the given motion is from satisfying constraint C. k number of motions are first selected, which are the ones best satisfying the constraints being solved. For example, in a typical reaching task, the k motion examples having the hand joint closest to the target will be selected. The k initial motions are therefore the ones with minimal error evaluations. The initial set of blending weights w_j, $j = \{1, \ldots, k\}$, are then initialized with a RBF kernel output of the input $e_j = f(M_j)$. The weights have been constrained to be in interval $[0, 1]$ in order to stay in a meaningful interpolation range, and also normalized to sum to 1. Any kernel function can be used, as for example kernels in the form of $exp^{-\|\boldsymbol{e}\|^2/\sigma^2}$. We chose to use RBF kernel for the property of smoothness, i.e. reducing jitters and jumps of the result postures when C is being continuously changed, see Chapter 4 for details. For this method, it is not necessary to optimize kernel functions in respect to the constraints as in [RBC98, MK05], instead the blending weights are optimized independently of the interpolation method. Our interpolation scheme computes a blended motion with:

$$\boldsymbol{M}(\boldsymbol{w}) = \sum_{j=1}^{k} w_j M_j, \quad \boldsymbol{w} = \{w_1, \ldots, w_k\}.$$

In order to enforce the given constraint C, our goal is to find the optimal set of blending weights $\boldsymbol{w}$, which will produce the minimum error e^* as measured by the constraint error function f:

$$e^* = min_{w_j \in [0,1]} \; f\left(\sum_{j=1}^{k} w_j M_j\right).$$

The presented formulation can easily account for multiple constraints by combining the error metric of the given constraints in a single weighted summation. To do so, two more coefficients are introduced for each constraint C_i, $i = \{1, \ldots, n\}$: a normalization coefficient n_i and a prioritization coefficient c_i. The goal of n_i is to balance the magnitude of the different constraint errors. For example, positional constraints depend on the used units and in general have much larger values than angular values in radians, which are typically used by orientation errors. Once the normalization coefficients are set, coefficients $c_i \in [0, 1]$ can be interactively controlled by the user in order to vary the influence (or priority) of one constraint over the other.

The result is essentially a multi-objective optimization problem, with the goal being to minimize a new error metric composed of the weighted sum of all constraints' errors:

$$f(\boldsymbol{M}(\boldsymbol{w})) = \sum_{i=1}^{n} \left(c_i \; n_i \; f_i \left(\boldsymbol{M}(\boldsymbol{w})\right)\right).$$

Independent of the number of constraints being addressed, when constraints are fully satisfied, $e^* \to 0$.

3.4 Action Parameterization with Inverse Blending

Figure 3.2 presents several examples obtained for parameterizing upper-body actions. Different types of spatial constraints were used. Constraint C_{pos} is a 3-DOF positional constraint which requires the end-effector (hand, finger tip, etc) to precisely reach a given target location. Constraint C_{line} is a 2-DOF positional constraint for aligning the hand joint with a given straight line in 3D space. Constraint C_{ori} is a 1 to 3 DOFs rotational constraint which requires the hand to comply to a certain given orientation. Note that all these constraints are only enforced at one given frame of the motion. Constraints can also be combined in order to allow additional ways of parameterizing motions. For example by combining C_{line} and C_{ori}, precise grasping targets can be achieved (Figure 3.2-d), and different hand orientations can be obtained when pin-pointing buttons (Figure 3.2-b). Fig. 3.3 shows another example for a collection of pointing actions with subtle variations obtained by combining different spatial constraints.

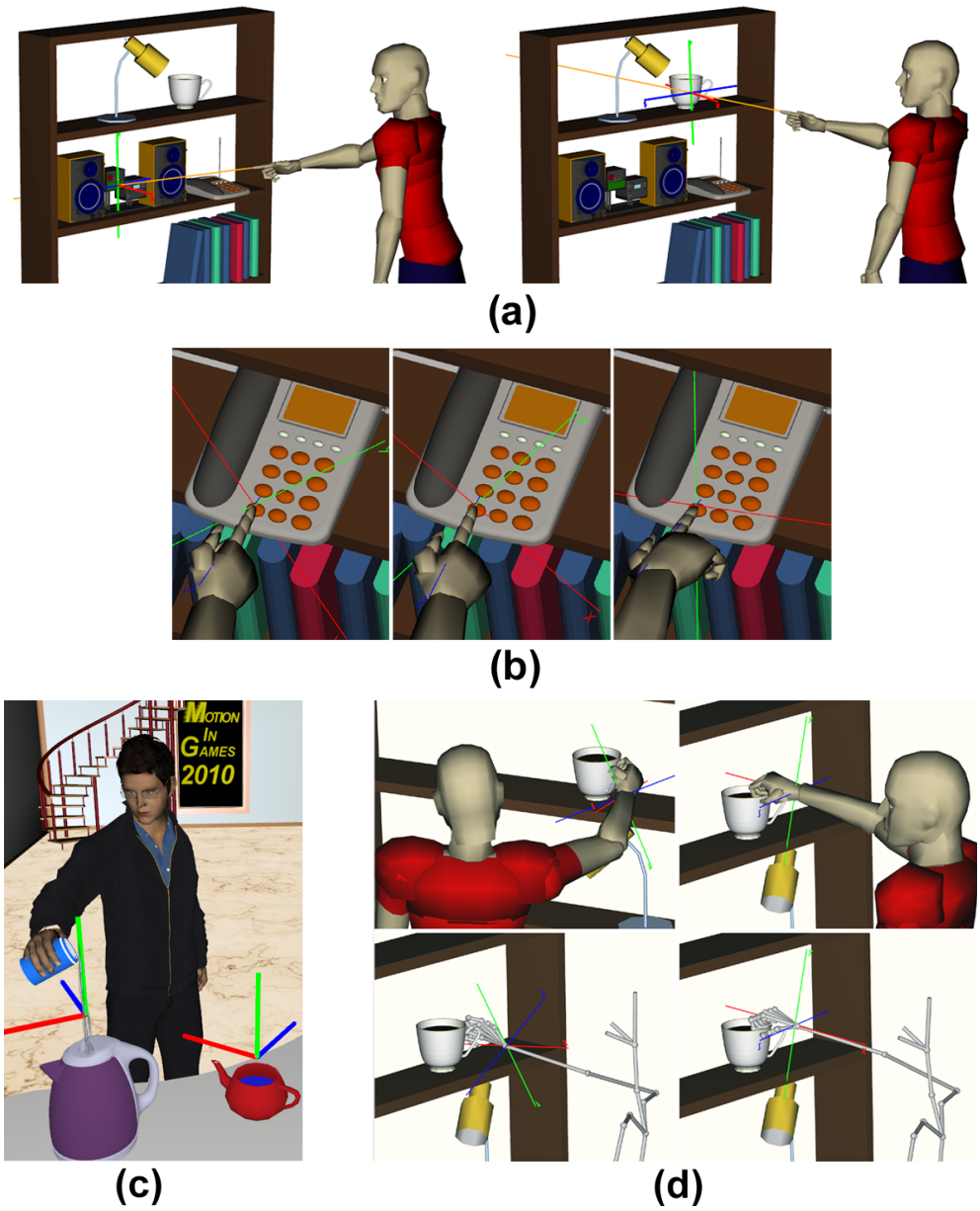


Figure 3.2: This figure presents several results obtained by inverse blending. A C_{line} constraint is used for precisely pointing to distant objects in (a) and for pouring exactly above the teapot in (c). Positional C_{pos} and rotational C_{ori} constraints are used for pin-pointing a button on the dialpad (b). Note that the finger tip precisely reaches the button, and the hand orientation matches that of the shown x-axis. Constraints C_{pos} and C_{ori} are also used for achieving precise grasping motions in (d).

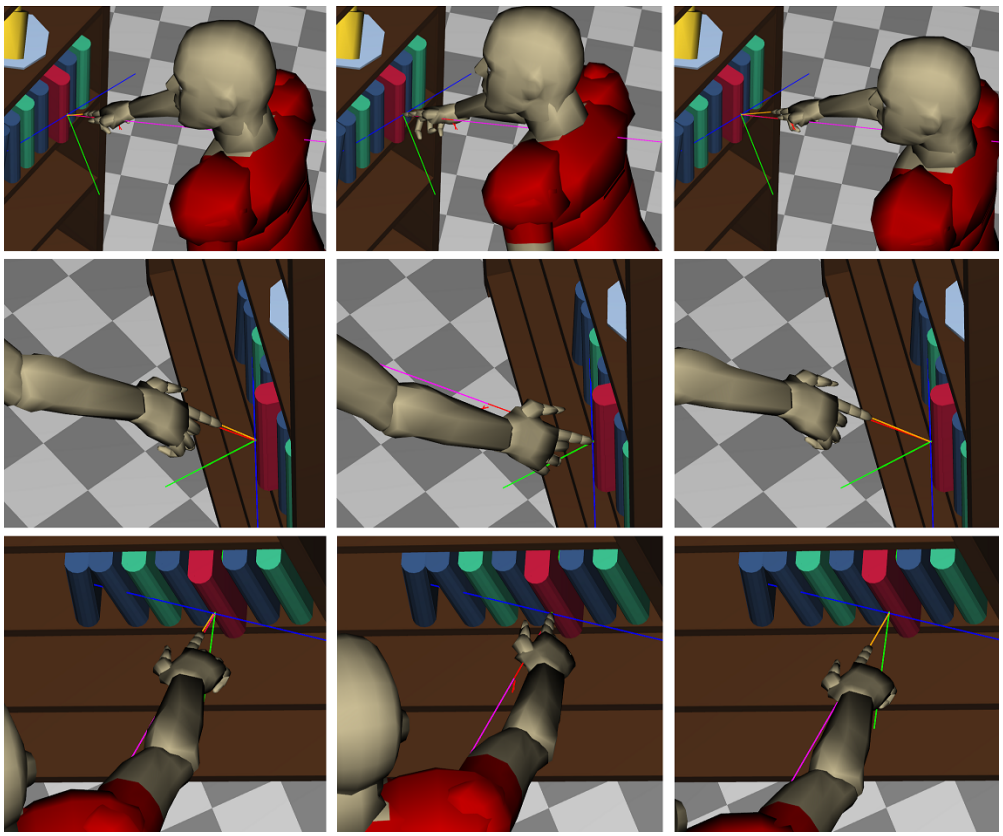


Figure 3.3: Each column illustrates a pointing action with subtle variations achieved by combining different spatial constraints. From left to right, the constraints used in each column are as follows: $0.54C_{line} + 0.46C_{ori}$, $0.61C_{line} + 0.39C_{pos}$, and $0.45C_{line} + 0.35C_{pos} + 0.20C_{ori}$. The three images in each column are the same action but from different perspectives. The tri-axes manipulator is used to specify the pointing target location C_{pos} and the preferred hand orientation C_{ori}. The magenta line specifies the preferred pointing direction, i.e. C_{axis}. The orange line is traced from the pointing finger showing the actual pointing direction.

3.5 parametrized Character Locomotion with Inverse Blending

This section demonstrates our inverse blending framework for generating parametrized walking motions with precise control of feet placements for character navigation.

First, two sets of motion examples are prepared with clips obtained from motion capture. The first set $\boldsymbol{R}_m$ consists of right foot stepping motions. Each example motion $M_r \in \boldsymbol{R}_m$ represents one full step forward with the right foot while the left foot remains in contact with floor as the support foot, see Figure 3.4-a. The second set of example motions $\boldsymbol{L}_m$ is prepared in the same way but containing stepping examples for left foot.

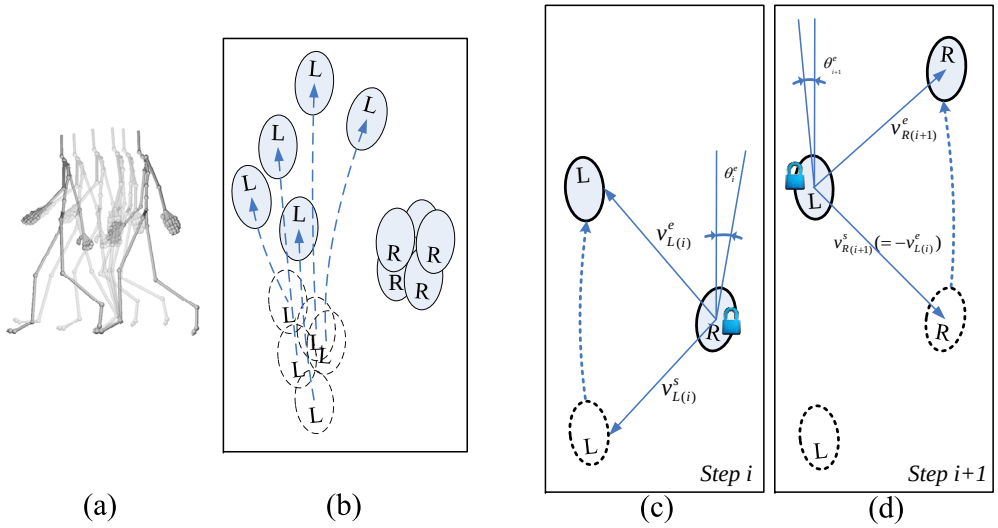


Figure 3.4: (a) shows snapshots of one right foot stepping motion from $\boldsymbol{R}_m$ example set. (b) is a top-down footprint view of several left foot stepping motions used in $\boldsymbol{L}_m$ example set. The footprints are from diverse walk segments and do not need to be aligned. (c) shows the constraints for computing step i: θ^e is the rotational constraint for the support foot (with lock icon), v^s and v^e are positional constraints for the stepping foot at the start and end of the motion respectively. (d) shows the constraints for step $i+1$, which immediately follows step i.

The motions in both sets contain many variations, e.g. step length, step direction, body orientation, velocity, root joint position, etc. These should well span the variations of interest, which are to be precisely parametrized by inverse blending. Figure 3.4-b illustrates how some of the motions in our database look like. No alignment of the motions is needed and the variations will actually be explored by the inverse blending optimization in order to reach any needed alignments on-line. The example motions were also mirrored in both sets in order to guarantee the same number of examples (and variations) are available in each set.

As we are interested in precisely controlling each footstep location during walking, the length and direction of each step is parametrized while the support foot contact

on the floor is maintained. Let θ^e be the orientation of the support foot at the starting frame of one stepping motion example (rotational constraint). Let v^s and v^e be vectors encoding the position of the stepping foot in respect to the support foot at the start and at the end frames respectively (positional constraints). Figure 3.4-c illustrates these parameters for one left step. Each stepping motion of interest is then specified as a function of these parameters with $M(v^s, v^e, \theta^e)$, which will be obtained by inverse blending procedures based only on the stepping examples available in the $\boldsymbol{R}_m$ and $\boldsymbol{L}_m$ motion sets.

With the given constraints v^s, v^e, θ^e described above, the process for obtaining $M(v^s, v^e, \theta^e)$ is composed of 4 phases, as described in the next paragraphs.

Phase 1: the set of blending weights $\boldsymbol{w}^s$ is computed by inverse blending such that the stepping foot respects the positional constraint v^s at the start frame t^s. As these weights are computed to meet constraints v^s, we use the notation $\boldsymbol{w}^s(v^s)$ for the obtained blending weights.

Phase 2: a new inverse blending problem is solved for determining the blending weights $\boldsymbol{w}^e$ at the end frame t^e in order to meet two constraints: the positional constraint v^e for the stepping foot and the rotational constraint θ^e for the support foot. Therefore the obtained weights $\boldsymbol{w}^e(v^e, \theta^e)$ produce an end posture with the stepping foot reaching location v^e, while the support foot respects the orientation specified by θ^e.

Phase 3: the average lengths l_{avg} of the example motions in phase 1 and 2 is used to time-align the blended motions. The blending weights used to produce the required stepping motion is finally obtained as a function of the frame time t, such that $\boldsymbol{w}(t)$ $=interp(\boldsymbol{w}^s(v^s), \boldsymbol{w}^e(v^e, \theta^e), t), t \in [0, l_{avg}]$. The interpolation function $interp$ employs a smooth in and out sine curve and each of the motions are time warped to l_{avg} in order to cope with variations of step lengths and speeds in the used motion set. The final parametrized stepping motion is then obtained with $M(v^s, v^e, \theta^e) = \boldsymbol{w}(t)\Sigma_{i=1}^{k} M_i$. This process is illustrated in Figure 3.5.

Phase 4: this phase consists of a velocity profile correction [YKH04] in order to maximally preserve the overall quality of the original motions since several blending operations have been performed at this point. To do so, the algorithm extracts the root joint velocity profile from the motion example that gives the most contribution in the inverse blending procedures (i.e. example with the largest weight). The time parameterization of $\boldsymbol{w}(t)$ is then adapted on the fly in order to obtain a motion with the root joint velocity profile matching the selected reference profile. Figure 3.5 bottom-right exemplifies the root increment against frames during a two-step walking sequence showing how root velocity changes over time. This has been proven to well preserve the quality of the obtained results.

The procedure described above is applied each time a stepping motion has to be generated. For producing stepping motions for the right foot, $M_R(v^s, v^e, \theta^e)$ is obtained by using the example motions from $\boldsymbol{R}_m$. Left foot stepping motions $M_L(v^s, v^e, \theta^e)$ are similarly obtained using examples from $\boldsymbol{L}_m$. As a result a walking sequence achieving precise feet placements at each step can be obtained with the following concatenation

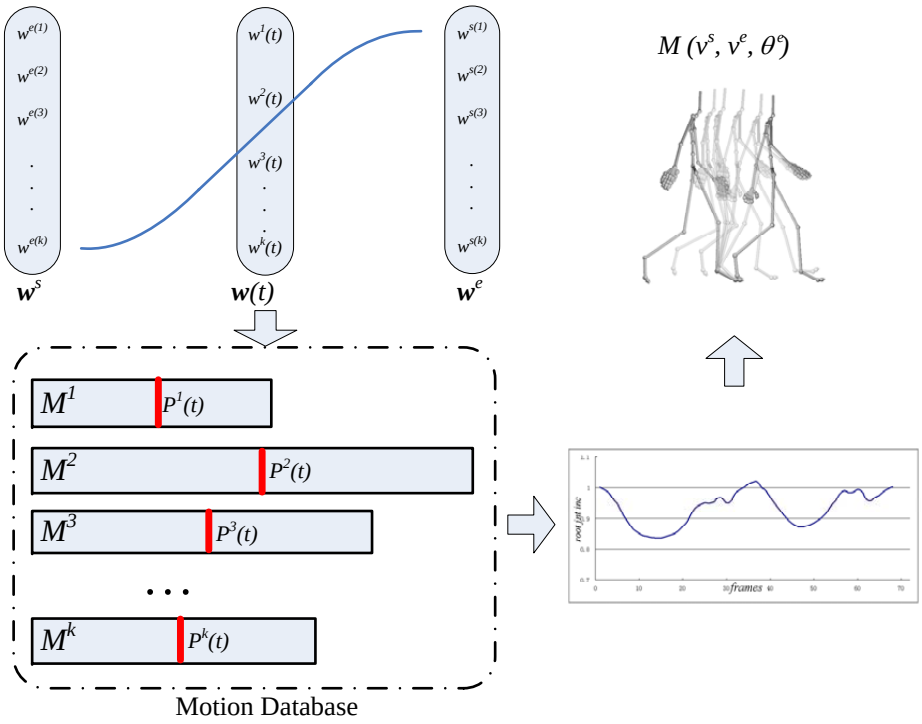


Figure 3.5: The motion examples selected by the inverse blending $M^i(i = 1 \sim k)$ are blended with interpolated weights $\boldsymbol{w}(t)$ which ensure spatial constraints both at the start and end frame of the motions. Velocity profiles are adapted on-line to preserve the original quality and realism.

of alternating steps: $M_L(v_1^s, v_1^e, \theta_1^e)$, $M_R(v_2^s, v_2^e, \theta_2^e)$, $M_L(v_3^s, v_3^e, \theta_3^e)$, $\cdots$.

During each stepping motion in the sequence above, the character is translated at every frame to make sure the support foot does not slide on the floor (i.e. its location and orientation are maintained), this will essentially make the character walk forward. At the end of each stepping, the support foot changes, and its location and orientation are updated, ready for the following step. With this, the common problem of feet-sliding is here eliminated.

When computing each stepping motion, constraint v_{i+1}^s is set equal to $-v_i^e$ from the previous step (see Figure 3.4-c and 3.4-d), for smooth transition between step i and step $i + 1$. The negation appears because the support foot and stepping foot are swapped from step i to $i + 1$.

Figure 3.6 shows the end posture P_L^e (thick red line) of the left step $M_L(v_i^s, v_i^e, \theta_i^e)$ and the start posture P_R^s (thick green line) of the right step $M_R(v_{i+1}^s, v_{i+1}^e, \theta_{i+1}^e)$. With $v_{i+1}^s = -v_i^e$, inverse blending generates postures P_L^e and P_R^s matching the constraints and with body postures which are very close to each other. The small difference in the body posture is handled by smoothly concatenating the stepping motions with a brief ease-in blending period from M_L going into M_R, achieving a smooth overall transition.

In the examples presented in this chapter, only six stepping motion examples have been used in each motion set, and yet the described inverse blending procedure can precisely control each foot placement within a reasonable range. If larger databases are used, a wider range for the constraints can be specified. Figure 3.7 shows several results obtained by our real-time walking control application.

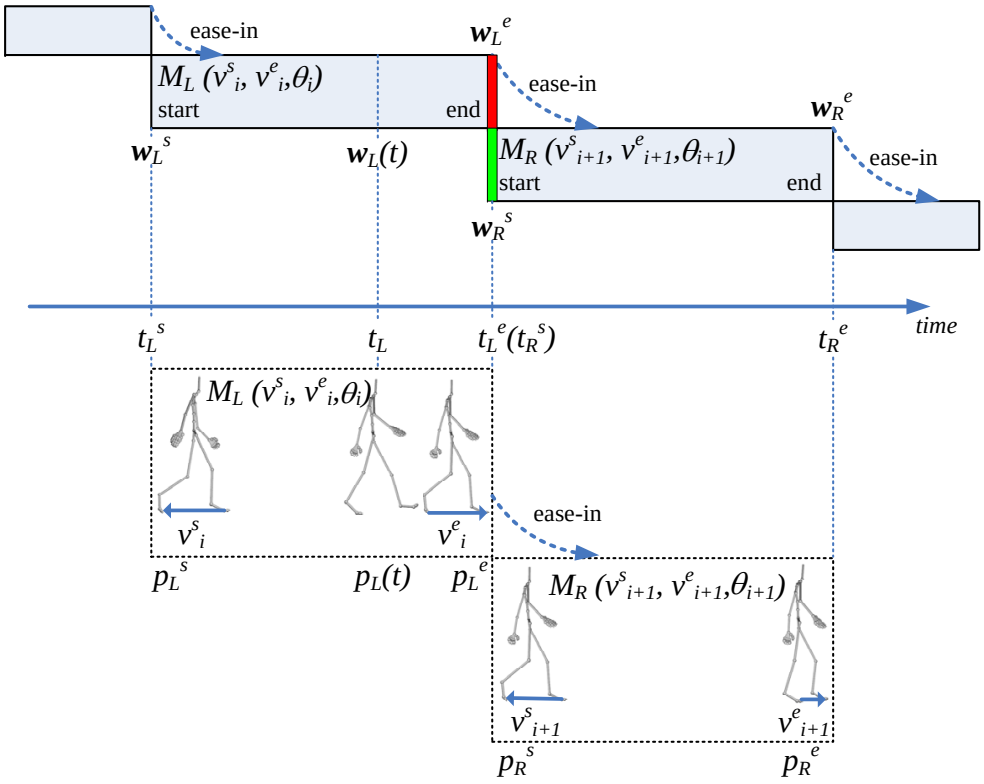


Figure 3.6: This figure illustrates the generation of $M_L(v_n)$, and the concatenation of $M_L(v_n)$ and $M_R(v_{n+1})$ for achieving the final walking synthesis.

3.6 Results and Discussion

With suitable example motions in a given cluster, inverse blending can produce motions exactly satisfying given spatial constraints and fast enough for real-time applications. Several of the figures in this section illustrate the many experiments successfully conducted in different scenarios. To evaluate the performance of our method, a reaching task was designed to measure the errors produced by our method against a single RBF interpolation, with the 16 reaching motion database from Mukai et.al [MK05]. A total of 144 reaching targets (shown as yellow dots in Fig 3.10, each specifying a 3-DOF C_{pos} constraint) were placed evenly on a spherical surface within reach of the character. The end locations of the hand trajectory in 16 example motions are shown as gray dots.

For each reaching target we first applied standard RBF interpolation alone to gen-

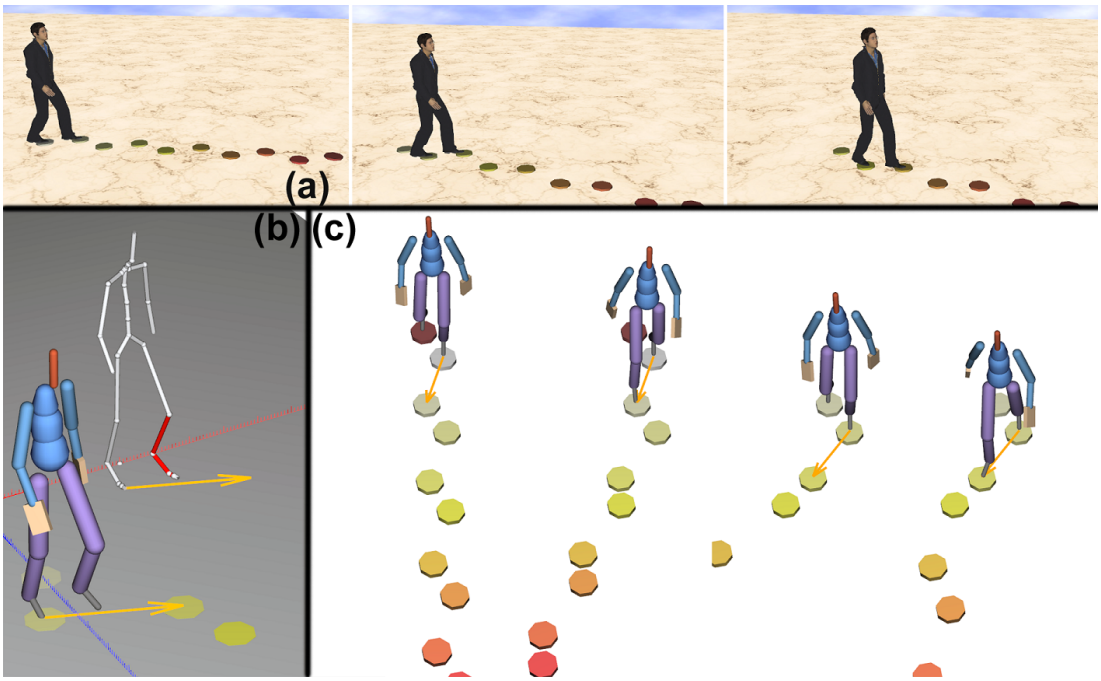


Figure 3.7: The figure presents several snapshots of obtained walking motions where each step precisely meets the given feet targets (a and c). The targets for each step can be adjusted on the fly achieving a controllable locomotion generator with precise feet placements. The generation of the stepping motions is illustrated in the lower-left image (b), where the gray skeleton shows the inverse blending solution for the left foot, prior to concatenation.

erate a reaching motion and record the final hand position where the character actually reaches. These 144 final positions were used to construct a mesh grid, which is shown in Fig 3.10-a. Each triangle on the mesh is colored in respect to the average errors from its vertices, representing the distance error between the final hand positions and their corresponding reaching targets. Next, inverse blending optimization was used to perform the same tasks, and the constructed mesh is shown in Fig 3.10-b. The reaching motions generated by inverse blending can precisely reach most of the targets. Errors measured are practically zero across most of the mesh, and increase only at the boundary of the surface. In this specific task, the radius of the spherical surface was set to $80cm$, and both methods used eight example motions from the database ($k = 8$) for computing each reaching task. Fig. 3.8 shows the error evaluation without the spatial information. The XY plane corresponds to the 2D grid containing 144 samples in the constraint space. Distance errors are plotted on the vertical Z-axis, in centimeters. Inverse Blending achieved fairly good precisions over most of the grid region, with distance errors under 5 to 10 millimeters for given 3-DOFs positional constraints. By contrast, RBF interpolation produced much larger errors over the same grid, with errors reaching 3 to 15 centimeters.

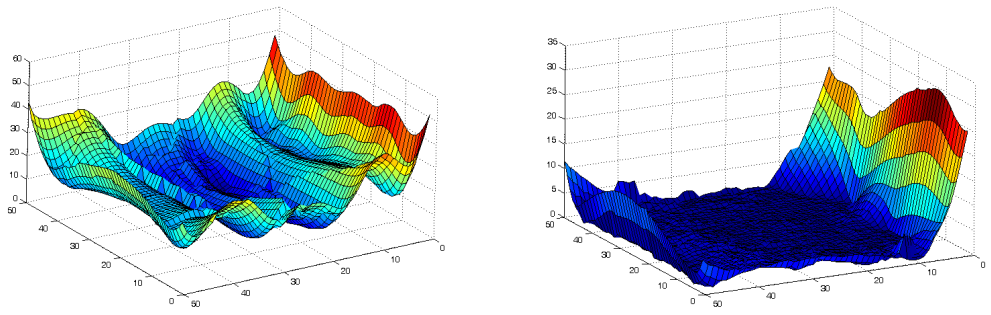


Figure 3.8: Error evaluation without the spatial information. Left image plots the errors measured from RBF interpolation, right side plots errors from our method.

	motion type	*k*	optimization method	
			steepest descent	gradient descent
avg comp time (milliseconds)	**pointing**	10	1.088	0.945
		15	1.286	1.153
		30	2.287	1.743
	pouring	10	1.126	1.065
		15	1.135	1.065
	walking	6	2.734	2.487

Figure 3.9: The table shows average computation times (in milliseconds) in solving 5000 inverse blending tasks, within three scenarios: pointing, pouring and walking. Comparison is made between two of the optimization procedures implemented: steepest descent and gradient descent. k denotes the number of blended motions used in each task. Total number of motion examples in each dataset: 35 pointing motions, 16 pouring motions, and 6 stepping motions.

Additional experiments were performed by varying the radius of the spherical surface to be $65, 70, 75, 80, 85$ and $90cm$. Again a total of 144 reaching targets were generated on the spherical surfaces, covering a large volume of the workspace. These spherical surfaces are shown in Figures 3.10-c and 3.10-d. The constructed meshes by inverse blending are shown in Fig 3.10-d, and the results obtained with the RBF interpolation are shown in Fig 3.10-c. It is possible to note that the inverse blending optimization produces a smooth mesh very well approximating the yellow dots, and the errors produced by our method are clearly much lower, with most areas in pure blue.

Using standard optimization techniques [PTV07] our inverse blending problems could be solved under 1.16 ms of computation time on average, with worse cases taking 2.29 ms (with non-optimized code running on Core 2 Duo 2.13 GHz single core). Three scenarios (character performing pointing, pouring water and walking with precise feet placements) were used for this evaluation, with each scenario solving 5000

inverse blending problems towards random placements. Details of the measurements are summarized in Fig. 3.9. Note that for each step in the walking synthesis task, 2 inverse blending problems are being solved: one at the beginning of the step, and the other one at the end of that same step.

The approach is therefore suitable for real-time applications, and in addition, it does not require pre-processing of the motion database, making it suitable for systems interactively updating the motion database (as in [CHK10]).

In terms of limitations, two main aspects have to be observed. First, the ability of enforcing constraints greatly depends on the existing variations among the motion examples being blended. The number of needed example motions also depends on the size of the target volume space. For example, our walk generator can produce good results with only 6 stepping example motions (6 for left foot stepping ranging from -10° to 65°) and mirrored to become 12 for both turning directions, due to great variations available in the motions. However more example motions are typically needed to well cover a large reaching or pointing volume, in certain cases 35 example motions were used. In extreme cases, the motion examples needed to enforce specified constraints is exponential to the number of DOFs. For example, adding one extra DOF constraint to an existing dataset for 3-DOFs positional constraint synthesis may require doubling the size of the dataset. In practice however, it is less demanding on the quantity of new motions to add when introducing additional constrains. For example, to synthesis reaching motions with a 3-DOFs positional constraint typically requires a dataset of 16 to 35 examples that well cover the reachable space. Adding one extra rotational constraint in practice would normally requires an additional 8 to 15 carefully chosen examples that cover the extremities. The second limitation, which is related to the first, is that the computational time required for finding solutions will also depend on the quality and number of motion examples (the k value). However, as shown in our several examples, these limitations are easy to address by appropriately modeling example motions, and balancing the coverage versus efficiency trade-off specifically for each action being modeled.

To conclude, this chapter presents an optimization framework for satisfying spatial constraints with motion blending. Our approach is simple and can handle any type of spatial constraints. Several different actions (pointing, grasping, pouring, and walking) were successfully modeled and parametrized with precise placement of end-effectors. Our inverse blending framework has therefore shown to be a simple and powerful tool for achieving several useful motion parameterizations. We believe this overall framework can significantly improve the process of modeling full-body motions for interactive characters.

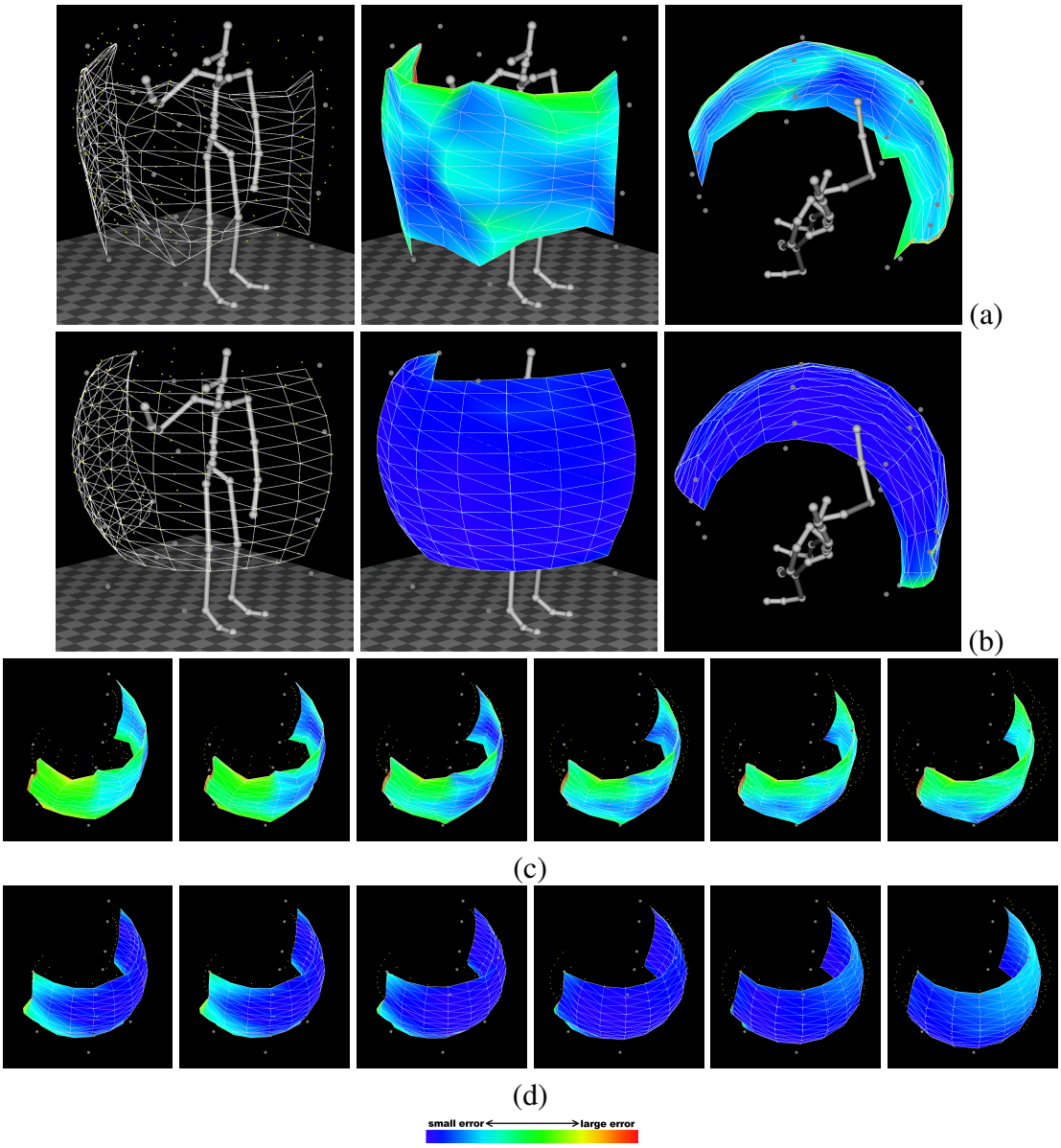


Figure 3.10: Error evaluations. The meshes constructed by a standard RBF interpolation (a and c) result in much larger errors than by our inverse blending optimization (b and d), which most often produces no error.

CHAPTER 4

An Analysis of Motion Blending Techniques

Motion blending is a widely used technique for character animation. The main idea is to blend similar motion examples according to blending weights, in order to synthesize new motions parameterizing high level characteristics of interest. We present in this chapter an in-depth analysis and comparison of four motion blending techniques: Barycentric interpolation, Radial Basis Function, K-Nearest Neighbors and Inverse Blending optimization. Comparison metrics were designed to measure the performance across different motion categories on criteria including smoothness, parametric error and computation time. The blending methods have been implemented in the character animation platform SmartBody, and several visualization renderings are presented here to provide a window for gleaning insights into the underlying pros and cons of each method in an intuitive way.

4.1 Introduction

Motion blending, also known as motion interpolation, is widely used in interactive applications such as in 3D computer games and virtual reality systems. It relies on a set of example motions built either by key-framing animation or motion capture, and represents a popular approach for modifying and controlling high level characteristics in the motions. In essence, similar motion examples are blended according to blending weights, achieving parameterizations able to generate new motions similar to the existing examples and providing control of high level characteristics of interest to the user.

Although several different methods have been proposed in previous works, there is yet to be a one-solves-all method that works best in all scenario. Each method has its own advantages and disadvantages. The preferred motion parameterization for an application highly depends on the application type and required constraints. For locomotion animation, it is more important to ensure the weights vary smoothly as the parameterization generates them, in order to ensure visually pleasing results. On the other hand, a reaching motion forms a non-linear parameter space and requires precise goal-attainment for grasping. Therefore methods with less parametric error are preferred for accuracy. The goal of this work is to thoroughly analyze several existing methods for motion parameterization and to provide both visual and numerical metrics for discriminating different methods.

This chapter presents an detailed analysis among 4 popular motion blending meth-

ods implemented on the character animation platform SmartBody [TMM08, USC12], including Barycentric interpolation, Radial Basis Function (RBF) interpolation, K-Nearest Neighbors (KNN) interpolation, and Inverse Blending (InvBld) optimization [HK10]. The goal is to include methods that are fast and straightforward to implement, without heavy computations normally found in the machine learning-based methods.

A motion capture dataset was carefully built containing 5 different type of motions, and comparison metrics were designed to measure the performance on different motion categories using criteria including parametrization accuracy of constraint enforcement, computation time and smoothness of the final synthesis. Comparison results are intuitively visualized through dense sampling inside the parametrization (blending) space, which is formulated using the corresponding motion parameters. These results provide valuable insights into the underlying advantages and disadvantages of each blending method.

4.2 Related Work in Motion Blending

Motion blending is a powerful technique to generate variations of existing motions. It can be used to synthesize locomotion animation to steer a virtual character [PSS02, PL06], produce reaching and grasping motions that satisfy spatial constraints [AN99], or generate motions of different styles [RBC98].

Several blending methods have been proposed in the literature to produce flexible character animation from motion examples. Different approaches have been explored, such as: parameterization using Fourier coefficients [UAT95], hierarchical filtering [BW95] and stochastic sampling [WH97]. The use of Radial Basis Functions (RBFs) for building parameterized motions was pioneered by Rose et al. [RBC98] in the Verbs and Adverbs system, and follow-up improvements have been proposed [RSC01, PSS02]. RBFs can smoothly interpolate given motion examples, and the types and shapes of the basis functions can be fine tuned to better satisfy constraints for different dataset. The method assigns a function for each blending weight that is the sum of a linear polynomial and a set of radial basis functions, with one radial basis function for each example. If the requested parameters are far from the examples, blending weights will largely rely on the linear polynomial term, which undermines the accuracy of the final result. Verbs and Adverbs formulation has been selected as the RBF interpolation method used in the presented comparisons.

Another method for interpolating poses and motions is KNN interpolation. It relies on interpolating the k nearest examples measured in the parametrization space. The motion synthesis quality can be further improved by adaptively adding pseudo-examples in order to better cover the continuous space of the constraint [KG04]. This random sampling approach however requires significant computation and storage in order to well meet constraints in terms of accuracy, and can require too many examples to well handle problems with multiple constraints.

In all these blending methods spatial constraints are only handled as part of the employed motion blending scheme. One way to solve the blending problem with the objective to well satisfy spatial constraints is to use an Inverse Blending optimization procedure [HK10], which directly optimizes the blending weights until the constraints are best satisfied. However, a series of smooth variation inside the parametrization space may generate non-smooth variations in the weight space, which lead to undesired jitter artifacts in the final synthesis. Another possible technique sometimes employed is to apply Inverse Kinematics solvers in addition to blending [CHP07], however risking to penalize the obtained realism.

Spatial properties such as hand placements or feet sliding still pose challenges to the previously mentioned blending methods. Such problems are better addressed by the geo-statistical interpolation method [MK05], which computes optimal interpolation kernels in accordance with statistical observations correlating the control parameters and the motion samples. When applied to locomotion systems, this method reduces feet sliding problems but still cannot guarantee to eliminate them. The scaled Gaussian Process Latent Variable Model (sGPLVM)[GMH04] provides a more specific framework targeting similar problems with optimization of interpolation kernels specifically for generating plausible poses from constrained curves such as hand trajectories. The approach however focuses on maintaining constraints described by the optimized latent spaces. Although good results were demonstrated with both methods, they remain less seen in animation systems partly because they are less straightforward to implement.

The analysis also includes the performance of selected blending methods for locomotion parametrization, which is a key problem in character animation. Many blending-based methods have been proposed in the literature for locomotion [KS05, KGP02, Joh09], for interactive navigations with user input [KS05, ALP04], for reaching specified targets or dodging obstacles during locomotion [SH07, HG07], and also for space-time constraints [KHK09, SKF07, LLK11]. Here in this chapter the performance of the selected blending methods are investigated on the synthesis of locomotion sequences.

In conclusion, diverse techniques based on motion blending are available and several of these methods are already extensively used in commercial animation pipelines for different purposes. In this chapter valuable experimental results are presented to uncover the advantages and disadvantages of four motion blending techniques. The rest of this chapter is organized as follows: In Chapter 4.3 and 4.4 each blending method is briefly reviewed, then the setup of the experiments and selected analysis metrics are described. Chapter 4.5 present detailed comparison results, and Chapter 4.6 presents the conclusions.

4.3 Motion Parameterization Methods

In general, a motion parameterization method works as a black box that maps desired motion parameters to blending weights for motion interpolation. We have selected four methods (Barycentric, RBF, KNN, and InvBld) for our analysis. This work focuses on

comparing the methods that are most widely used in practice due to their simplicity in implementation, and other methods like [GMH04] and [MK05] are not included. Below is a brief review of the 4 methods selected. Each motion M_i being blended is represented as a sequence of poses with a discrete time (or frame) parameterization t. A pose is a vector which encodes the root joint position and the rotations of all the joints of the character. Rotations are encoded as quaternions but other representations for rotations can also be used. Our interpolation scheme computes the final blended motion $M(\boldsymbol{w}) = \sum_{i=1}^{k} w_i M_i$, with $\boldsymbol{w} = \{w_1, \ldots, w_k\}$ being the blending weights generated by each of the methods.

4.3.1 Linear and Barycentric Interpolation

Linear and Barycentric interpolation is the most basic form for motion parameterization. It assumes that motion parametrization can be linearly mapped to blending weights. While the assumption may not hold for all cases, in many situations this simple approximation does achieve good results. Without loss of generality, we assume a 3D motion parameter space with a set of example motions M_i ($i = 1 \ldots n$). As the dimension of parametric space goes from 1D to n-D, linear interpolation becomes Barycentric interpolation. Specifically for a 3D parametric space, the tetrahedralization $V = \{T_1, T_2, \ldots T_v\}$ is constructed to connect these motion examples M_i in space, which can be done either manually or automatically using Delaunay triangulation. Therefore, given a new motion parameter $\boldsymbol{p'}$ in 3D space, the algorithm can search for a tetrahedron T_j that encloses $\boldsymbol{p'}$. The motion blending weights $w = \{w_1, w_2, w_3, w_4\}$ are given as the barycentric coordinates of $\boldsymbol{p'}$ inside T_j. Similar formulations can be derived for 2-D and n-D cases by replacing a tetrahedron with a triangle or a n-D simplex respectively. The motion parameterization $\boldsymbol{p'}$ is not limited to a 3D workspace but can also be defined in other abstract parameterization spaces, with limited ability for extrapolation outside the convex hull.

4.3.2 Radial Basis Function (RBF)

RBF is widely used for data interpolation since first introduced in [RBC98]. In this chapter the method is implemented by placing a set of basis functions in parameter space to approximate the desired parameter function $f(p) = w$ for generating the blend weights. Specifically, given a set of parameter examples $p_1, p_2, \ldots p_n$, $f(p) = w_1, w_2, \ldots w_n$ is defined as sum of a linear approximation $g(p) = \sum_{l=0}^{d} a_l A_l(p)$ and a weighted combination of radial basis function $R(p)$. The function for generating the weight w_i is defined as :

$$w_i(p) = \sum_{j=1}^{n} r_{i,j} R_j(p) + \sum_{l=0}^{d} a_{i,l} A_l(p)$$

where $R_j(p) = \phi(\|p - p_j\|)$ is the radial basis function, and $A_l(p)$ is the linear basis. Here the linear coefficients $a_{i,l}$ are obtained by fitting a hyperplane in parameter space where $g_i(p_j) = \delta_{i,j}$ is one at i-th parameter point and zero at other parameter points. The RBF coefficients $r_{i,j}$ are then obtained by solving the following linear equation from linear approximation to fit the residue error $e_{i,j} = \delta_{i,j} - g_i(p_j)$.

$$\begin{pmatrix} R_1(p_1) & R_1(p_2) & \dots \\ R_2(p_1) & \dots & \dots \\ \vdots & \dots & \dots \end{pmatrix} r = e$$

RBF can generally interpolate the example data smoothly, though it usually requires some tuning in the types and shapes of basis function to achieve good results for a specific data set. The parameterization space could also be defined on an abstract space like motion styles [RBC98], with the ability to extrapolate outside the convex hull of example dataset. However the motion quality from such extrapolation may not be guaranteed, especially when p travels further outside the convex hull.

4.3.3 K-Nearest Neighbors (KNN)

KNN interpolation finds the k-closest examples from an input parameter point and computes the blending weights based on the distance between the parameter point and nearby examples. Specifically, given a set of parameter examples $p_1, p_2, \dots p_n$ and a parameter point p', the method first finds example points $p_{n_1}, p_{n_2}, \dots p_{n_k}$ that are closest to p'. Then the i-th blending weight for p_{n_i} are computed as :

$$w_i = \frac{1}{\|p - p_{n_i}\|} - \frac{1}{\|p - p_{n_k}\|}$$

The blending weights are then normalized so that $w_1 + w_2 + \dots + w_k = 1$. The KNN method is easy to implement and works well with a dense set of examples in parameter space. However, the result may be inaccurate when the example points are sparse. To alleviate this problem, pseudo-examples are usually generated to fill up the gap in parameter space [KG04]. A pseudo-example is basically a weighted combination of existing examples and can be generated by randomly sampling the blend weights space. Once a dense set of pseudo-examples are generated, a k-D tree can be constructed for fast proximity query at run-time. In our implementation for a dataset containing 20 examples, pseudo-examples on the scale of 500 to 2000 are generated.

4.3.4 Inverse Blending (InvBld)

InvBld was designed for precise enforcement of user-specified constraints in the workspace [HK10]. Each constraint C is modeled with function $e = f^C(\boldsymbol{M})$, which returns the error evaluation e quantifying how far away the given motion is from satisfying constraint C under the given motion parametrization p'. First, the k motions $\{M_1, \dots, M_k\}$

best satisfying the constraints being solved are selected from the dataset, for example, in a typical reaching task, the k motion examples having the hand joint closest to the target will be selected. An initial set of blending weights w_j ($j = \{1, \ldots, k\}$) are then initialized with a radial basis kernel output of the input $e_j = f^C(M_j)$. Any kernel function that guarantee smoothness can be used, as for example kernels in the form of $exp^{-\|\boldsymbol{e}\|^2/\sigma^2}$. Weights are constrained inside $[0, 1]$ in order to stay in a meaningful interpolation range, they are also normalized to sum to 1. The initial weights $\boldsymbol{w}$ are then optimized with the goal to minimize the error associated with constraint C:

$$e^* = min_{w_j \in [0,1]} f^C \left(\sum_{j=1}^{k} w_j M_j \right).$$

Multiple constraints can be accounted by introducing two coefficients n_i and c_i for each constraint C_i, $i = \{1, \ldots, n\}$, and then solve the multi-objective optimization problem that minimizes a new error metric composed of the weighted sum of all constraints' errors: $f(\boldsymbol{M}(\boldsymbol{w})) = \sum_{i=1}^{n} \left(c_i \, n_i \, f^{C_i} \left(\boldsymbol{M}(\boldsymbol{w}) \right) \right)$, where c_i is used to prioritize C_i, and n_i to balance the magnitude of the different errors. Details of the InvBld are provided in Chapter 3.

4.4 Experiments Setup

4.4.1 Dataset Overview

To compare and test these motion parameterization methods, five categories of motion examples have been captured, including the following actions: reach, jump, punch kick and locomotion. Corresponding features are formulated for each motion category to define a parameterization space. Also, three metrics are defined for the evaluation: computation time, parameterization accuracy (or parametric error) and smoothness.

Reach: 24 example motions of a character pointing his arm to various points around him. A typical reach motion makes use of the character's full body movements, including bending down or turning around in order to reach for target points on the ground or behind the character. The parameterization is defined as the 3D position $p = (x, y, z)$ of the wrist joint when character's hand is on the target.

Punch: 20 example punch motions, with the character starting in a fighting stance and then hitting various points in space while remaining feet contact. Since it would be awkward to perform punch at a target in the back without moving the feet and turn, the punch is restricted to hit targets mostly in front of the actor. We defined the motion parameterization as the target hit point $p = (x, y, z)$ in space.

Kick: Kick motions are organized in a similar manner to the punch motions. 20 examples are captured where the character starts from a fighting stance, performs straight kicks at various targets in the air, and then returns to the fighting stance. What's different from the punching is that kicking motions usually cover a larger area in the

workspace since the character can perform back kicks without turning to the back. The parameterization is defined as the 3D position $p = (x, y, z)$ of the ankle joint at the kick apex point.

Jump: 20 jumping motions to various target locations. Targets are on the floor plane to various distances and directions, and also with varied heights during the flight phase to a same target point. The parameterization is defined as $p = (d, \theta, h)$, where d is the jumping distance, θ is the jumping direction, and h is the maximum flight stage height.

Locomotion: 20 example motions in the locomotion set. Each example animation contains two walking or running cycles of moving straight, sideways, or turning around at various speeds. The parameterization is defined as $p = (v_f, \omega, v_s)$, where v_f is the forward speed, ω is the turning rate, and v_s is the speed of walking sideways.

The table and figures in Fig 4.1 gives an overview of the datasets.

4.4.2 Performance Metrics

The main application of motion parameterization is to synthesize new motions interactively based on input parameters. In order to numerically compare the methods, three metrics are defined: computation time, parametrization accuracy and smoothness.

Parametrization Accuracy: While each motion parameterization method can generate a unique set of blending weights given some input parameters, there is no guarantee that the blended motion will satisfy the input parameters. The parametric error is defined as the squared difference between the desired input parameter and the actual motion parameter derived from the blended motion. Depending on the type of applications, this error may be of less importance: for application that requires precise end-effector control such as reaching, punching and kicking a given target, the parameter error would directly determine whether the result is valid for a given task; for abstract motion parameterization space such as emotion (happy walk v.s. sad walk) or style (walk forward v.s. walk sideways) control, only qualitative aspects of motion are of interest.

Computation Time is divided into pre-computation phase and run-time computation phase. Pre-computation time is the amount of time taken for a method to build the necessary structures and required information to be used in the run-time phase. While this may usually be negligible for Inverse Blending or Barycentric, it may require significant amount of time for KNN and RBF depending on the number of pseudo examples or size of the dataset. A method require little to no pre-computation is more flexible in changing the example data on-the-fly, which can be beneficial for applications that require on-line building and adjusting motion examples [CHK10]. Run-time computation phase is the time required for the method to compute the blending weights based on given input parameters, which reflects the real-time performance.

Smoothness determines whether the blending weights would change smoothly when

category	number of examples	parametrization	joint paramtrized	note
Reach	24	$p=(x,y,z)$	wrist	full body reaching with bending down and turning around
Punch	20	$p=(x,y,z)$	wrist	start and end with fighting stance; targets are mostly in front
Kick	20	$p=(x,y,z)$	ankle	start and end with fighting stance; p is ankle position at kick apex
Jump	20	$p=(d,\theta,h)$	base	d: jump distance; θ: jump direction; h: max height during jump
Locomotion	20	$p=(v_f,\omega,v_s)$	base	v_f: walk forward speed; ω: turning rate; v_s: walk sideways speed

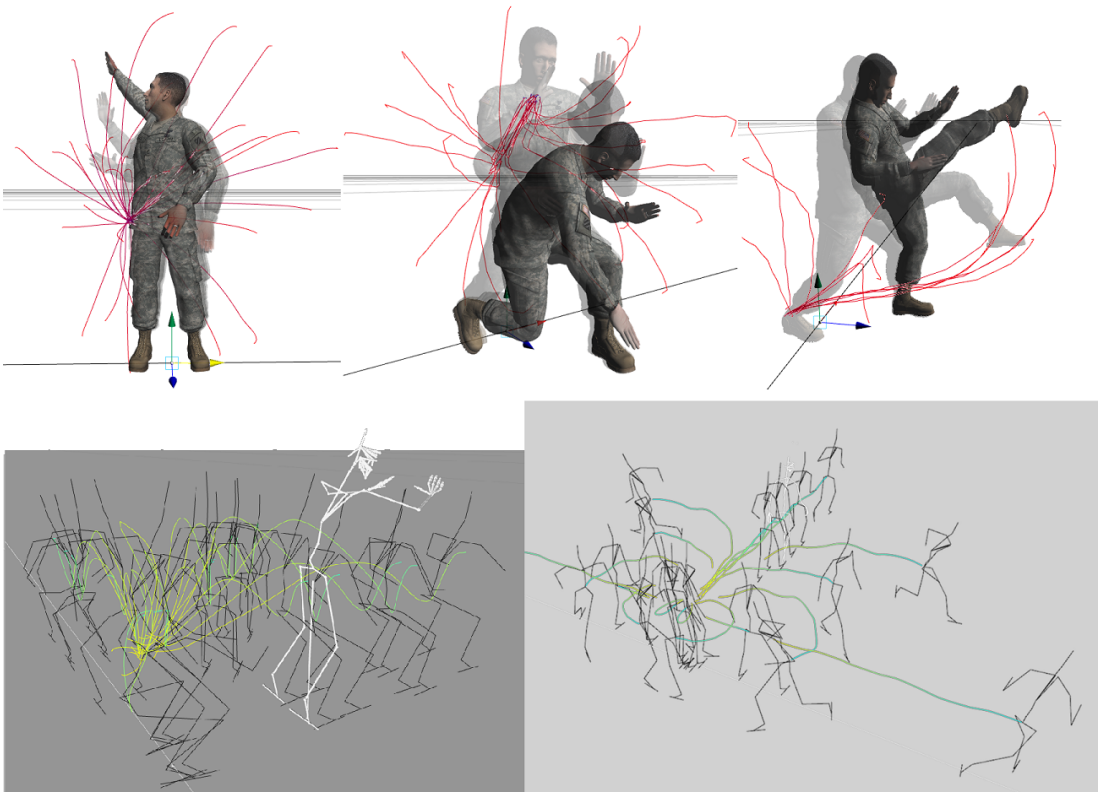


Figure 4.1: An overview of the motion capture dataset used for our analysis. Top row from left to right: reach, punch and kick. Trajectories of character's end-effector are plotted. Bottom row: jump and locomotion. Trajectories of the root joint are plotted.

motion parametrization varies. This metric is of more importance when parameters are changed frequently during motion synthesis. Specifically speaking, smoothness may be less required for motions like reach, kick and punch where parametrization usually stays constant during each action execution. However it is critical for other motion parametrization such as locomotion where parameters may need to be changed continuously even within each locomotion gait. And for such applications, jitter artifacts would occur and degrade the quality of synthesized motions if smoothness cannot be guaranteed. The smoothness of the blending weights is numerically defined as the curvature of the blending weights $w_{x,y}$ over a $m \times m$ surface grid G.

$$G = \{p = (x, y) | (0 \leq x \leq m, 0 \leq y \leq m)\}$$

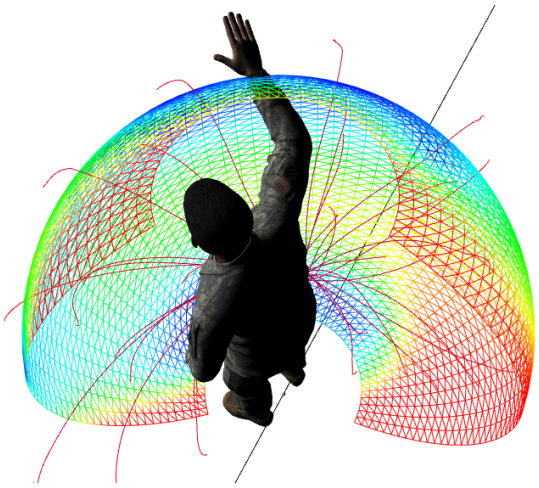


Figure 4.2: Parametric space for reaching dataset.

For a given G, the curvature $\kappa_{x,y}$ is computed at each grid vertex $p = (x, y)$ as :

$$\kappa_{x,y} = \frac{1}{8} \| \sum_{a=-1}^{1} \sum_{b=-1}^{1} (w_{x,y} - w_{x+a,y+b}) \|$$

This curvature is computed over several grids to uniformly sample the volume within the 3D parametric space and use the average curvature $\bar{\kappa}$ as the smoothness metric.

Motion Vector Flow: This chapter also proposes visualizing the smoothness (visual quality) of the final synthesis with motion vector flows: each vector denotes the absolute movement of a particular skeleton joint (or pseudo-joints filling in the gap of longer bones) as it traverses the 3D workspace between two consecutive motion frames. Distinguishable colors are assigned to the vectors representing sudden change in vector length compared against local average of the length computed with a sliding window, thus highlighting the abnormal speed-ups (warm color) and slowdowns (cool color) caused by jitters and such. Fig 4.5 shows the motion vector flow from 150-frame locomotion sequences generated by 4 blending methods, each showing a character transitioning from slow walk to jogging over the same course of variations inside parametrization space. Motion frames are selectively plotted with stick figures on top of the vector flow.

4.5 Results and Discussions

For each method, we uniformly sampled inside the parametric space and measured the obtained errors. Since the parametric space for reach, punch and kick naturally coincides with the 3D workspace, the parameter point $p = (x, y, z)$ was sampled over a spherical surface and compute the error as the euclidean distance between the target

p and where the wrist/ankle actually reaches, see Fig 4.2. For jump and locomotion where the parametric space represents abstract values such as turning angle and speed, the parameter point was sampled on a rectangular grid.

4.5.1 Parametric Error Comparison

The parametrization accuracy visualizations for each method are shown in Fig 4.3. The first 3 rows showing the result for reach, punch and kick respectively, and the surface used to sample p is to fix parameter z (distance from the character) in mid-range of the dataset coverage. Similarly for jump and locomotion (row 4 and 5), jump height h and sideways speed v_s (see Chapter 4.4) are chosen respectively in mid-range. InvBld by comparison tends to be the most accurate as it relies on numerical optimization to find blend weights that yield minimal errors. KNN also performs relatively well as it populates the gap in parametric space with pseudo examples to effectively reduce the error. Thus for applications that require high parametrization accuracy such as reaching synthesis, it is preferred to apply either InvBld or KNN with dense data. On the other hand Barycentric and RBF numerically tend to generate less accurate results, however this does not necessarily mean the motions generated are of poor quality. In fact, as human eyes are more sensitive to high frequency changes than to low frequency errors, Barycentric and RBF are able to produce reasonable motions for locomotion and jumping, which are parameterized in the abstract space. The table and chart in Fig 4.6 (left side) lists the average parametric error using results from a more densely sampled parametrization space ($60 \times 60 \times 5$ samples on average).

4.5.2 Smoothness Comparison

Although InvBld outperforms in parametrization accuracy, it falls behind in terms of smoothness, which can be observed both visually (Fig 4.4) and numerically (Fig 4.6 right side). By comparing the error and smoothness maps with other methods, it can be observed that there are several semi-structural regions with both high errors and discontinuity in smoothness. Depending on the initial condition, InvBld optimization procedure may get trapped in local minimal at certain regions in parametric space, which results in high error and discontinuous regions shown in column 3 of Fig 4.4 and 4.3. KNN also suffers from similar smoothness problems (Fig 4.4 column 2), and since KNN requires a dense set of pseudo examples to reduce parametric errors, the resulting parametric space tends to be noisier than others. Moreover, for KNN and InvBld, there can be sudden jumps in blending weights due to changes in the nearest neighbors as the parametrization changes, leading to the irregular patterns in the smoothness visualizations (Fig 4.4).

Barycentric produces a smoother parameterization as the blending weights only change linearly within one tetrahedron at any given time. However obvious discontinuities occur when moving across the boundaries between adjacent tetrahedra. Note that although both KNN and Barycentric interpolation have similar numerical smooth-

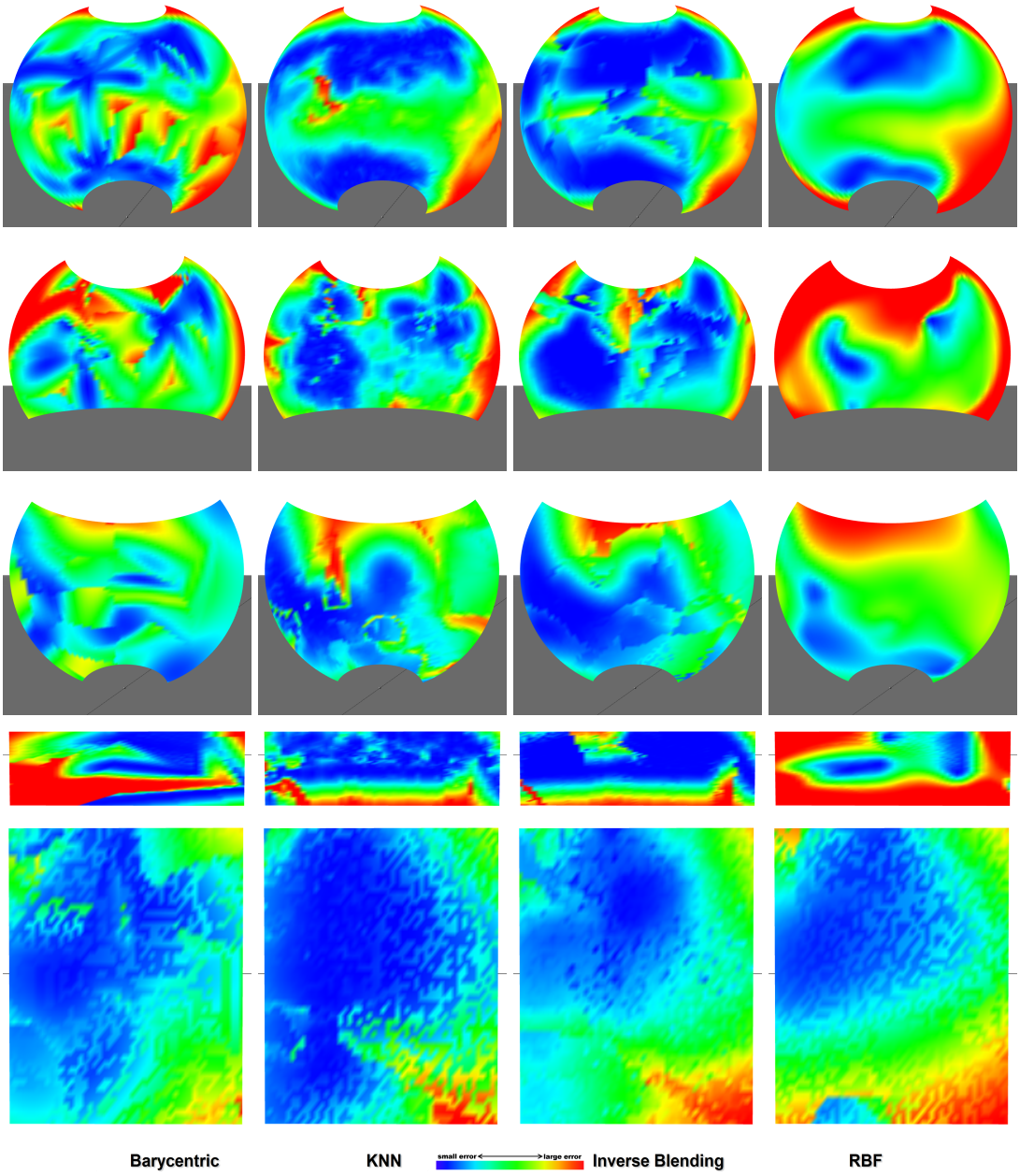


Figure 4.3: Parametrization accuracy visualizations for 4 blending methods on different motion dataset. From top row to bottom are reach, punch, kick, jump and locomotion; from left column to right are: Barycentric, KNN, InvBld and RBF.

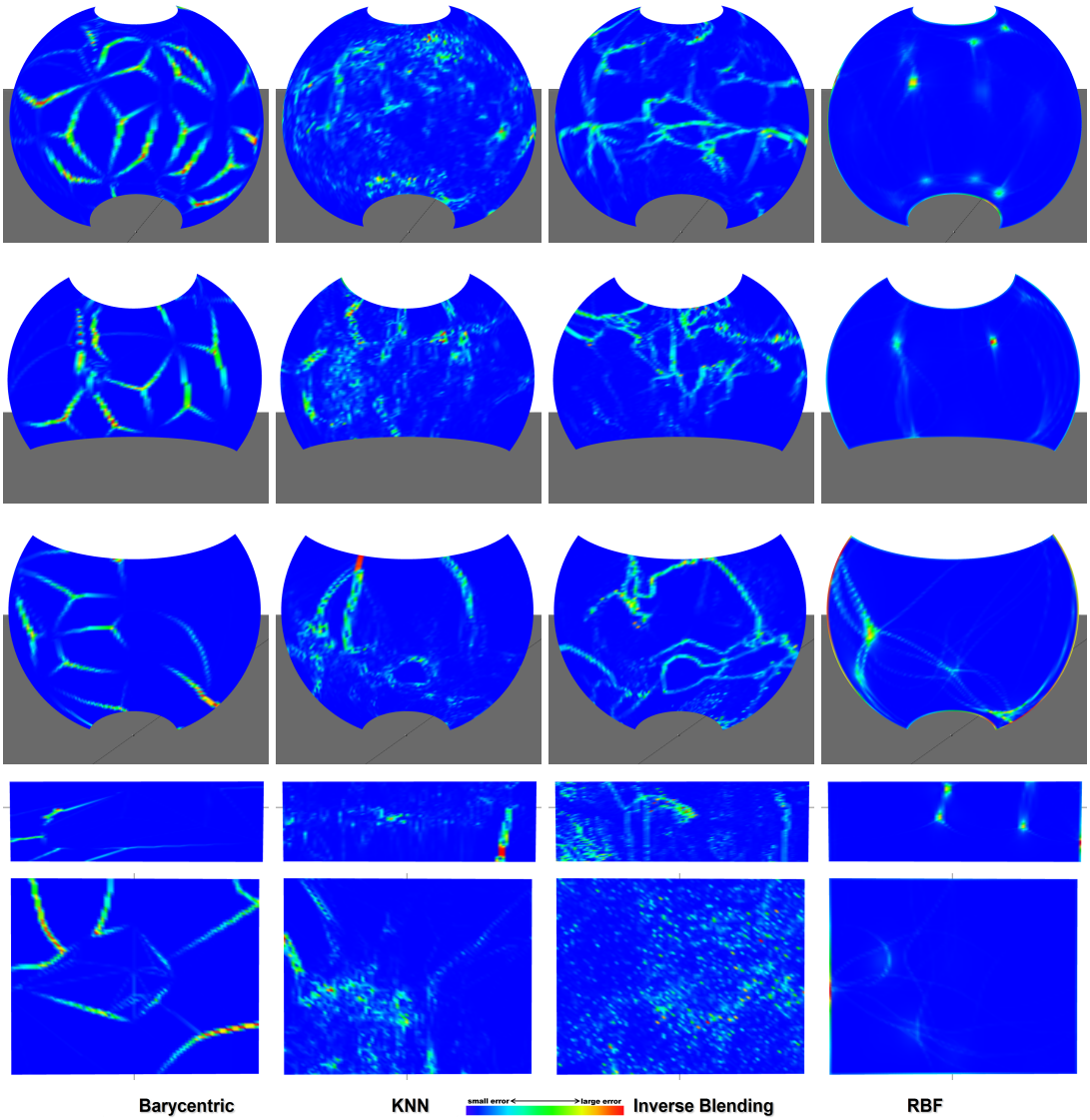


Figure 4.4: Smoothness visualizations for 4 blending methods on different motion dataset. From top row to bottom are reach, punch, kick, jump and locomotion; from left column to right are: Barycentric, KNN, InvBld and RBF.

ness in certain cases, the resulting motions from Barycentric usually look more visually pleasing. This is because the weight discontinuity is only visible when moving between different tetrahedra for Barycentric, while for KNN the irregular blending weights could cause constant jitters in the resulting motions. Finally, RBF tends to generate the smoothest result visually and numerically, which may be a desirable trade-off for its high parametric error in certain applications. Low performance in numerical smoothness corresponds to low visual quality of the final synthesis, as shown in Fig 4.5 and also the accompanied video where more jitters and discontinuities can be observed.

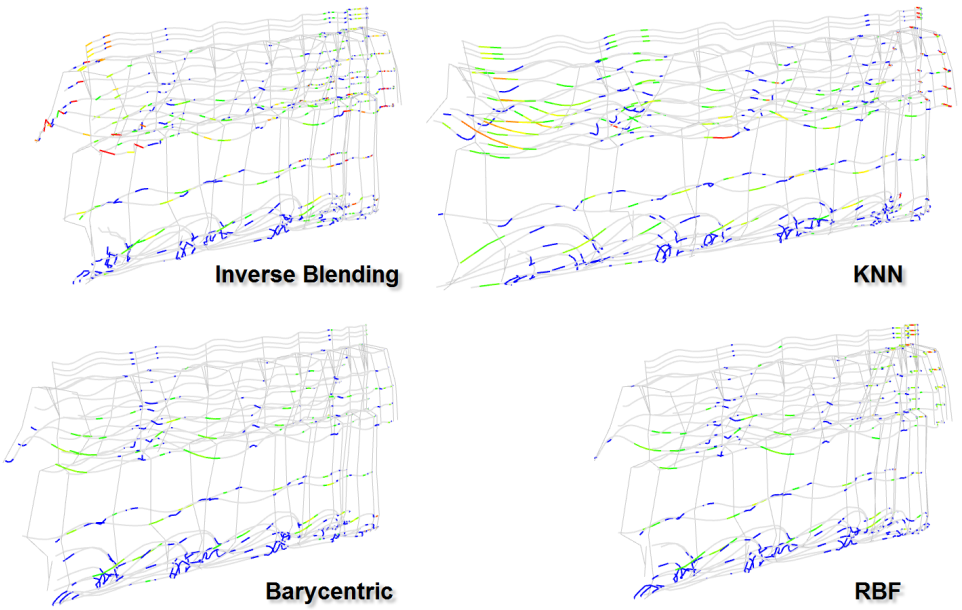


Figure 4.5: Motion vector flow visualization of a 150-frame locomotion sequence transitioning from slow walk to jogging. Color segments indicates jitters and unnatural movements during the sequence. By comparison results from InvBld and KNN (top row) contain more jitters than Barycentric and RBF (bottom row).

4.5.3 Computation Time Comparison

KNN requires more pre-computation time than other methods for populating the parametric space with pseudo-examples as well as constructing a k-D tree to accelerate run-time efficiency. Moreover, whenever a new motion example is added, it needs to re-build both pseudo examples and k-D tree since it is difficult to incrementally update the structures. This makes KNN less desirable for applications that require on-line reconstruction of new parametric space when new motion examples are added. RBF on the other hand can usually be efficient in dealing with a small number of examples, however the cost of solving linear equations increases as dataset gets larger. Barycentric requires the tetrahedra to be either manually pre-specified or automatically computed, and may become less flexible for high dimension parametric space. InvBld by

comparison is more flexible since it requires very little pre-computation by moving the computational cost to run-time.

evaluation / methods	average parametrization accuracy					average smoothness					average compute time
	Reach	Punch	Kick	Jump	Locomotion	Reach	Punch	Kick	Jump	Locomotion	
Barycentric	0.2060188	0.2351264	0.4723036	0.3499212	1.3287728	0.0377356	0.0248206	0.0224772	0.0157104	0.0212154	0.625 ms
KNN	0.1676776	0.177986	0.3386378	0.1629752	1.0714436	0.0376208	0.0318934	0.0291025	0.0241884	0.0182888	0.221 ms
InvBld	0.1511574	0.17205	0.3217224	0.0756766	1.2572206	0.0698706	0.0785622	0.0801812	0.0609856	0.0602558	4.394 ms*
RBF	0.241626	0.325049	0.5883928	0.3816458	1.7279502	0.0035818	0.0023684	0.0021972	0.001998	0.0012648	0.228 ms

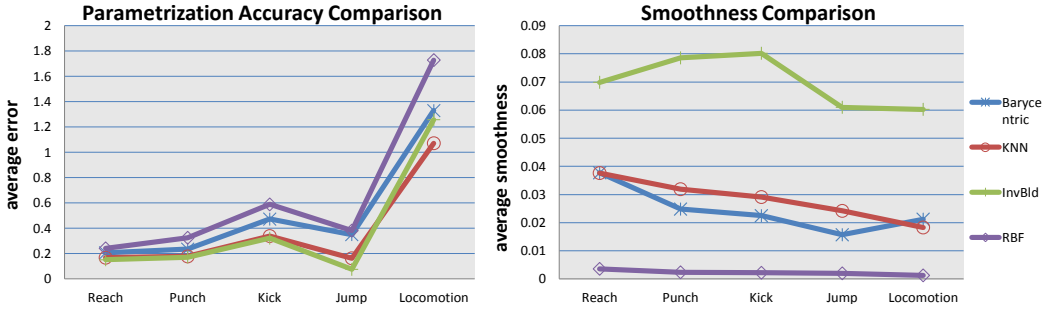


Figure 4.6: Parametrization accuracy (unit: centimeters) and smoothness comparison chart across four blending methods on different motion sets. * Computation time (unit: milliseconds) measured on a machine with Quad Core 3.2GHz running on single core. InvBld can expect $2 \sim 3X$ speed-up with optimized code on kinematic chain updates [HK10].

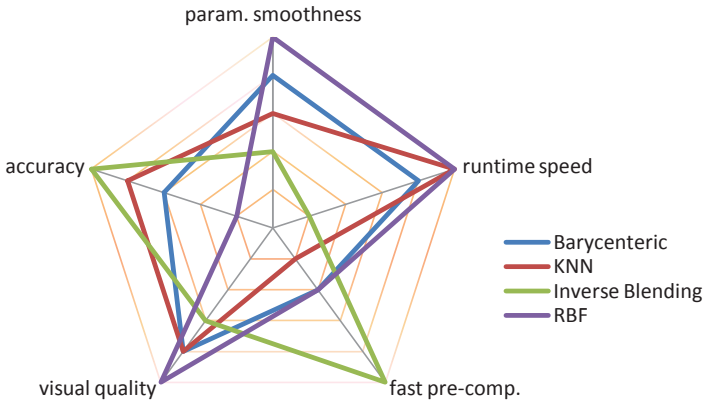


Figure 4.7: Performance overview across four blending methods. Note that the measurements in this figure are not to scale.

For run-time performance, all methods can perform at interactive rate, see last column of the table in Fig 4.6. However, InvBld is significantly more expensive than other methods as it requires many numerical iterations with kinematic chain updates to obtain optimal results. Also, the computation time greatly depends on the initial estimation of the blending weight and therefore may have large variations across different optimization sessions, posing big challenges on its real-time performance for multi-characters simulations. The other methods require only a fixed number of operations (well under 1 millisecond) and are much more efficient for real-time applications.

The overall performance for each blending method is summarized in Figure 4.7. In terms of parametric error and smoothness, InvBld has the most precise results but is poor in smoothness. On the opposite end, RBF produces the smoothest parametric space but is the least accurate method as a trade-off. KNN and Barycentric fall in between with KNN being slightly more accurate and less smooth than Barycentric. In terms of computation time, KNN requires the most pre-computation while InvBld requires none. RBF and Barycentric require some pre-computation and may also require user input to setup tetrahedra connectivity or fine tune the radial basis kernel. Therefore InvBld is most suitable for on-line update of motion examples, with the trade-off being most expensive for run-time computation while the other methods are all very efficient at run-time.

These performance results suggest that there is no method that works well for all metrics. To gain advantages in some metrics, a method would need to compromise in other metrics. InvBld and RBF show a good example of such compromise that are in the opposite ends of the spectrum. Overall, for applications that do not require high accuracy in parametric errors, RBF is usually a good choice since it is mostly smooth, easy to implement, and relatively efficient both at pre-computation and run-time. On the other hand, if parametric accuracy is very important for the application, InvBld provides the best accuracy at the cost of smoothness in parametric space. KNN and Barycentric fall in-between the two ends, with Barycentric being smoother and KNN being more accurate. Note that KNN may require much more pre-computation time than other methods depending on how dense the pseudo-examples are generated, which may hurt its applicability in certain interactive applications.

4.6 Conclusion

This chapter presents an in-depth analysis of four different motion blending methods. The results show that there is no one-solves-all method for all the applications and compromises need to be made between accuracy and smoothness. This analysis provides a high level guidance for developers and researchers in choosing suitable methods for character animation applications. The metrics defined in this chapter would also be useful for testing and validating new blending methods. A new motion blending method that satisfy or make better compromise at both parametric error and smoothness would be desirable for a wide range of applications.

As future work, we would attempt to include methods such as geostatistical interpolation [MK05] for the completeness of this comparison. We expect it to perform in a more balanced way between accuracy and smoothness, however with the sacrifice of computation time both in pre-processing and at run time.

CHAPTER 5

An Immersive Platform for Motion Modeling

Virtual humans are widely used in interactive training, education and therapeutic applications. However, building animations which are both realistic and with parameterized variations in respect to a given scenario remains a complex and time consuming task. In certain situations, only experts in a given training subject may be able to demonstrate how specific motions have to be performed. Intuitive user interfaces for motion modeling via direct demonstration are therefore useful as they allow users to easily demonstrate and parameterize motions inside given scenarios.

This chapter presents a motion modeling platform that is based on the interactive construction of motion databases that can be parameterized for later use in training applications. Based on low-cost wearable motion sensors, or any other suitable motion tracking devices, the platform allows the creation of new motion demonstrations on-the-fly, playback, as well as motion editing and parameterization. Together with tools providing different forms of visual exploration of database coverage, it could assist in refining the coverage of the current set of motions inside the virtual environment. This framework has been designed for use in immersive visualization systems, achieving a powerful and intuitive approach for programming generic parameterized motions by demonstration.

5.1 Introduction and Related Work

The motivation of building this platform is to facilitate the process of programming virtual agents in order to achieve effective virtual assistants that can learn, train and assist people in interactive applications. Due to the great need of variations and precise control of actions and gestures in many scenarios, modeling and parameterization of realistic motions for virtual agents become key problems in a wide range of applications. Common solutions rely on either hand-crafted motions [TMM08, GKK03, SDO04] with commercial modeling tools, or gesture synthesis with algorithmic procedures such as Inverse Kinematics [KW04, Kal08]. However it remains difficult to achieve both controllable and realistic results, and every attempt to solve the problem purely algorithmically will require specific adaptations and models for every action and situation being modeled. On the other hand, motion blending techniques with motion capture data [RBC98, RSC01, KG04, MK05] provide powerful interpolation approaches for parameterizing pre-defined example animations according to high-level characteristics. While intensive research has been dedicated to find suitable interpolation schemes

and/or motion style controls, less attention has been given to the development of techniques to enable intuitive interfaces for building suitable motion databases, that can well cover the simulated workspace with dedicated blending and parameterization procedures. This is especially important for tasks that require parameterizations in respect to spatial constraints within the environment. For instance, the interactive construction of real-time motion controllers has been proposed before [CHP07], but without the inclusion of immersive interfaces for interactive editing and visualization of the obtained models.

Our immersive modeling solution has been recently introduced in [CHK10], which allows interactive motion modeling approach via direct demonstration implemented on an immersive multi-tile stereo visualization facility, shown in Fig 5.2. The system can be operated in two distinct phases: in the *modeling phase* the expert, who has the specialized knowledge of how to correctly perform the required motions, will demonstrate the needed motions to our system interactively. Later, in the *training phase*, by relying on the database of motions previously collected from the expert, the virtual human trainer is then able to reproduce the motions in interactive sessions with apprentice users learning the training subject, with the ability to reproduce the motions in respect to arbitrary target locations inside the environment.

While this approach has proven to be effective, achieving an interactive modeling interface that is useful in concrete applications still remains a critical problem. In particular for the motion modeling phase, intuitive interfaces are important to allow expert trainers to focus on the motions being demonstrated rather than on irrelevant details. A good set of tools for inspection, parameterization and testing is also very important in order for users to effectively build suitable motion databases. Existing motion capture interfaces hardly offer any of these features, and in most cases the user is required to work tedious hours in front of a computer using the keyboard and mouse to perform post-processing operations after the capture session.

This chapter presents a solution to improve the modeling phase of our existing motion modeling framework. Our previous framework targets at the interactive capture of scenario-specific motion examples. With the new platform, users not only can easily playback the demonstrated motions inside the virtual scenario for inspection of captured data, but also perform key editing functions needed for populating parameterized databases (or motion clusters). Furthermore, tools are provided to visualize the spatial coverage of each motion cluster inside a simulated workspace, guiding the on-line programming of scenario-specific examples.

5.2 System Overview

The motion modeling phase of our system is designed as follows. While the user demonstrates new motions on-the-fly, the user can also immediately playback and edit each captured segment of motion. The user can then quickly observe the coverage of the database inside the virtual environment with different visualization tools, allow-

ing the user to improve the database coverage as needed. The coverage of a database here refers to how well parameterized motions interpolated from the discrete motion examples in the current database are able to satisfy precise spatial constraints in the environment. Figure 5.1 outlines the main modules of our system.

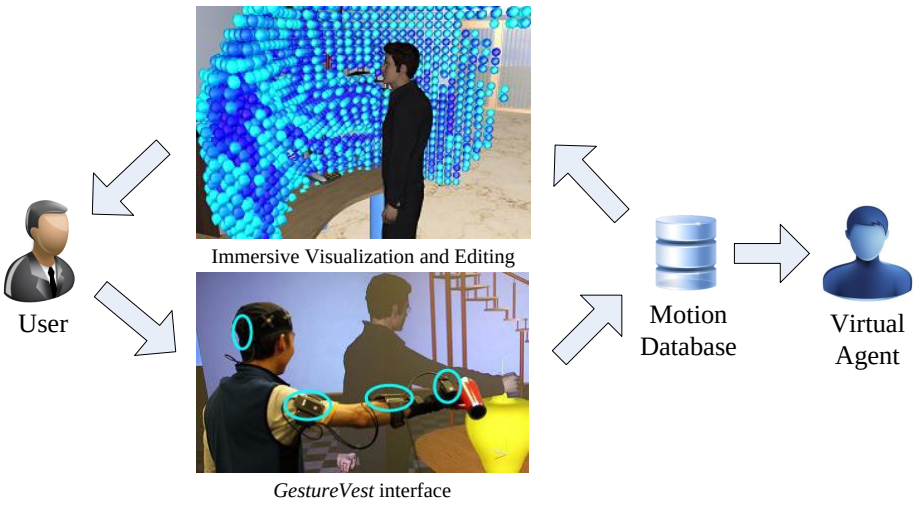


Figure 5.1: The pipeline for the modeling phase of our platform.

The platform models each parameterized gesture or action of interest with a cluster of example motions of the same type, but with variations in respect to the spatial parameterization to be considered. For example, a gesture cluster for a certain way of pointing will typically consist of several examples of similar pointing gestures, but with each pointing motion pointing to a different location.

Figure 5.1 shows one example scenario where the expert is modeling a pouring motion cluster on the platform. For the illustrated pouring action the spatial parameterization is the target container, which can be placed anywhere on the table in the scenario. By providing a few pouring actions for key locations on the table, example motions are interpolated to reach arbitrary targets on the table. This technique is used to parameterize a captured motion cluster so that blended motions can meet given spatial constraints, such as target locations for the end-effector of the character to reach, point, pour, etc. This offers precise control of end-effectors during the execution of diverse upper-body actions, which is crucial for demonstrative gestures and actions as the arbitrary positions and specific targets need to be referred to as part of the information to be delivered.

Figure 5.2: Example scenario (parameterized water pouring motions) of our motion modeling phase running on the UC Merced PowerWall system. Motions are captured on-line with *GestureVest*.

5.3 Interface Description

5.3.1 On-line Motion Capture

The platform targets at the situation where the user, normally an expert in the training subject, is able to model the needed actions and gestures via direct demonstration, who might not have previous experience with the system. We mainly focus on capturing upper-body gestures and actions performed with single arm/hand. The *GestureVest* [HK09b] is used for this capture process, which was specifically developed for this purpose. It contains a set of low cost wearable motion sensors, a data glove for capturing finger movements, and a WiiMote controller used to remotely operate the capture application. The vest captures 13 Degrees of Fredom (DOFs) of upper-body (single-arm) movements and the employed data glove captures 14 DOFs finger movements. Details are provided in Chapter 9. Note that other motion tracking devices could also be used here, such as Vicon, Microsoft Kinect and Sony Move. Fig 5.6 shows an example where Vicon tracking system is used to tracking user's movements.

We focus on modeling motions for interactive training applications that require gestures and actions to be reproduced realistically and with precise parameterizations in respect to spatial constraints in the environment. This setup enables the interactive customization of gestures needed for programming interactive virtual demonstrators for a broad range of applications. The definition of clusters for collecting example motions is an important concept of the system. It is necessary for specifying each parameterized action or gesture. When the user chooses to start a new cluster, every recorded motion becomes associated with that cluster. Motions in the same cluster will be blended together during the training phase and therefore they must be consistent yet able to represent the variations within the motion type. For instance, a pointing cluster con-

tains several pointings of the same type but each pointing to a different location in the environment. The capture process is very straightforward. The user first initializes a new motion cluster, and then holds down the record button on the WiiMote to begin capturing, and release the button when the performing is done. Motions are captured at 60 frames per second, and stored in memory for later editing.

5.3.2 Playback and Editing Mode

Using a WiiMote controller, the user can instantly switch from on-line capture mode to playback/editing mode, scroll through the already captured motions, delete unsatisfactory clips from memory, mark the start and end of each motion segment to crop out unnecessary parts, and most importantly mark the stroke frames (stroke times) for each segment before populating them into the database.

Since great variations can be found among different gestures and actions, the expert who performs the demonstrations needs an easy way to go through each motion cluster and to quickly crop lead-in/lead-out frames and annotate the motion strokes. For that purpose, buttons on the WiiMote are used to select captured motions and perform the editing functions. We define an editing interaction mode where the trajectory of the user's hand is captured with the vest and mapped into a linear horizontal movement in real time, and the movement is then mapped directly to the motion playback slider. This enables the user to intuitively scroll through the motion playback with a simple horizontal hand movement, allowing the user to remain inside the captured area and conveniently validate and edit the motion cluster (instead of operating interfaces based on keyboard and mouse in front of a far away computer), and this has shown to be important for maintaining consistency of the capture quality. Figure 5.3 shows several snapshots of the playback interface being used to analyze a recently captured pointing motion.

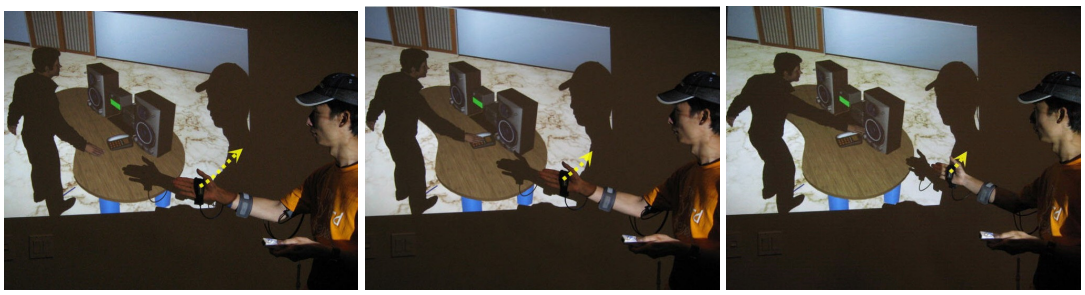


Figure 5.3: Our interface allows the user to easily scroll through captured motions, crop out unwanted frames, and most importantly mark stroke frames to be parameterized. This image shows the user inspecting one motion segment controlled by hand movement.

Note that the stroke time [TMM08] is in particular addressed here due to its main importance for parameterizing demonstrative gestures and actions, as each motion seg-

ment forming a cluster will be time-warped such that all motions have their strokes time-aligned in order to achieve a meaningful interpolation of the example motions. Automatic procedures are also included to assist the user. For example, the stroke frame for a pointing gesture is often the zero-crossing of the velocity vector of the finger tip motion. Several other phases can also be marked for time alignment in order to guarantee better blending results. For complex motions, multiple stroke frames may be annotated for each motion inside the cluster in order for them to be correctly parameterized.

5.3.3 Motion Parameterization

The Inverse Blending technique [HK10] is used to parameterize the motion database created during the previous motion modeling step in order to meet desired spatial constraint parameterization C. This technique offers precise control of constraints such as end-effector positions, joint orientations or a combination of different constraint types during the execution of diverse upper body actions.

When new motions need to be synthesized, the system first selects a subset of k example motions which are closer to the given constraints. The used distance metric (or error metric) will vary according to the type of constraint. Based on the distance values from each of the selected k example motions initial weights for motion interpolation are assigned. Then the inverse blending procedure will optimize the weights until an optimal set of blending weights with respect to C is determined and is used to synthesize the final motion best satisfying the given constraints. Multiple C can be combined and enforced at the same time, allowing additional ways of motion parameterizations. Details are provided in Chapter 3, and an in-depth analysis of 4 motion blending algorithms is also provided in Chapter 4.

5.3.4 Database Spatial Coverage Visualization

The ability to enforce constraints using inverse blending greatly depends on the existing variations among the example motions being interpolated. In general, the size of motion database is proportional to the volume of the workspace.

In order to produce quality motions satisfying many possible constraints spanning the whole workspace, it is important to determine which example motions to capture during the capture process. This will ensure that a well-built cluster of motions is formed, with good coverage of the regions of interest (ROIs) inside the workspace.

On the other hand, defining an overly fine subdivision of the constraint space with too many examples is inefficient and impractical as it requires capturing too many example motions to populate a database. Not only the database would be redundant, this would also impose a huge workload on the user. Instead, since similar examples can often be interpolated to produce valid new motions with good quality, a small number of carefully selected example motions is better in providing good coverage for the

ROIs in the workspace. Achieving an efficient database is also key to ensure lag-free interactivity of the system.

The approach of using a palette of colors inside the workspace as visual guidance for ergonomic design was proven to be informative in previous work [YSA08, RPB03, ZHH10]. We propose two specific visualization methods rendering a palette of colors [YSA08, RPB03, ZHH10] inside the workspace to intuitively guide the user during the process of adding new motions to refine the database for improved coverage: Workspace Volume Visualization **(WV)** and Local Coverage Visualization **(LV)**.

5.3.4.1 Workspace Volume Visualization

WV conducts a coarse uniform sampling of the workspace and presents the overall spatial coverage with colored cubes for the entire workspace without the need to define an overly fine subdivision of the constraint space. Each cube represents a reaching target (spatial constraint), and a motion synthesized towards each cube is measured by reaching precision (error e^*) using a constraint evaluation function, and the value $e^*/e_{max}(\in [0,1])$ is mapped onto a hue color space then assigned to each cube. For a reasonably sized database WV takes a few seconds to generate, then the user can immediately spot areas with low coverage by the color of the cubes (red or orange), and add additional motion towards these areas.

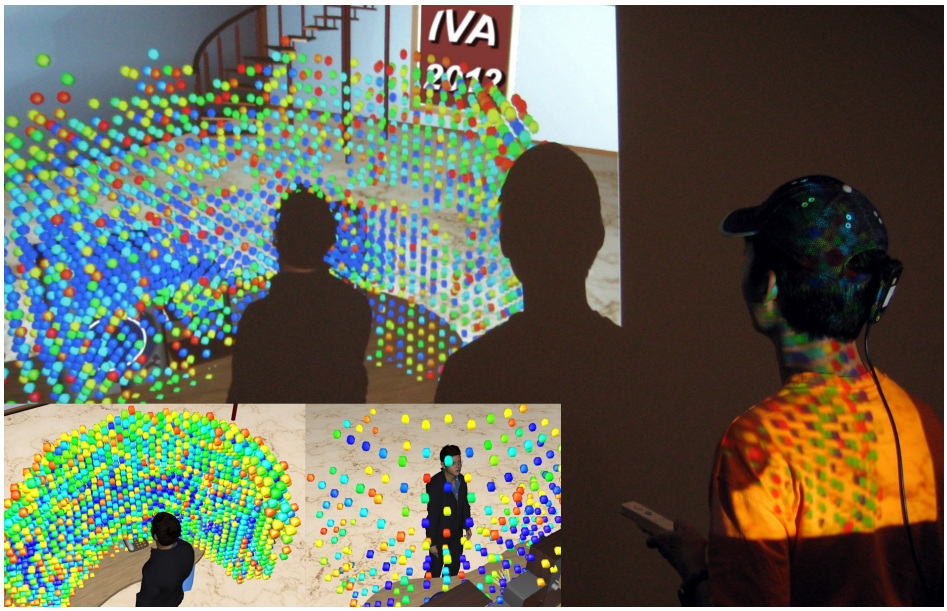


Figure 5.4: Workspace Volume Visualization mode gives an overview of database coverage. The sampling density and error threshold can be adjusted for clear viewing. User intuitively adjusts the viewing angle by moving his head.

5.3.4.2 Local Coverage Visualization

In certain cases, the global error-based volume visualization is not needed when the user is fine tuning the coverage of a small region, or when only a small local region is of interest. In addition, the pre-computation time can impose undesirable lags when editing large motion sets. These can be solved with LV.

LV renders a transparent colored mesh geometry covering a small ROI, delimiting the coverage evaluation within its volume. It focuses on the local coverage visualization taking only milliseconds to be computed, and it is suitable for fine tuning coverage of smaller volumes when only small local regions are of interest. LV uses the same color mapping as WV, and error ratio is assigned to the corresponding vertex. Color on the mesh surface comes from Barycentric color interpolation with Gouraud shading.

LV follows the movement of the user's hand, its size, shape and resolution can be iteratively changed with the WiiMote controller for either fast sweeping over large ROIs (a table surface) or for carefully checking small ROIs (buttons, etc). LV is also able to utilize motions dynamically added to the database in real-time applications without any pre-computation lag. Please refer [HK10] and Chapter 3 for details on motion synthesis and error evaluation with spatial constraints.

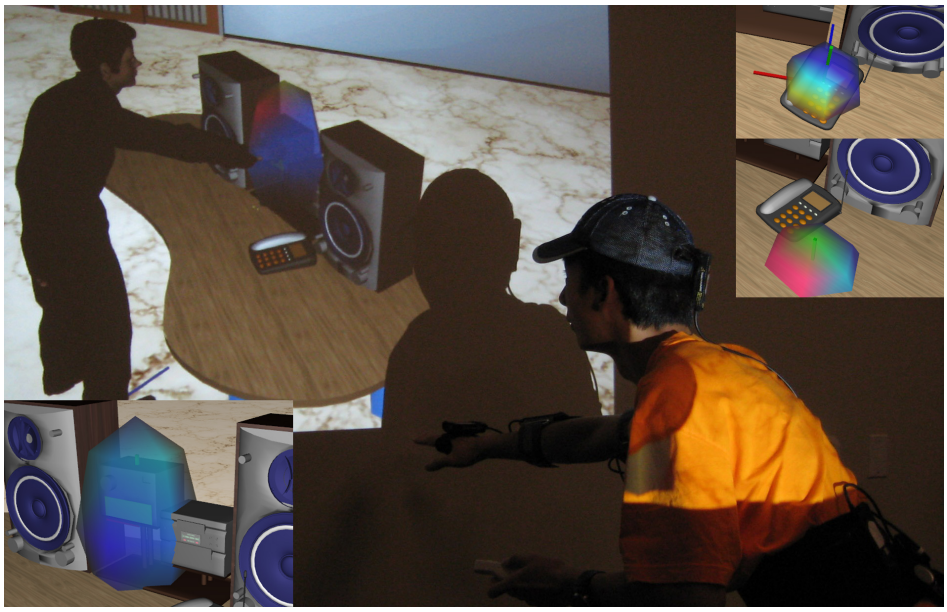


Figure 5.5: Local Coverage Visualization mode. The rendered surface follows the movement of user's hand, ideal for checking small ROIs like dials and buttons.

5.4 Conclusion

The proposed platform has been experimented with several scenarios and the overall approach constitutes a powerful approach for programming virtual agents. The *GestureVest* enables the user to easily capture motion clusters and perform motion editing functions while immersed in the target virtual scenario. The coverage visualization tools effectively allow the user to observe the database coverage immersively within the workspace and during the on-line collection of example motions. The interactive visualization is able to guide the user to concentrate on capturing motions where they are needed, in regions with less coverage. The user can thus intuitively construct suitable databases for parameterized actions or gestures with guaranteed coverage within the specified precision. The platform could adapted different hardware configurations. For instance, Figure 5.6 illustrates the local coverage visualizer following the movement of a WiiMote controller tracked by a fixed optical tracking system.

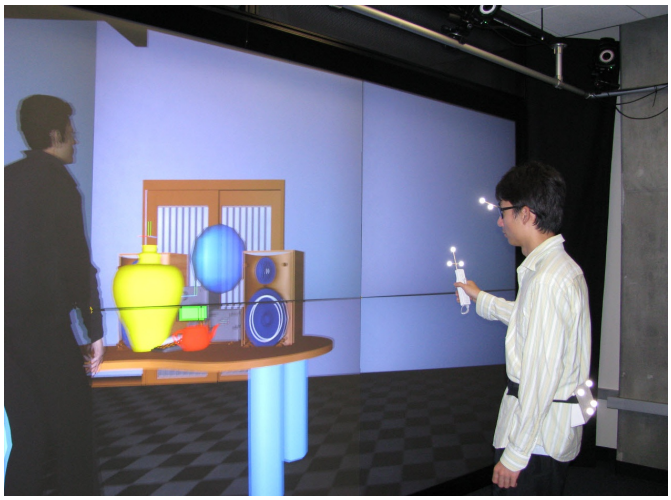


Figure 5.6: The platform integrated with Vicon tracking device with viewer-dependent rendering to provide truly immersive motion modeling experience.

This platform could significantly improve the process of programming interactive virtual humans for inside immersive training scenarios. Figure 5.7 shows a sequence of snapshots during a on-line modeling process. First, the user quickly examine the workspace coverage with the Local Coverage Visualization tool. The control panel region happened to be of low coverage (red color), and as a result, a pointing motion towards the target location (specified by the yellow cone) cannot be precisely synthesized. The user switched back to modeling mode and demonstrated a few new examples, without the need to sit down and operate conventional input devices like keyboard and mouse. Immediately the new examples became part of the database without any pre-computations. And several synthesis afterwards confirms that the spatial coverage for the ROI in particular has been greatly improved (showing blue color). As future work, we plan to integrate a better GUI system and apply this platform on a concrete

Figure 5.7: The sequence demonstrates the process of on-line modeling and dataset refining. Top row: user spotted ROI with low coverage, note the red color shown by the LCV tool. Middle row: user teaches agent new motions towards the low coverage region via direct demonstration; Bottom row: coverage is improved for this ROI, as a result the agent could better synthesize pointing motions towards this ROI. Note the color shown by the LCV tool turned into blue, indicating improved local coverage.

modeling application, physical therapy as an example. Human participants would then be invited to evaluate the usability of the system.

CHAPTER 6

Human-like motion planning in Blending Spaces

This chapter introduces an approach for enabling sampling-based planners to compute motions with humanlike appearance. The proposed method is based on a space of blendable example motions collected by motion capture. This space is explored by a sampling-based planner that is able to produce motions around obstacles while keeping solutions similar to the original examples. The results therefore largely maintain the humanlike characteristics observed in the example motions. The method is applied to generic upper-body actions and is complemented by a locomotion planner that searches for suitable body placements for executing upper-body actions successfully. As a result, our overall multi-modal planning method is able to automatically coordinate whole-body motions for action execution among obstacles, and the produced motions remain similar to example motions given as input to the system.

6.1 Introduction

Despite several successes in the motion planning domain, achieving whole-body humanoid motions with humanlike characteristics remains a challenge. One main difficulty that emerges from this problem is to strike the right balance between how much to explore solutions during motion planning and how much to constrain the search space in order to obtain solutions with humanlike characteristics.

Our proposed method starts by decomposing the problem into a multi-modal planning problem, where basic skills are planned individually and also coordinated with each other. This approach facilitates addressing the specific needs of each skill, and is inspired by how humans may solve real motion planning problems. This chapter addresses the case of planning motions composed of two skills, locomotion and generic upper-body actions, and example motions from motion capture are used as a way to build search spaces with humanlike characteristics.

For example, consider the simple scenario where a character walks toward a light switch and turns it on, as shown in Fig. 6.1 and 6.10. Solving this problem requires a locomotion planner able to place the character in a suitable location near the switch, and then an upper-body action is required for reaching and pushing the light switch. Our overall method solves such class of full-body motion problems.

Upper-body actions are synthesized with a novel motion planner that searches for solutions in a space of blendings between example motions. The method produces

Figure 6.1: Our overall planning approach achieves precise end-effector placement for action execution among obstacles and produces humanlike results in coordination with locomotion.

collision-free motions to precise targets and achieves solutions with humanlike characteristics similar to the ones observed in the original example motions. The planner is therefore limited to explore the variations embedded in the example motions, and a suitable body placement is essential for enabling the upper-body planner to be successful.

The upper-body planner is thus complemented with a standard locomotion planner based on a motion graph. The locomotion planner will search for suitable body placements in coordination with the upper-body planner until an overall whole-body motion for performing the target upper-body action is found.

As a result, our combined approach is able to automatically coordinate locomotion with generic actions, and the produced motions are realistic, collision-free, and can precisely interact with the environment. Our overall method is able to address a broad range of real-life tasks and is therefore useful to a number of applications in ergonomics, training, education, entertainment, and also humanoid robotics.

6.2 Related Work in Motion Planning

Traditional motion planning methods [Lat90, Lau98, LaV06] are based on the systematic search of configuration spaces. Among the several techniques, sampling-based methods [KSL96, LaV98, KL00b] have become extremely popular for planning in continuous configuration spaces. Such methods are also popular for planning humanoid motions, in particular for planning footsteps [KNK03, CLC05, CNK07] and reaching motions [KKK94, KKN03, BKD06, DN06, DK08].

Multi-modal planning has recently emerged for humanoids and has been developed for locomotion and climbing in difficult terrains [Bre06, HBL05, HBH06], and also to sequentially coordinate walking and pushing [HNG07]. With a focus on locomotion, extensions to the basic PRM method for handling multi-modal problems have been proposed [HBH08], and generic multi-skill planners have been developed [KHB10]. However, no previous work in motion planning has addressed the computation of motions with humanlike characteristics.

In contrast, methods originated from the computer animation area focus on achieving humanlike results from motion capture, without much importance given to searching for collision-free motions in complex environments. Probably the most popular approach for computing realistic full-body motions is to extract motions from a motion graph structure. Motion graphs are built by connecting the frames of high similarity in a database of motion capture examples [KGP02, AF02, LCR02, PB02, LWS02, Saf06]. Once the motion graph is available, a graph search is performed in order to extract motions with desired properties. The main drawback of motion graphs is that a prohibitively large structure would be needed in order to produce motions satisfying many constraints, such as around obstacles and addressing precise placements of end-effectors.

Planning methods have been integrated with motion capture data in many ways. For instance, Lau and Kuffner [LK05] plan over a behavior-based finite state machine of motions [LK06], Choi et al. [CL03] combine motion capture with probabilistic roadmaps, and many other planning methods have been proposed for synthesizing full-body motions among obstacles [KAA03, EAP06a, KL00a, LK06, PZL10]. However, none of these methods have proposed a solution for planning generic upper-body actions in a continuous space and in coordination with locomotion. The ability to search in a continuous action space allows planners to compute much more complex solutions. Our proposed method represents the first approach for solving this problem with humanlike results, and is based on a novel sampling-based search defined on a space of motion blendings.

6.3 Locomotion Planner

It can be observed that correct body positioning is essential for the execution of humanlike upper-body actions. A locomotion planner is therefore needed to explore suitable

body placements nearby the action target location. Any locomotion planner could be integrated in the overall approach in solving this problem. A motion graph based locomotion planner is chosen in this section.

The planner starts with construction of the locomotion graph in a similar fashion to Kovar et al. [KGP02], but with a more efficient segmentation procedure that relies on feature-based rules for walking cycles detections and segmentations. This procedure is similar to that employed in [MRC05]. A foot-crossing binary test is done for all the motion frames from the capture database. It looks for whether the right ankle joint is behind or in front of the plane formed by the left hip, right hip and the left ankle joints. Frames that triggers a flip in the test result are marked as transition candidates for constructing the final locomotion graph. Next step is to perform a pair-wise test between the candidate transitions, specifically only between the initial and final frames of each segmented clip in order to determine the acceptable transitions. The same distance metric for aligning the frame-frame transformation is used as in the original motion graph [KGP02]. However the overall graph construction is sped up dramatically with our segmentation method compared with the traditional pair-wise comparison among all pairs of frames. No drawbacks were detected visually for employing the simplified segmentation procedure.

Let q_i represent the initial full-body posture of the character. The task of the overall planning problem is to find a full-body motion composed of two parts: locomotion for body positioning, and then upper-body action execution satisfying a given end-effector goal location q_g. The goal location may be a position target to point to, a 4 degrees of freedom (DOFs) vector encoding position and orientation of a precise hand placement for grasping, etc. The set Q_g denotes all possible body postures satisfying the action goal point q_g.

The task of the locomotion planner is to explore suitable body placements for enabling the action planner to reach a posture in Q_g. Once the locomotion graph is available, an A* search for *unrolling the graph* is employed with the cost-to-go heuristic attracting the search towards q_g, and only allowing collision-free motions to be expanded. Figure 6.2 illustrates several expansions obtained with the unrolling of the motion graph.

As shown in Figure 6.3-(1), Whenever the locomotion search expansion (blue solid branches) generates a character posture q_a that is close enough to Q_g, q_a is then considered as a transition point to the upper body action and q_a becomes the initial posture for the upper-body action planner, which will in turn launch a bidirectional search attempting to reach a posture in Q_g. The upper-body planner is described in Chapter 6.4.

If the upper-body planner (red dashed branches) is not successful after a fixed number of iterations, the locomotion planner continues to expand towards additional candidate body placements until the action can be executed or until a maximum time limit is reached, in which case the overall planner returns failure. Figure 6.3-(2) illustrates a final successful planning result.

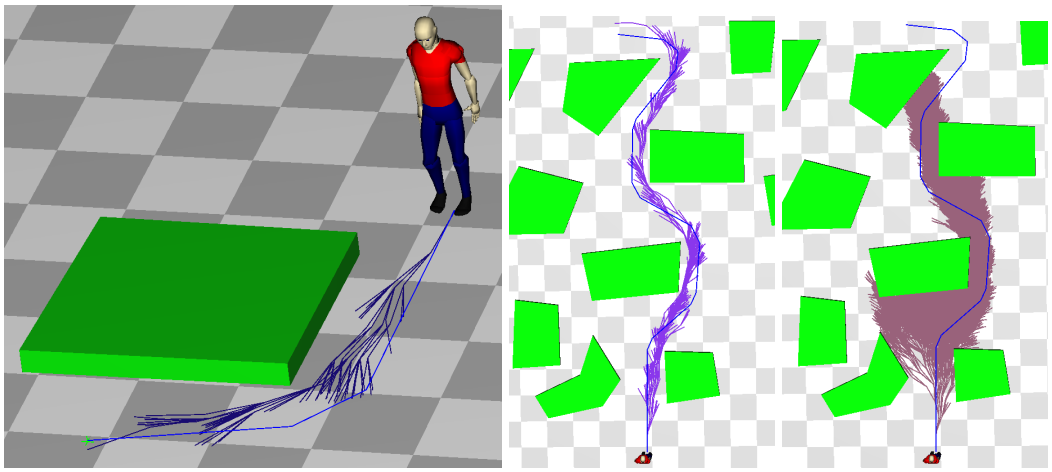


Figure 6.2: The images illustrate planner expansions obtained by unrolling the locomotion graph. Left: several possible expansions were generated near the obstacles, which will be later considered for upper-body action execution. If a long locomotion is being planned, the unrolling can be constrained to only allow expansions close to the 2D path planned on the floor plan (center), otherwise too many unnecessary expansions may occur (right).

6.4 Upper-Body Action Planner

Every time one branch of the locomotion graph expansion reaches a character pose q_a that is close enough to the action target, q_a becomes a candidate initial pose for initializing the action planner. Selecting a suitable proximity threshold is action-dependent. For example, for reaching motions a suitable distance will be related to the length of the character's arm, while for pointing actions larger values can be used for enabling pointing to targets from a certain distance.

The upper-body action is specified with a database of similar and time-aligned example motions, which are realistic upper-body action instances collected from motion capture. Let a motion $\boldsymbol{M}(t)$ be represented as a sequence of poses with a discrete time (or frame) parameterization t. With the character at pose q_a, an Inverse Blending optimization procedure (see Chapter 3) is employed in order to obtain a set of blending weights $\boldsymbol{w_g}$ creating an upper-body action motion $\boldsymbol{M}(\boldsymbol{w_g})$ precisely reaching the action target q_g: $\boldsymbol{M}(\boldsymbol{w_g}) = \sum_{j=1}^{k} w_j \boldsymbol{M_j}$, $\boldsymbol{w_g} = \{w_1, \ldots, w_k\}$, where k is a fixed parameter specifying the number of example motions from the database to be considered for blending. For a given q_g the k example actions that reach locations closest to q_g are selected. Figure 6.4 illustrates one database of example motions used in our experiments.

In our experiments $\boldsymbol{w_g}$ can be determined by inverse blending optimization under 2 milliseconds of computation. The routine returns when the error (proximity to target q_g) reaches zero or cannot be further minimized, in which case failure is returned and

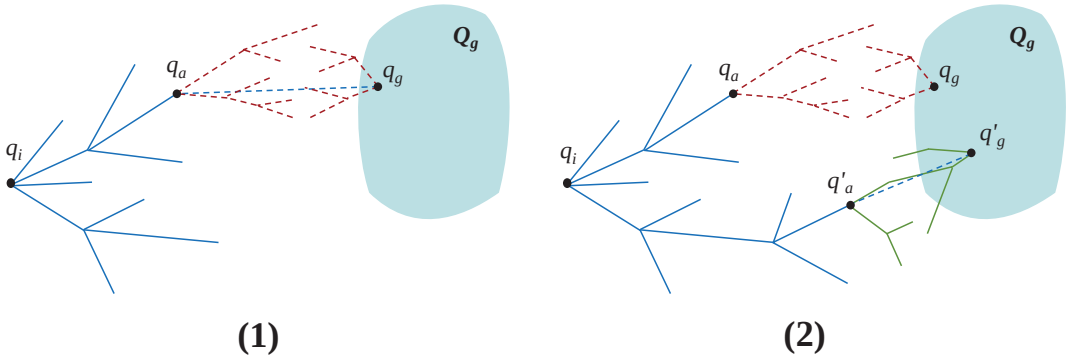


Figure 6.3: The locomotion planner unrolls motion capture locomotion clips (blue solid branches). Each candidate becomes an initial action posture q_a, from which a bidirectional upper-body action planning is performed (red dashed branches). As shown in (1) that q_a was not able to generate a solution within certain iterations of action planning to reach Q_g. In (2), a new body placement q'_a finally lead to successful planning.

q_a is discarded as a candidate body placement.

Whenever $\boldsymbol{w_g}$ is successfully computed from the inverse blending procedure, motion $\boldsymbol{M(w_g)}$ provides a realistic humanlike upper-body action precisely meeting the action goal q_g. Motion $\boldsymbol{M(w_g)}$ is then tested with several discrete collision checks over its time parameterization interval $[t_a, t_g]$. If no collisions are detected the overall planning problem is solved. Such tests are shown in Fig. 6.3 as blue dashed lines.

6.4.1 The Planner

When motion $\boldsymbol{M(w_g)}$ collides with the environment, all the non-valid frames are removed and the motion is split in two pieces: the first piece contains the adjacent valid frames starting from time t_a, and the second piece contains the adjacent valid frames leading to the frame at time t_g. These motion pieces will define two initial trees T_1 and T_2 to initialize the bidirectional action planner. A few frames equally spaced in time are taken from each motion piece to define the initial nodes of the trees. The upper-body planner is only initialized if posture q_g at time t_g is a valid posture, therefore trees T_1 and T_2 can be always initialized since posture q_a at time t_a is a valid posture determined by the locomotion planner.

With the initial search trees defined a bidirectional search procedure is repetitively called until a collision-free motion can be found to reconnect the two trees. The state space of the search is a weight-temporal space. The nodes stored in trees T_1 and T_2 will each contain a pair $(\boldsymbol{w}, t)$ where the blending weight vector $\boldsymbol{w}$ specifies the posture (by direct blending) associated with the node, and the time t indicates at which time that posture is specified along the motion being planned. In other words, each node $(\boldsymbol{w}, t)$ specifies a posture $q(\boldsymbol{w}, t)$ that is the frame of $\boldsymbol{M(w)}$ at time t. Fig 6.7 illustrates the

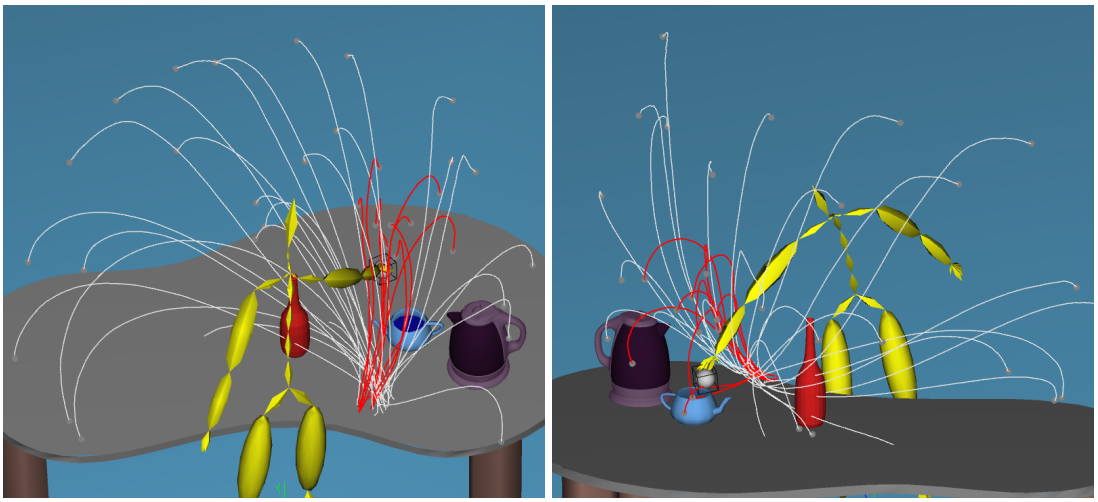


Figure 6.4: The images show the trajectories of the character's hand from a database of 36 pointing examples. The character's pose shows the final posture of the motion obtained by blending the 10 examples with the trajectories in red ($k = 10$). The blending weights were obtained by inverse blending optimization. The blended pointing motion successfully reaches the action target q_g, which is represented by the gray sphere.

bidirectional search procedure.

The strategy of searching in the weight-temporal space is the key element of the proposed planner: it enables a search procedure that explores the motion variations available in the set of example action motions. The planner will never synthesize new motions from scratch but instead will search for combinations of existing motion variations. The solution plan is a sequence of time parameterized blending weights that are afterwards interpolated to achieve a smooth solution motion with varying contributions from the example actions. See Figure 6.5,6.6 as an example.

The bidirectional search routine performs successive search expansions until trees T_1 and T_2 are connected, or until a maximum of N_{max} iterations pass, in which case failure is returned. The expansion routine of the action planner is detailed in Algorithm 1. At each expansion the algorithm tries to connect the trees, and if not possible it tries to grow each tree in the direction of the random time sample t_{rand} by a time interval t_Δ. The details of the algorithm are explained in the paragraphs below.

6.4.2 Sampling the Weight-time Space

In line 3 of the algorithm a $(\boldsymbol{w}_{rand}, t_{rand})$ pair is sampled. Time t_{rand} is randomly sampled in interval $[t_a + t^a_\Delta, t_g - t^b_\Delta]$. The $t^{a,b}_\Delta$ parameters concentrate the sampling on the parts of the environment where collisions are most often found, since the extremities of the original motion $\boldsymbol{M}(\boldsymbol{w_g})$ are always valid.

A more specific strategy for sampling the weight space is required for the algorithm

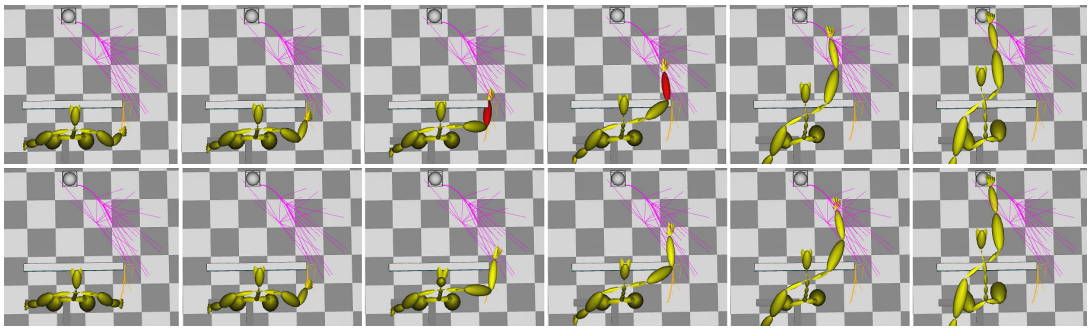


Figure 6.5: The top sequence illustrates the invalid frames in $\boldsymbol{M}(\boldsymbol{w_g})$ due to collisions (marked out in red), and the bottom sequence shows the final solution motion obtained by the action planner. The images also show the nodes and expansions on the search trees T_1 and T_2. The thicker edges represent the initial branches that were used to initialize the trees. The search was performed in the weight-time space and the position of the wrist joint was used to plot the edges of the search trees.

to better explore the motion variations embedded in the example actions being blended. We first sample one example motion index m_{rand} among the full set of example motions in the database. We also bias the index sampling to select more samples outside of the k motions used by the initial inverse blending solution in order to favor variations and avoid duplication of blended motions. Then a relatively large blending weight w_{rand} is sampled (in interval $[0.5, 1.0]$) and associated to the motion with index m_{rand}. The new weight w_{rand} is incorporated to the overall weight vector $\boldsymbol{w}_{rand}$ and the other weights are lowered by uniform scaling such that $\boldsymbol{w}_{rand}$ is re-normalized to 1 and the influence from example motion m_{rand} remains higher than the other influences. Our experiments have showed that this procedure generates more collision-free samples and yields a better overall planning performance than a simplistic uniform sampling over the overall weight space.

6.4.3 Node Expansion

After a node $(\boldsymbol{w}_{rand}, t_{rand})$ is sampled, the algorithm will then select on each tree the nodes $(\boldsymbol{w}_1, t_1)$ and $(\boldsymbol{w}_2, t_2)$ that are closest in time to t_{rand}, respecting the monotone time condition $t_1 < t_{rand} < t_2$, and then attempt to connect the two nodes (line 6 in the algorithm). If a connection is found the algorithm successfully terminates, otherwise the two trees are expanded.

Routine try_to_expand() in lines 9 and 10 of the algorithm will try to grow each tree in the direction of $(\boldsymbol{w}_{rand}, t_{rand})$ by a time step t_Δ. This procedure yields a new node $(\boldsymbol{w}'_1, t'_1)$ to be connected to $(\boldsymbol{w}_1, t_1)$ on T_1. Weight vector $\boldsymbol{w}'_1$ is an interpolation between $\boldsymbol{w}_1$ and $\boldsymbol{w}_{rand}$ at time t'_1, where $t'_1 = t_1 + t_\Delta$, adjusted such that $t'_1 \leq t_{rand}$. A discrete collision test is performed for $q(\boldsymbol{w}'_1, t'_1)$ and it is added to T_1 only if no collisions are detected. An equivalent expansion procedure is performed to add node

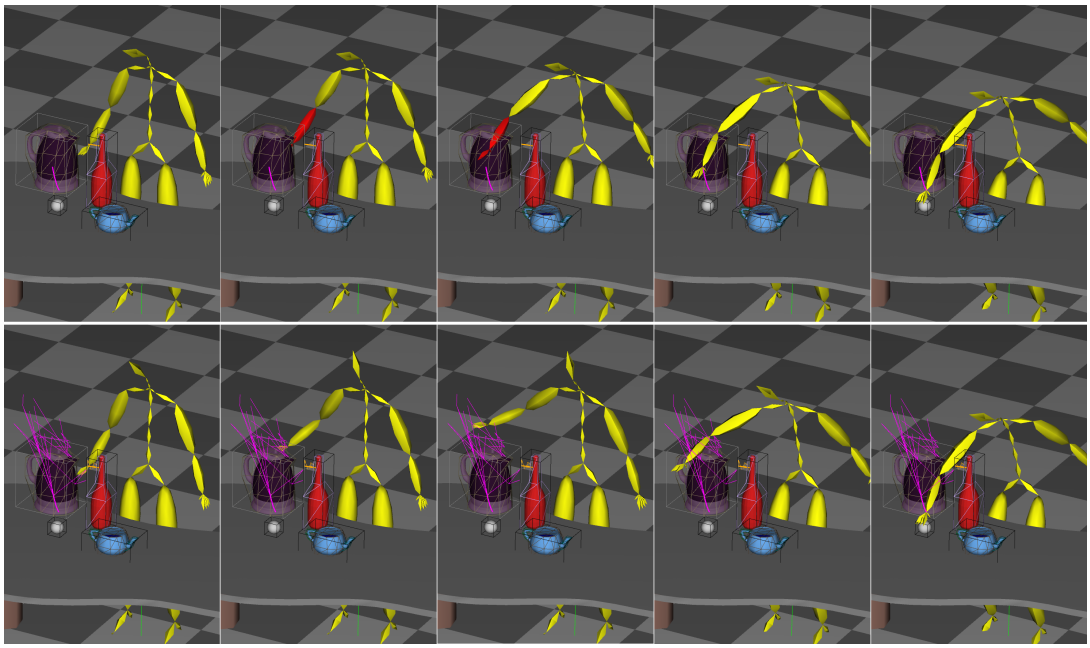


Figure 6.6: The two sequences illustrate the same scenario shown in Fig 6.5 but from a perspective viewing angle. The top one shows the original wrist trajectory with collision part discarded, and the bottom shows the final solution obtained by the action planner, with the search tree expansions plotted.

$(\boldsymbol{w}'_2, t'_2)$ to tree T_2.

6.4.4 Lazy Collision Tests

The algorithm postpones fine resolution of collision tests along tree edges until the trees are connected, in order to promote fast exploration of the search space. When the trees connect, discrete collision tests are performed over the segments composing the solution path that have not yet been tested for collision, and a solution is found if no collisions are detected. If collisions are found, the connection between the trees is broken at the invalid edge and the expansion iterations re-start in search of new connections. When a solution is found, the solution motion is generated by smoothly interpolating (with ease-in and ease-out interpolators) each pair of motions generated by adjacent weight vectors in the solution plan. Such interpolated concatenation of motion pieces is illustrated in Figure 6.8.

6.4.5 Extensions

Improvements have been obtained with the popular strategy of sampling nearby obstacles. We identify the nodes that, after a few iterations, could not connect to samples due collisions. These nodes indicate proximity to obstacles. The algorithm can then

Algorithm 1 Action Planner Expansion Routine

Expand_Bidirectional_Search (q_a, $\boldsymbol{w}_g$)

1. $T_1 \leftarrow$ tree starting at q_a
2. $T_2 \leftarrow$ tree reaching q_g
3. $(\boldsymbol{w}_{rand}, t_{rand}) \leftarrow$ sample in weight-time space
4. $(\boldsymbol{w}_1, t_1) \leftarrow T_1$ node closest to t_{rand}, $t_1 < t_{rand}$
5. $(\boldsymbol{w}_2, t_2) \leftarrow T_2$ node closest to t_{rand}, $t_{rand} < t_2$
6. **if** (connection $(\boldsymbol{w}_1, t_1)$ to $(\boldsymbol{w}_2, t_2)$ is valid) **then**
7. **return** CONNECTION_FOUND
8. **else**
9. try_to_expand ($(\boldsymbol{w}_1, t_1)$, $(\boldsymbol{w}_{rand}, t_{rand})$, t_Δ)
10. try_to_expand ($(\boldsymbol{w}_2, t_2)$, $(\boldsymbol{w}_{rand}, t_{rand})$, t_Δ)
11. **return** NOT_YET_FOUND
12. **end if**

prioritize the sampling (temporally) near these nodes, within a window of $\pm t_\Delta$, so that the success rate of finding collision-free nodes around these locations is improved.

Several other extensions have been integrated in our framework, in particular, the gaze of the character can be independently controlled to fixate the action target and the obstacles. The velocity profile of the end-effector in each solution motion obtained is also smoothed as a final post-processing operation in order to maintain the final motion with a velocity profile close to the velocity profile of the most similar motion (in duration) in the database.

6.5 Results and Discussion

Several animations have been produced for demonstrating the results of our overall method. See Figure 6.1, 6.9 and 6.11 for examples. The accompanying video also presents several obtained results. The locomotion planner quickly expands several body placements nearby the action targets, providing the upper-body planner several options for planning the upper-body action. The overall full-body planner therefore evaluates different body placements until a suitable, valid and collision-free action is found.

The decoupling between locomotion and upper-body actions avoids several problems related to large motion graphs including both locomotion and upper-body actions. The weight-time space of action blendings provides a continuous space for planning where only the humanlike strategies encoded in the example actions are explored. The action planner uses a relatively large t_Δ in order to quickly explore the space around the obstacles, such that the action planner can quickly return to the locomotion planner in case of failure, and it also minimizes jerky variations in found solutions. In practice the planner showed to always produce smooth motions with the only post-processing operation being the tuning of the final velocity profile for the end-effector. No other

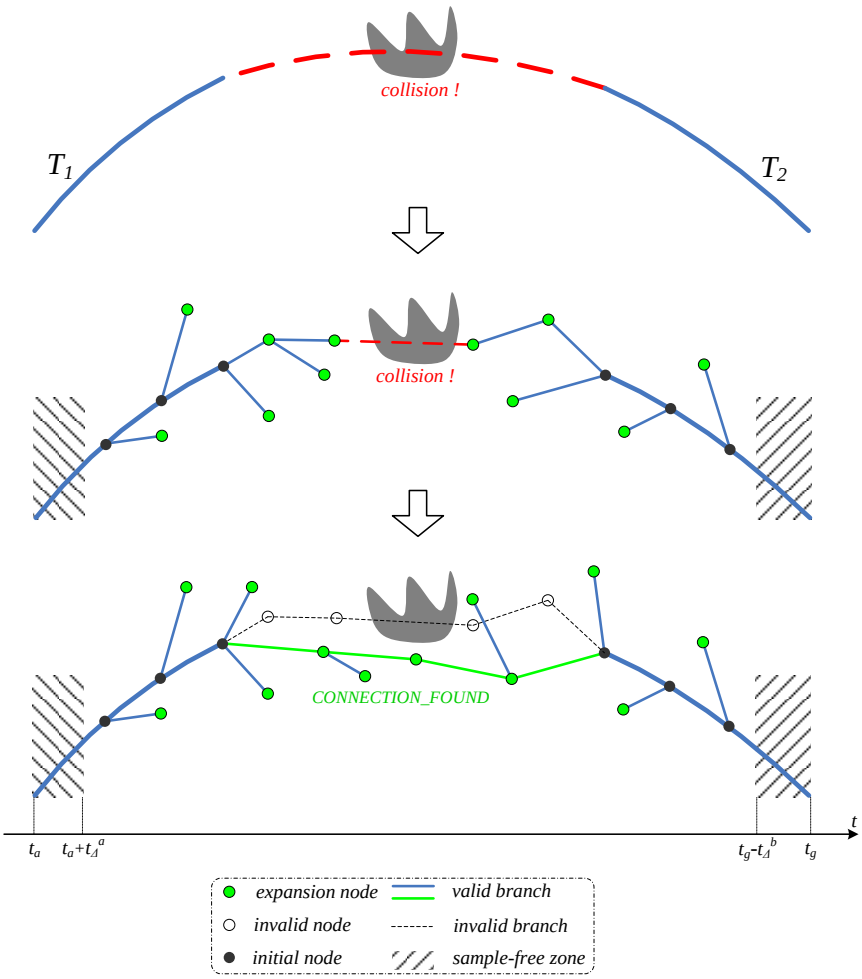


Figure 6.7: Illustration of the bidirectional expansion procedure.

post-processing smoothing procedures were required. The solution motion always resembles the given examples in the action database and "no surprises" in the final motion will appear.

The performance of the planner largely depends on the considered example motions and on the number of triangles considered for collision detection. In the example in Figure 6.9 a database of 36 motions (shown in Figure 6.4) was used and 5K triangles were considered for collision detection. A 10000-trial benchmarking test of the action planner with random obstacle and target locations found solutions in 78% of the cases, with each bidirectional search taking in average 4.92 milliseconds. For the remaining cases the planner returned failure after hitting the limit of 500 expansions within an average of 12.76 milliseconds of computation. However, we found our planner does take much longer on the collision checking after the bidirectional search is done, a bottleneck of the overall planning procedure. These times demonstrate that an extensive overall search composed of both locomotion and action planning can be executed in less than one second in reasonably complex environments.

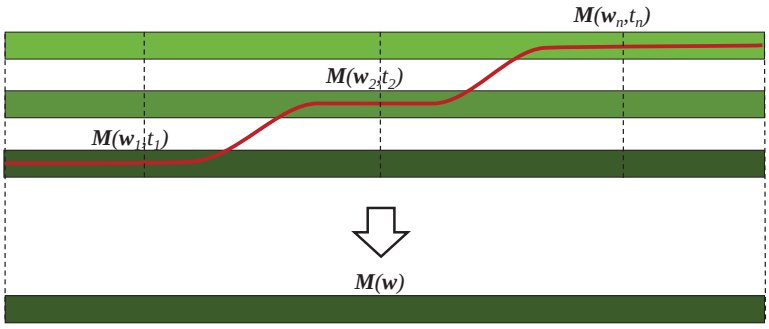


Figure 6.8: Nodes $(\boldsymbol{w_i}, t_i)$ along a tree branch are concatenated with smooth transitions to form the corresponding motion $\boldsymbol{M(w)}$.

6.6 Conclusions

This chapter describes a multi-skill motion planning approach that integrates discrete search in a locomotion graph with a systematic bidirectional expansion in a novel weight-time action search space. The search spaces are based on motion capture data, ensuring the achievement of humanlike results. The approach for the upper-body action planning represents the first sampling-based search algorithm defined on a continuous space based on humanlike motion capture examples. The overall approach is able to automatically coordinate locomotion with generic actions among obstacles, and the produced motions are realistic, collision-free, and precisely meeting given targets in the environment. Since actions are modeled from motion capture examples, any upper body actions can be planned, from pointing and reaching motions to generic gestures.

One drawback of the planner is that it just finds the first solution that is feasible, without taking into account aspects that may be imposed by other higher-lever goals. Improvements could be made by introducing behavior modeling, to determine for example if the body placement is suitable in the given environment, or in the case of planning demonstrative actions towards certain observer, whether the motion is visible to the observer. The next chapter attempts to address these issues.

Figure 6.9: The top sequence shows the initial motion computed by inverse blending. The initial motion collides with the environment and it is therefore used to initialize the action planner (Algorithm 1). The bottom sequence illustrates the collision-free solution obtained with the planner. Since the planner is restricted to search in a blending space of example actions, the particular strategy obtained in the solution also exists in the database of example actions. The solution of "pointing over obstacles" can therefore be considered to be a humanlike solution.

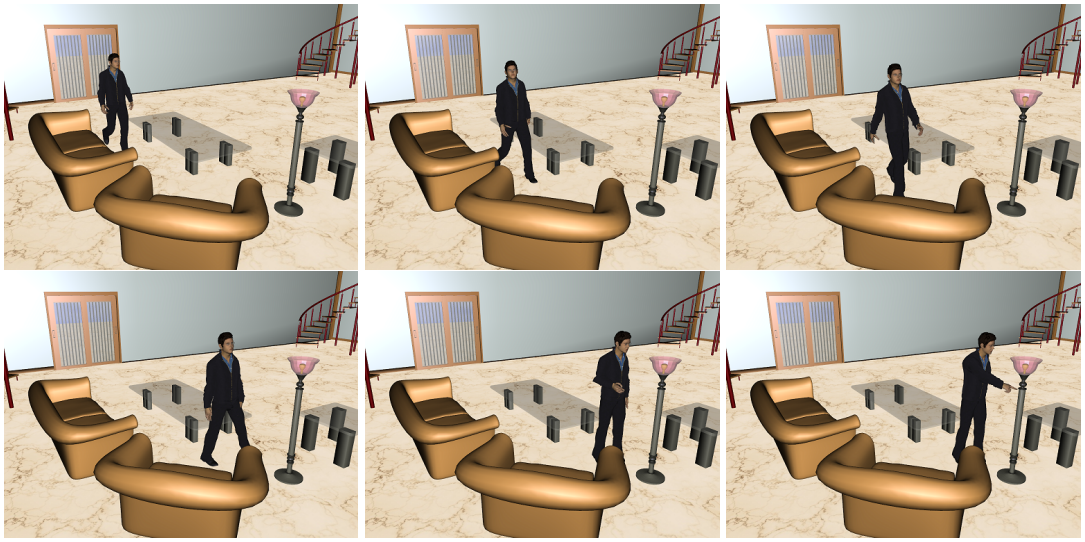


Figure 6.10: Another overall planning result that combines walking straight and sideways, and then turn on the light switch.

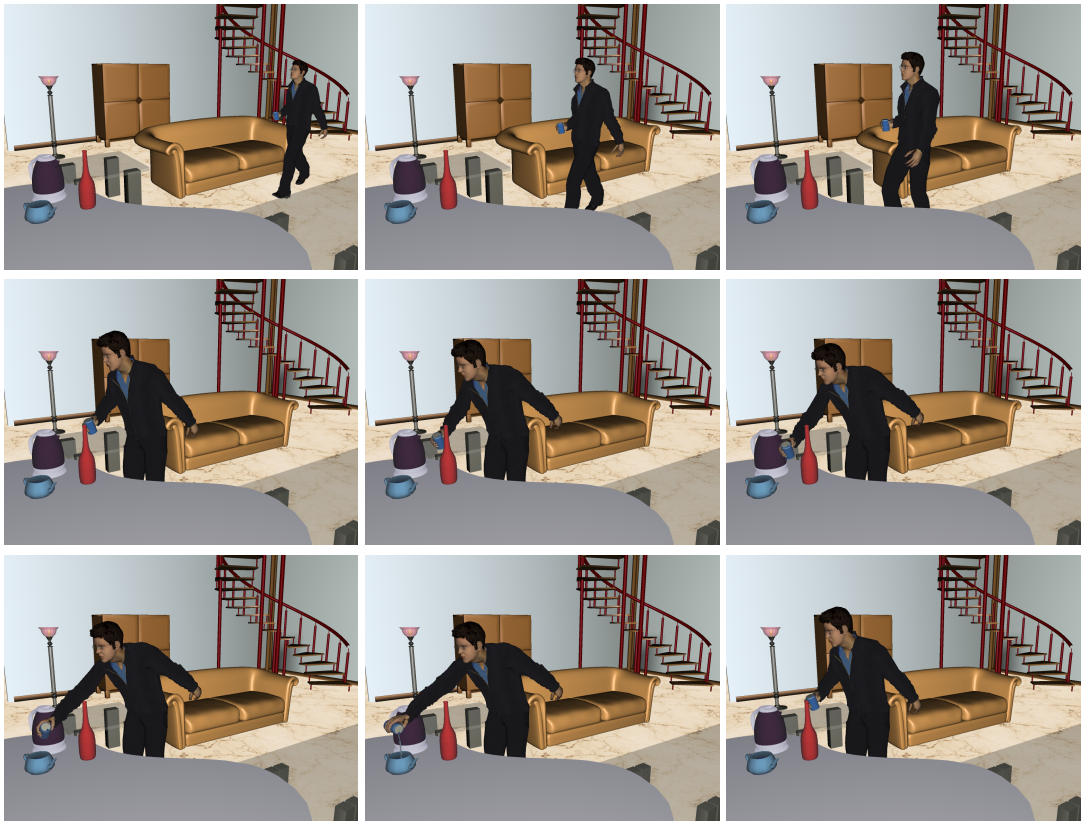


Figure 6.11: Example of pouring water toward a distant location on a table, among obstacles, and coordinated with locomotion. The behavior of the left arm goes backwards in order to assist with balance, which completely comes from the example motions being blended by the action planner.

CHAPTER 7

Whole-body Motion Planning with Body Coordination

In order to deliver information effectively, virtual human demonstrators must be able to well replicate interaction patterns observed during human-human interactions in real-world settings. Multiple levels of coordinations need to be addressed, from suitable body positioning to performing upper-body actions synchronized with meaningful gaze behaviors, and so on. This chapter introduces a whole-body motion planning system capable of synthesizing demonstrative tasks exhibiting complex coordination patterns in humanlike ways. Given a target object to be demonstrated to an observer, our planner synthesizes a complete full-body motion solution with coordinated locomotion, action execution and gaze behaviors. Human-likeness is achieved by respecting coordination rules extracted from human-human interactions through experiments with human subjects. Realistic motion results are generated using data-based synthesis techniques. A novel collision avoidance procedure in the blending space is employed for the action synthesis, and all planning stages produce realistic collision-free motions among obstacles. As a result, the planner is proposed with a complete framework for effective modeling and solving complex demonstrative tasks involving whole-body motion synthesis in realistic environments.

7.1 Introduction of Motion Planning and Coordination

Virtual humans and embodied conversational agents are promising in the realm of human-computer interaction applications, with a central goal of developing virtual assistants that can effectively interact, learn, train, and assist people in a wide variety of tasks. Consider the case where a virtual agent performs a demonstrative task specifically towards a near-by observer inside an everyday-like environment: the agent walks from a starting location, avoiding furniture and other obstacles along the way, stops at an appropriate location close to the target object, with a clear view of the observer, interacts with the object (e.g. demonstrates how to operate the device), while visually engaging with the observer, making sure effective delivery of this demonstrative information with gesticulatation. It may seem trivial and effortless for any of us to carry out such task, but it is essentially an orchestration of movements with high level of complexity. It consists of multiple levels of coordinations from locomotion to body positioning and then to the combination of gesture with eye contacts, and all must appear smooth and seamless.

This chapter attempts to dive into these harmonious, multi-level combinations of

actions and behaviors, analyze and model the underlying coordinations, and build a planner capable of generating natural-looking whole-body motion sequences for virtual agents in similar settings.

In order to model the overall problem of synthesizing humanlike demonstrations, several experiments were conducted with human subjects. The experiments consist of object demonstration tasks. Human subjects were asked to freely approach the target object and position themselves with respect to an observer that stands at various locations. These experiments provide ground truth for modeling the overall planning problem, specifically the coordination rules and models which are extracted from these motion data.

The result is a whole-body motion planning system capable of synthesizing humanlike motions for virtual agents executing demonstrative actions and tasks effectively. The proposed planner consists of five main aspects: 1) planning for optimal body positioning with respect to the target object and observer at arbitrary locations; 2) collision-free locomotion planner from starting location to the body placement planned; 3) collision-free, humanlike upper-body action planning; 4) coordination module that smoothly concatenates action motions with locomotion sequences; 5) engagement planner that models the gaze and gaze-related behaviors which governs the eye contacts of the virtual agent with the observer as well as the target object. Among those, 1) 4) and 5) are based on data captured from human subjects to ensure the realism of the final synthesis. These five aspects are integrated together, each represents an essential part of the overall task, in an attempt to mimic how humans solve similar tasks in resembling setups.

The main contribution of the proposed planner is the definition, modeling and effective solution of complex demonstrative tasks involving mobile action execution respecting humanlike patterns. The overall problem has not been addressed before and the proposed solutions take into account all relevant aspects in an unified way.

In addition, several new techniques are proposed, including a locomotion planner that is capable of generating walk sequences along a pre-defined path with precise starting/arriving positions and facing direction, which is key to achieve the overall planning problem. This chapter also proposes a novel upper-body action planner capable of generating collision-free motions with precise spatial constraint enforcement at stroke point(s), which is also an important aspect within the overall problem. Our system is able to produce coordinated whole-body synthesis under a fraction of a second with reasonably-sized motion capture databases.

The rest of this chapter is organized as follows: after literature review, our experiment settings are presented in Chapter 7.3), then the planner sub-modules are described in Section 7.5 through 7.9. Section 7.10 shows the results and summaries the conclusions.

7.2 Related Work

The immediate goal of our work is to generate humanlike full-body motions that are effective for display and demonstration of physical target objects and devices to human observers by means of a virtual agent. In our approach, the agent needs to position itself at the best suitable location for the demonstrative task at hand, taken into account the spatial locations of the target object, human observer and surrounding obstacles, which is in particular important to guarantee that the demonstrative action delivered is visible to the observer. Also as important is that the agent should perform these gestures and actions with clarity and precision in order to appropriately reference the target objects without ambiguity [HHK11b, HNS04]. Human observers are highly attuned to subtle differences in motor behavior, including minor shifts in velocity and pointing velocity related to the size of an object, as in Fitts' Law [SM04].

In this chapter we attempt to analyze and model the whole-body motion for such virtual demonstrator identifying and delivering information about target objects to a human observer at varied locations. In an attempt to mimic how humans solve tasks of similar kind, we intuitively propose to break this planning problem into five parts: body-positioning, action/gesture modeling, locomotion planning, coordination between locomotion and action, and visual engagement between the agent and observer. Below is a brief review for some of these aspects.

Throughout the literature on motion synthesis for autonomous virtual agents and motion planning for humanoid robots, there has been many works that focus on upper-body gesture and action modeling, including stroke-phase blending [TMM08], action synthesis with varied spatial constraint(s) [SZG05, HK10, MK05], and motion style control [RBC98, GMH04]. There have also been many works on goal-directed walk gait and locomotion synthesis, with or without explicit path planning and path following, such as [HG07, KGP02, TLP07].

The planning problem becomes more interesting when these two aspects are jointed together, such as [EAP06b, SKF07] that combine arm planning (reaching or grasping) on top of locomotion, however in many cases the arm movement were merely superposed onto the walk sequence without explicit model for the coordination. Although algorithms from the robotics domain [HL09, KHB10, HN10] tend to plan individual "primitives" such as arm motion with RRT- or PRM-like probabilistic sampling extending the idea of multi-modal planning framework, others in computer animation share similar framework but prefer data-driven synthesis with motion capture data [FXS12, PZL10] or a hybrid method [HMK11].

When it comes to jointing upper- and lower-body motions together for whole-body motion synthesis, there are in general two categories of the existing researches: space-time constraint enforcement (e.g. with Gaussian process estimators [IAF09]), and smoothly jointing decoupled motions together [HKG06]. Finally with the user-study on blending window length and just-noticeable-difference in blending artifacts [WB04] which gave us insights on how to achieve better ease-in and ease-out blending transi-

tions.

All these works are a source of inspiration for our planner. The motivation is that all existing works in general failed to address the planning problem with a third-party person as the audience observing the planned motion sequence. This is crucial in demonstrative tasks with inter-person interactions, because not only this visual contact in the form of gaze behaviors add to the human-likeness of the overall synthesis, but it has also been proven to greatly improve the amount of information the audience memorizes and recalls, boosting the amount of information delivered as shown in the study of robot storytellers [MHF06] and narrative virtual agent in a CAVE system [BWA10]. Although many work in computer graphics and cognitive science have focused on modeling these gaze behaviors (e.g. eye-ball movement, etc) with great details, few has given attention to modeling body positioning and body placement in association with actions/gestures and locomotions, which is a very much neglected aspect in modeling such behaviors.

[SA76] is one of the pioneering works to introduce the concept of "Territoriality" for human-human interactions with many illustrations, however it lacks detailed computational models. More recently [POO09] proposed the modeling of interactions between virtual walkers based on experimental data, but only applies to scenarios involve multi-agent locomotion collision avoidance. Our main contribution is a much more detailed coordination model that handles the simulation of body positioning, locomotion, gaze and upper-body action. Here, a whole-body planning system is presented that affects many aspects of the overall planning problem: the virtual demonstrator must know how to position itself, and also how to visually engages with the observer, for better delivery of information. Details are provided in Chapter 7.3 through 7.9. In the current line of work, we focus on the problem of modeling motions for interactive training applications that demand complex gestures and actions that can be reproduced in a human-like way. In particular, we are interested in modeling natural and realistic gaze behaviors and natural body positions in the context of performing demonstrative tasks. Special interests have been put in factors related to these behaviors and also the virtual environment, including target objects that are being referenced and the observers to whom the demonstrative tasks are addressed.

7.3 Modeling Demonstrative Motions from Human Subjects

We started to model the overall problem of performing humanlike demonstrations with the help of experiments performed with human subjects. Four human participants were recruited to perform a variety of basic pointing tasks with full-body motion capture without eye tracking. Six small target objects T were placed on a horizontal coarse mesh grid inside the 8x12 $sqft$ capture area. Participants were asked to perform pointing actions towards each T specifically for a human observer O standing at various positions O_{P1} though O_{P5} (creating different perspectives), see Fig 7.1 and 7.3. Small-sized targets were in particular chosen to reduce the possible side-effects that the size of T might have on the behaviors captured.

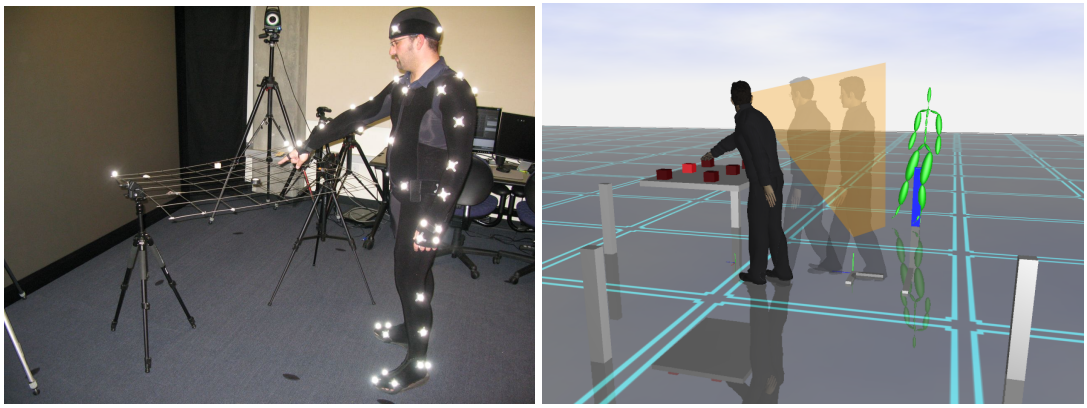


Figure 7.1: Left: experiment setup. Right: illustration of one reconstructed motion. The observer location is represented with the green character and the maximum head orientation performed in the direction of the observer is shown with the orange plane.

The captured pointing motions was segmented into trials, each containing one full pointing action with locomotion and associated behaviors: the participant (1) stands about 4 feet away from the mesh grid, (2) walks towards the grid, (3) points to one of the targets T, (4) either names or briefly describes T, (5) direct the attention of O by gazing back and forth at the O and T, then finally (6) steps back to the starting position and prepares for the next trial. Each capture session includes 30 trials: O maintains the observing position O_{Pi} until all 6 Ts had been addressed, then moves to the next O_{Pi+1}. The sequence of target selections was random.

Fine-grained manual annotations were conducted for each trial and marked out the occurrences and timing of all key events for gaze and related behaviors. The key events can largely be summarized into 3 categories: locomotion (step towards table and back), gaze (at floor, target and observer) and action phases (start/stroke/end). Please see [HKM11] and Chapter 7.9 for details on the experiment, annotation and gaze modeling.

7.4 PLACE Planner Overview

Given the observed experiments and annotations, it became clear to brake down the multi-modal planning problem into 5 main aspects summarized as **PLACE**: body-**P**ositioning, **L**ocomotion, **A**ction, **C**oordination and **E**ngagement planner. Specifically speaking, it relies on the following steps to solve the overall planning problem:

1) solve the optimal body positioning: given a target object T to be demonstrated by virtual agent A with action towards an observer O at an arbitrary position O_{px}, the planner determines the best body placement P for the agent A. For the gesticulation to be effective and ambiguity-free, A should stand within reachable distance from T [HHK11a], such that the demonstrative action (e.g. pinpointing, grasping, etc) can be executed. It must ensure successful reachability of P with a locomotion sequence, as

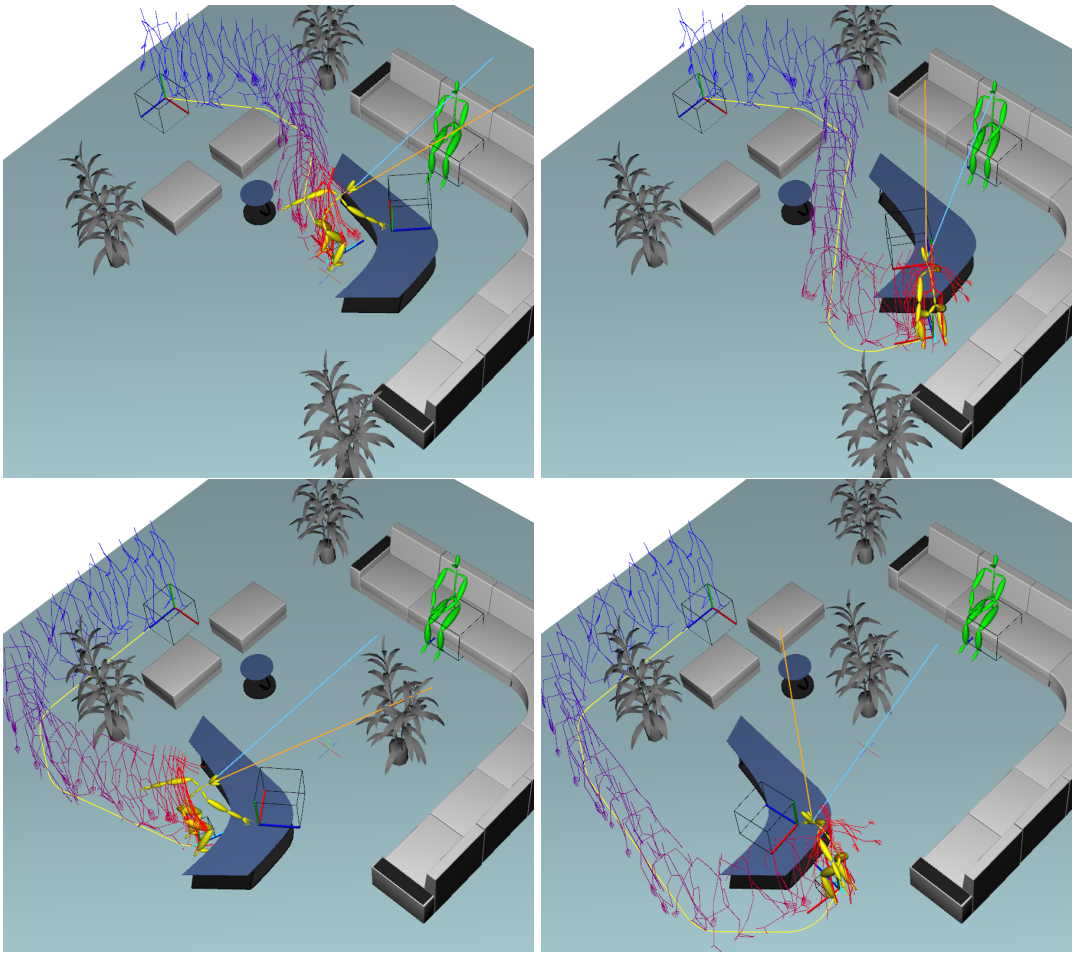


Figure 7.2: In the top two scenarios the computed demonstrations reasonably face the observer, while in the bottom two cases a visual occluder (the house plant) leads to solutions with non-trivial placements. The orange and blue lines respectively represent the head and the eye gaze orientations, at the demonstration action stroke point. The resulting gaze always reaches eye contact with the observer.

well as successful action execution towards T, and also respecting visibility constraints between A and O.

2) solve the locomotion sequence: after P is determined in step 1, a collision-free locomotion sequence is computed so that A is able to walk from its current location to the suitable body placement configuration P with specific stop location and facing direction.

3) solve the upper-body action delivering the demonstrative information at the end of the locomotion synthesis: the action must appear human-like, must be able to enforce precise spatial constraints for end-effector placement, and must be capable of accommodating fast obstacle avoidance without losing the realism.

4) solve the coordination in-between locomotion and action: smoothly transitions

from the end of locomotion into the beginning of action by adapting patterns extracted from human participants.

5) solve for the visual engagement aspects: controls the gaze and related behaviors in coordination with rest of the body movements by maintaining natural agent-observer eye contacts during the demonstration.

The proposed PLACE planner targets specifically at the following scenario: given a virtual environment of the workspace with obstacles and the location of a target (goal of the action, 3D coordinates only), a virtual agent demonstrates the target object towards an observer inside the same environment, for example by pinpointing a location on a map, or shows how certain device works. Figure 7.2, 7.16 and 7.17 present several results generated under different environment settings.

7.5 Optimal Body-Positioning Module

Given a virtual environment of the workspace with obstacles and the 3D location of a specified target object T (i.e. goal of the action), a virtual agent A demonstrates T specifically towards an observer O close by. A model is defined to determine the optimal body positions for A carrying out the gesticulation specifically towards O in a virtual setting, which includes the following 3 aspects shown in Fig 7.3-right: (1) standing location at the end of locomotion that includes a comfortable distance between A and T for action executing, and relative direction β for A to stand by T. (2) A's body facing direction θ. (3) maximum head rotation for the gaze behavior during the demonstration towards O.

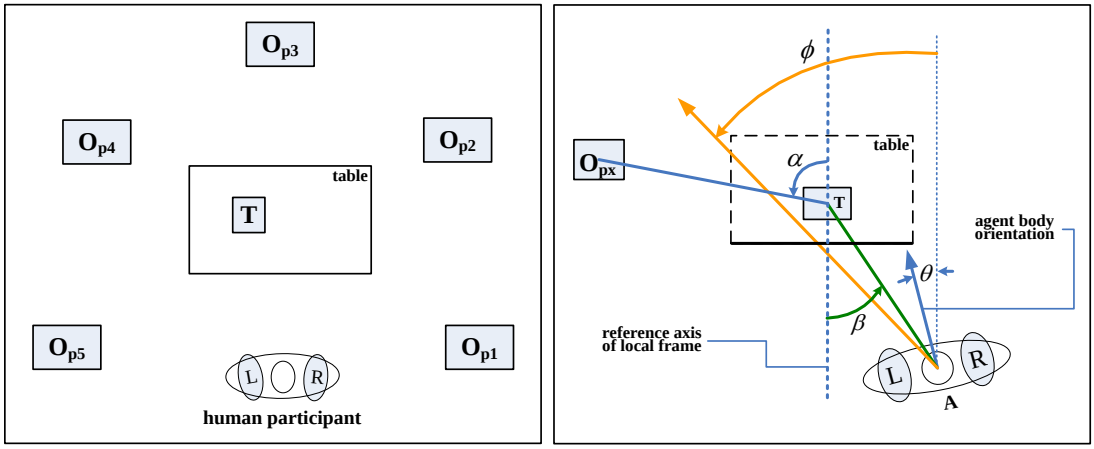


Figure 7.3: An illustration of the body positioning model. Left: top-down view of the experiment setup. Right side shows the parameter definition: α is the relative location of O; β is the relative location of the virtual agent A being modeled; θ: body orientation when agent stops to execute the action towards O; ϕ denotes the maximum head rotation for the gaze-shift plane when A visually engages with O.

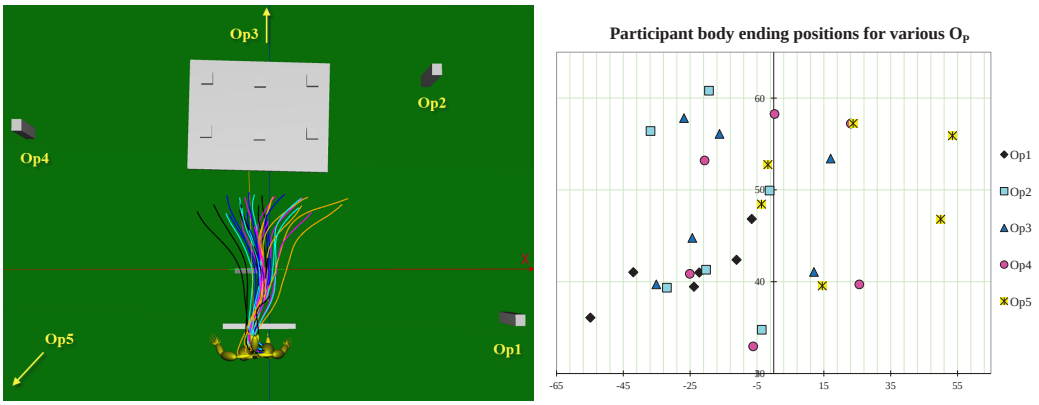


Figure 7.4: A series of raw trajectories (left) and ending positions of participant's root joint (right) from one particular trial. Both plots use the world coordinate frame.

To derive this model, full-body motion capture sessions were conducted with human participants recruited to perform a variety of basic pointing tasks (without eye tracking). 6 small target objects were placed on a horizontal coarse mesh grid simulating a table. Each participant's action was observed by a human observer O standing at different locations ($O_{p1} \sim O_{p5}$), see Fig 7.3-left. For each capture trial, the participant (1) stands 4 feet away from the mesh grid, (2) walks towards the table grid, (3) points to one of the Ts specified by O, (4) verbally and visually engages with O by saying the name and function of T, (5) direct the attention of O as needed by naturally gaze back and forth at O and T while pointing and talking, and lastly (6) steps back to the starting position and prepares for the next trial. Each capture session includes 30 trials where O maintains its observing position O_{pi} until all 6 targets have been addressed, then moves on to the next location O_{pi+1}. This is repeated until all targets are named and described 5 times. The whole capture was annotated manually with our annotation tool to mark out and extract the occurrence and timing of a series key events within each trial.

Local coordinate frame of target T was chosen to measure the angular parameters α, β, θ and ϕ. The vertical line in Fig 7.3-right is the reference axis of T's local frame. The reason to use local frame is that these angular parameters are of great correlations with the location of T. Fig 7.4 plots a series of raw trajectories (on left) and end positions (on right) of A's root joint within one particular capture trial using world coordinate frame. 5 distinct colors represent different observer's locations $O_{p1} \sim O_{p5}$, each color corresponds to the plotting of 6 trials towards the same T while O holds its location. Note that major overlaps can be seen from the plots. However when using local coordinate frame of T, plots of the same session form nicely into clusters, as shown in Fig 7.5. An intuitive explanation is that since the actions were performed specifically towards T, making T as the center of the action, similar to the "axis" concept [SA76] to describe interactional connections. Hence we choose to use the local frame of T as the reference frame for measuring the angular parameters.

The next step is simply estimating β, θ and ϕ values for any given α with non-

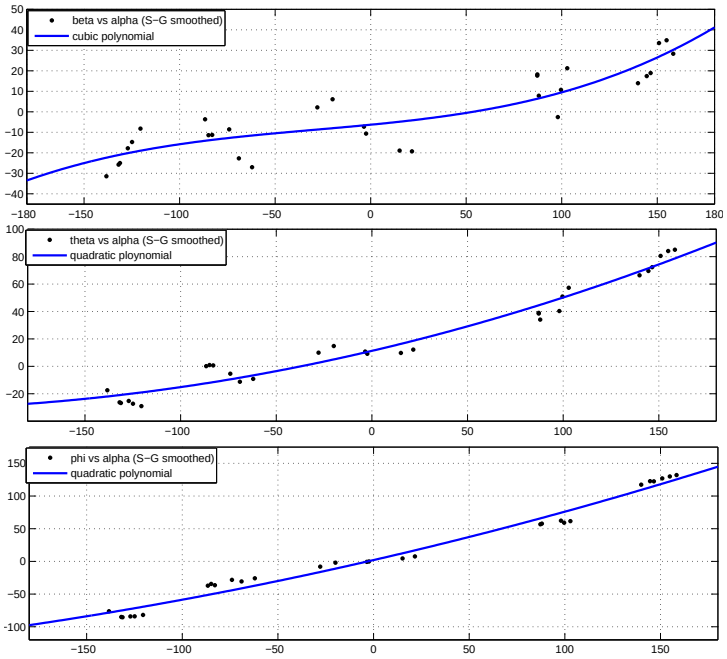


Figure 7.5: Model the body positioning with non-linear regression on filtered data points. X axis is α value; Y axis from top to bottom represents β, θ, ϕ respectively.

linear regressions. After smoothing the raw measurements with Savitzky-Golay filter, quadratic and cubic polynomial curves were fitted for β, θ and ϕ, shown in Fig 7.5. Details of the fitting parameters and goodness of fit are shown in Appendix 11.1.

The **PLACE** planning procedure are described as the follows steps. First within the body positioning module:

1. Queries the upper-body action planner (Chapter 7.7) for *comfort zone*, which determines the best range (in terms of euclidean distance) for A to stand away from T. When A is inside the zone, action planner is more likely to find solutions (i.e. A's end-effector precisely reaching T). For unreachable T, a distant pointing motion would be generated. The computation of *comfort zone* relies on the location (or height) of T, style of action synthesized (choosing a particular style within the action example database), and also the size of the table when T is not reachable.

2. Determines an initial orientation for the reference frame (see Fig 7.1-right), either perpendicular to table's edge, or directly facing the observer (T is in-between A and O, making O at the center of A's field of view). Second case is favored as it generally yields better visibility among A, O and T.

3. The planner searches for a better orientation of the reference frame by discretely sampling around T (e.g. in 5° increment). For each sample, the parameters β, θ, ϕ are computed and A is placed accordingly. Samples that cause A to collide

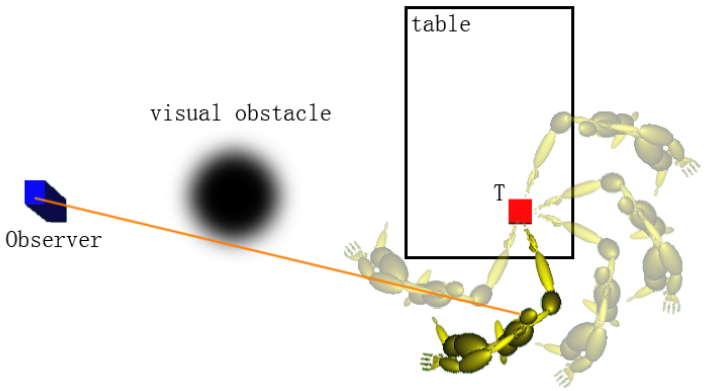


Figure 7.6: Sampling of the body positioning around T in search for the optimal candidate, i.e. the collision-free candidate that is solvable by the locomotion planner and with the best "visual clearance".

with obstacles in the environment are rejected, leaving the valid ones to later query the locomotion planner, and finally pushes those with feasible collision-free locomotion solutions into a queue of the body-positioning candidates.

4. All body-positioning candidates are ranked in descending order of "visual clearance" to penalize those with visibility blocked or partially blocked. Visibility is computed simply by measuring how much A's gaze plane intersects with obstacles in between A and O. Candidates with the best visibilities are picked to proceed, and those with zero visibility are rejected, shown in Fig 7.6. Note that visibility is measured with a continuous number in $[0, 1]$, i.e. seeing through a house plant may yield 0.5 (partial blockage). Here only agent-observer visibility constraint is enforced, and the planner assumes agent-target and observer-target visibilities always exist. The planner favors solutions with better visibility, meaning that A might need to turn its head $180°$ to gaze at O standing behind. This is because uncomfortable body positioning/gaze-shift is still better than bad visibility, since A must be able to have visual contact with O.

Following the successful planning of body positioning, the locomotion module (Chapter 7.6) synthesizes the walk sequence, then the action planner (Chapter 7.7) generates the needed action motion, and finally the coordination module (Chapter 7.8) joints the two sequences with smooth transition plus gaze related behaviors (Chapter 7.8).

7.6 Locomotion Synthesis

A path following-based locomotion planner is integrated in our system. The goal of the proposed planner is three-fold:
(A) it quickly checks for locomotion solvability when queried by body poisoning mod-

ule of Section 7.5 and reject samples with no feasible paths. Each check must be fast, ideally on a scale of a few milliseconds.
(B) it navigates the agent through narrow passages with precise starting and ending body configurations.
(C) the overall locomotion must resemble the captured trial motions from human participants, i.e. sharp body turnings occur at the starting and the end of the locomotion sequences with very small turning radius.

For (A) the query is based on an algorithm for computing shortest paths from triangulated mesh that are constructed from a polygonal representation of the virtual environment [Kal10] with built-in local clearance info. It allows fast query of high quality collision-free polygonal path with arbitrary clearance distance away from obstacles for the agent to pass through. The path query takes under a few milliseconds of computation time for scenarios similar to Figure 7.2.

As discussed in Chapter 2, due to the nature of discrete search, it remains challenging for motion graph-based locomotion planners (such as what's proposed in Chapter 6) to enforce higher DOF constraints, i.e. to have the character arrive precisely at any specified 2D goal location facing precisely at any specified body orientation, a total of 3-DOFs constraints as described in (B). with the additional requirement (C) to specifically model the locomotion patterns observed from the trials, we propose an unique locomotion planner solution tailored to solve this particular problem. However, the trade-off is that with the focus in meeting the three requirements, the quality of our final synthesis may not be as good as the motion graph-based proposed in the previous chapter.

With human-like quality synthesis in mind, three specific types of locomotion sequences have been collected: 1) starting steps from rest posture to walk towards various directions 2) arriving steps from the ending walk cycles to stopping at rest posture facing various directions, and 3) a normal straight walk cycle. 2) is especially tricky to model as it usually contains sharp turnings in very tight spaces, which is a main reason for us to choose data-driven synthesis, see left most image in Figure 7.7. The walk cycles in these captures are segmented manually.

The locomotion sequence is synthesized with the following steps: After the path is queried, the starting and arriving clips that best fit the path are picked and placed at the exact starting and ending body positioning. Next the cyclic walk clip is repeatedly applied onto A and concatenated following the starting clip along the path. Finally to smoothly transition from the cyclic walk into the arriving clip, inverse blending is used here to generate one single transition step by best matching the feet placement and body orientation, see Chapter 3.5 for details. After a seamless transition is generated, the last arriving clip is appended. Figure 7.7 shows the final result of the locomotion result, which typically takes roughly 50 milliseconds to generate the whole sequence. Any existing locomotion planner that meets the requirements of both (A), (B) and (C) could be integrated to replace our solution, however such planner is less seen in the literature. Additional results are provided in Figure 7.15, 7.16 and 7.17.

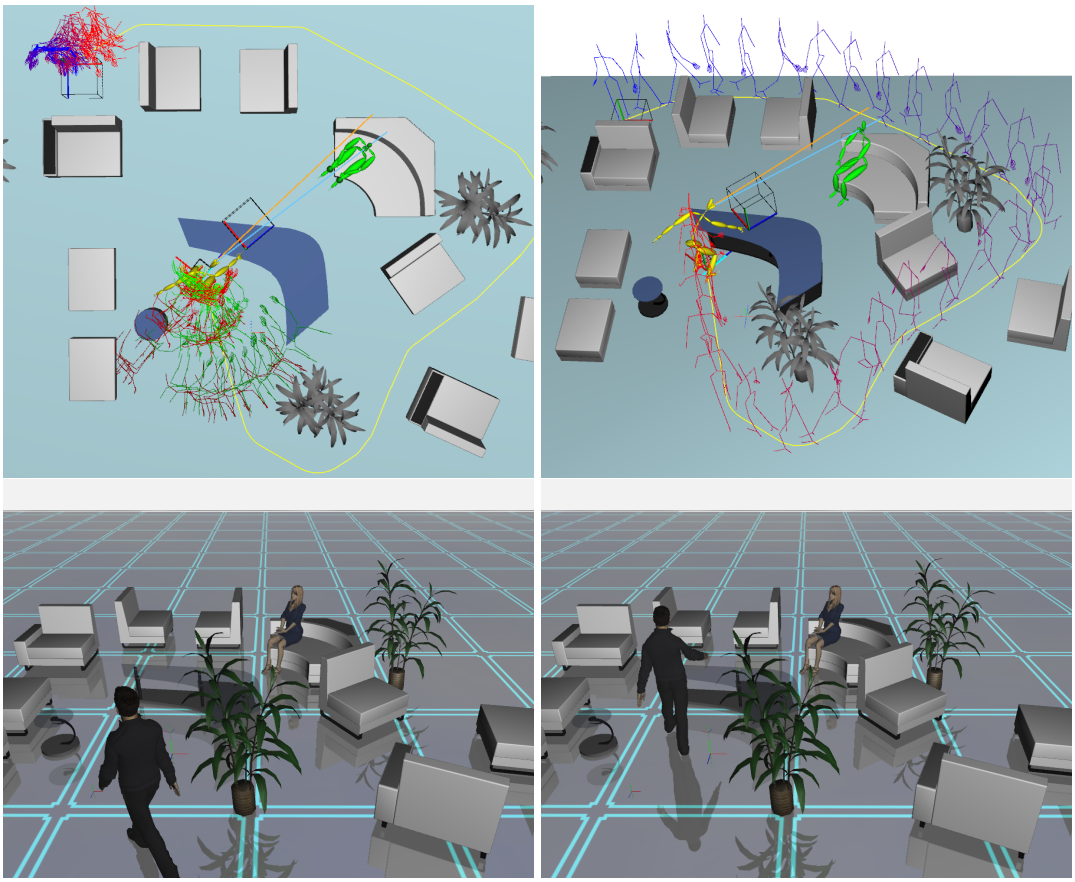


Figure 7.7: Images showing one particular locomotion planning result from different perspectives. The top two images show the arriving step database and whole planed walk sequence illustrated using skeleton trails. The bottom figures are rendered with a skinned character as the final result.

7.7 Upper-body Action Synthesis

The upper-body action and gesture (e.g. pointing) motions are computed with *Inverse Blending* (InvBld, see Chapter 3.3) with an extension to handle collision avoidance using *potential fields*. This algorithm is referred as PF-InvBld. The original InvBld is able to compute motion sequences with the end-effector precisely meeting specified spatial constraints, i.e. the end-effector of the agent would precisely reach/grasp/interact with the target at the stroke point within the action. It relies on a collection of similar and time-aligned example motions, which are realistic upper-body action instances either from motion capture or keyframe animations. With the agent at pose q_a, InvBld optimization procedure is employed to obtain an upper-body action motion precisely reaching the action target at pose q_g of the goal configuration. An error (e) evaluation function measures how well the specified constraints are enforced. Fig 7.8-left illustrates one database of example motions used in our experiments. InvBld can be computed under 2 milliseconds measured on a 2GHz quad core CPU. The routine ter-

minates when e reaches 0 or cannot be further minimized. If e is unacceptable, q_a would be discarded as candidate for body placement. Note that the optimal body positioning module queries InvBld for *comfortable zone*, the optimal distance between A and T for action execution, see Chapter 7.5 for details.

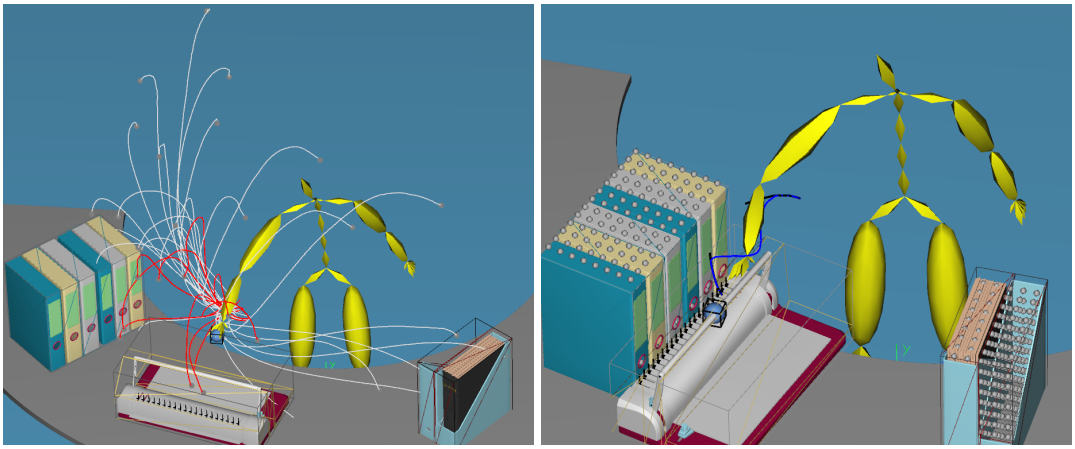


Figure 7.8: Left: result of the original InvBld. Agent's pose is a blending of 10 examples (in red) out of 36 examples with the optimal blending weights to meet the given spatial constraint, as a result the end-effector precisely reaches the target (blue ball). Right: one frame of the final planning result generated by PF-InvBld algorithm. The gradient of P_c guides the weight optimization in order to avoid collisions with the obstacles.

In our recent work [HMK11] Inverse Blending was coupled with bi-directional RRT to randomly sample the weight-time blending space in an attempt to explore the workspace and plan collision-free motion sequences (see Chapter 6). While this allows rapid exploration with human-like random posture samples, the random exploration inside the blend space may take 500 to 1000 milliseconds to finish in many cases, which would become the bottle neck if employed in the PLACE planner. The approach of force fields for reactive planning has been proposed, such as [ZLM09] where smooth vector fields are computed over cell decompositions for feedback motion planning. In this section a new algorithm is proposed named Potential Field Inverse Blending (PF-InvBld). It extends the original InvBld by incorporating the "potential of collision" (P_c) generated from obstacles inside the workspace, used to guide the weight optimization process, reactively avoid the obstacles and quickly synthesize collision-free motions. As a result, the new method is about twice faster than the blend space planner. Details of the planning process are described as follows.

InvBld is used to first compute a set of initial weights $\boldsymbol{w_g}$ that meet given spatial constraints at q_g, and the result motion $\boldsymbol{M}(t)$ is checked for collision at every frame. If it collides, PF-InvBld algorithm is started to find a collision-free solution. First, dummy spheres are generated inside workspace filling up the bounding volume of each obstacle, see Fig 7.8-Right. Next, $\boldsymbol{M}(t)$ is discretized into small segments, all colliding segments are discarded and only two valid segments are kept (Fig 7.9-top): one segment

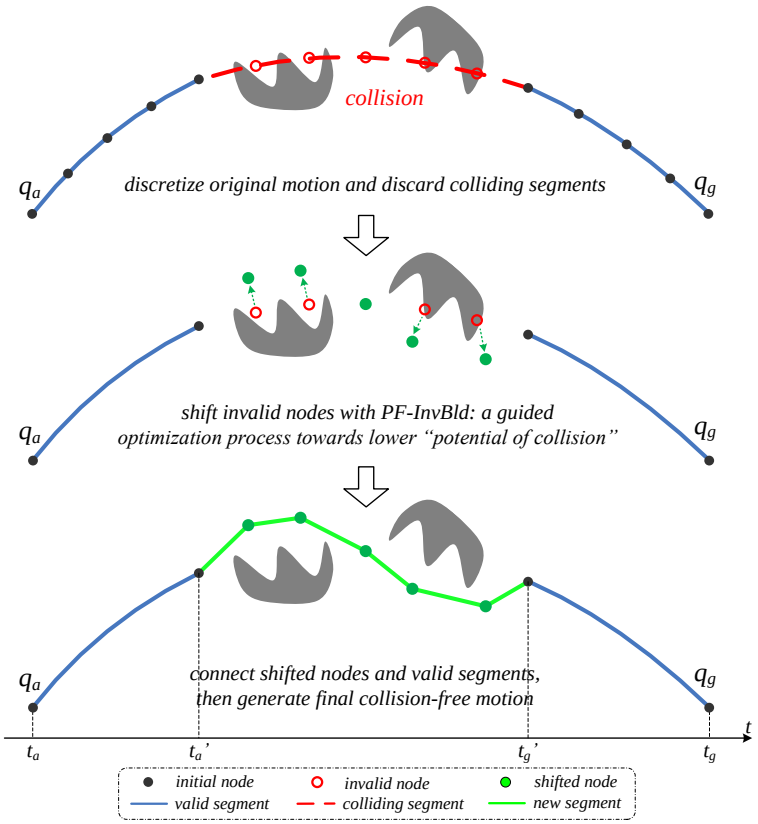
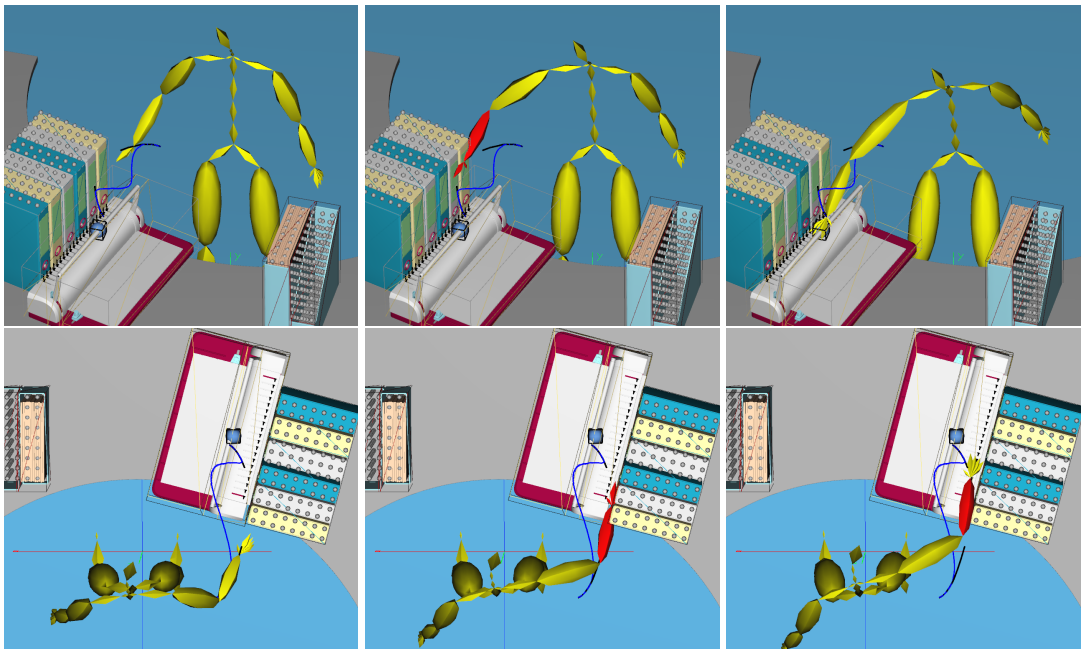


Figure 7.9: An illustration of the PF-InvBld procedure for generating collision-free upper-body actions.

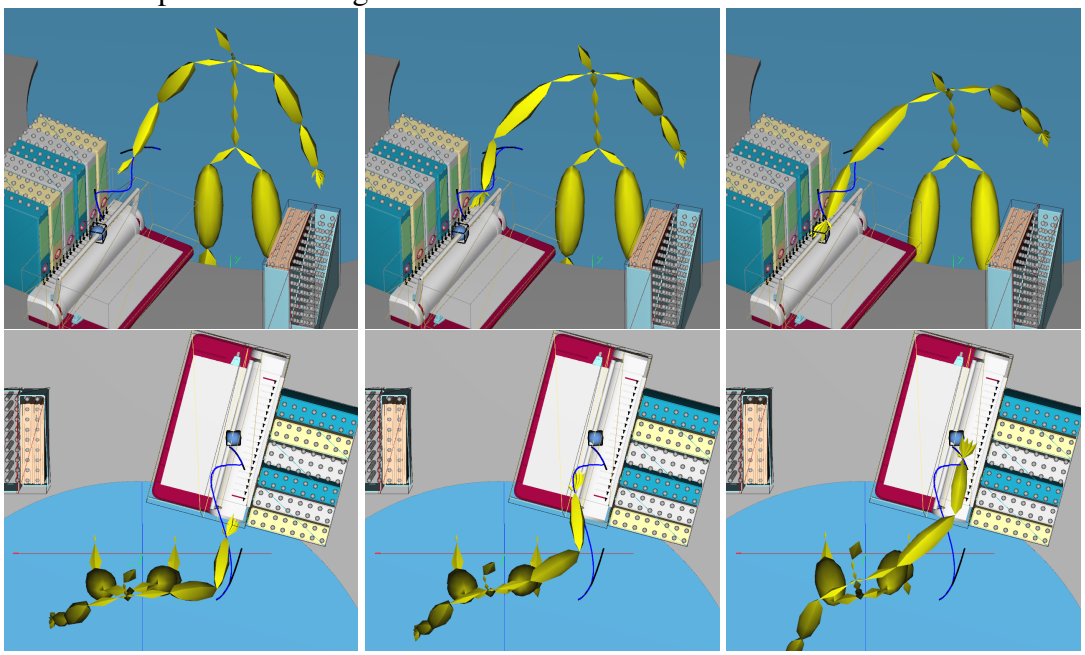
starts at rest posture q_a from t_a to t'_a; the other segment from t'_b to t_b that ends q_g. If collision happens at q_a or q_g, PF-InvBld terminates as no solution can be found.

Then PF-InvBld attempts to shift each colliding nodes $N_i(\boldsymbol{w_i}, t_i)$ out of collision along the direction of P_c towards lower potential (Fig 7.9-middle). This optimization process starts with the initial weights $\boldsymbol{w_i}$ ($\boldsymbol{w_i} = \boldsymbol{w_g}$) and stops at new weights $\boldsymbol{w'_i}$ as in $N'_i(\boldsymbol{w'_i}, t_i)$ without any collisions. Note that each $N_i(\boldsymbol{w_i}, t_i)$ corresponds to a pose generated by blending poses from the example motions at time t_i with $\boldsymbol{w_i}$.

There is no need to pre-compute the vector field for each discretized cell in the workspace (or volumetric cell in 3D). Instead, inside the weight optimization loop, PF-InvBld queries P_c which is the sum of all potentials ($\sum p_c$) generated by each dummy towards N_i. A Gaussian kernel is used to compute p_c for each N by transforming the Euclidean distance between the dummy and the end-effector location for that specific pose at N. The gradient of P_c effectively guides the weight optimization that in turn generates new poses $N'_i(\boldsymbol{w'_i}, t_i)$ with the end-effector further away from obstacles. In environment like Fig 7.8, this process typically takes 25 to 60 iterations to converge. This process is done for each N_i on the colliding segments, and terminates when all segments become valid without collisions, see Fig 7.9-bottom. Since the number of



Top two rows: original InvBld result without collision correction.



Bottom two rows: collision avoidance with PF-InvBld planner.

Figure 7.10: Action synthesis corrected by force fields in blending space. The top two rows (side view and top view respectively) show an action that produced collisions with obstacles; and bottom two rows show a collision avoidance solution (following the blue trajectory) for removing the collision.

N_i is typically less than 10, bringing the total integrations to 600 or less. Experiments show that even with the overhead of potential fields computation, PF-InvBld planner generally takes less than 300 milliseconds to plan the action motion, roughly half of the computation time or less than the blending space planner proposed in Chapter 6. More over, the planner provides more control on the overall computation time compared to the randomized sampling method, and it also reduces the collision checking overhead which is a bottleneck in our planner implementation. Final synthesis is the smooth concatenation of a series new motion segments each generated by smooth blending from N'_i to N'_{i+1}. Figure 7.10 shows the PF-InvBld results compared against original InvBld without collision correction.

Note that certain obstacle configurations may be unfavorable and unsuitable for the reactive planning nature of PF-InvBld, as the optimization procedure could get stuck in local minima regions within the potential fields. Complex environment poses greater challenges for PF-InvBld compared to our blending space planner, because the configuration space being explored by PF-InvBld is smaller. Similar to the blending space planner, in cases when no action solution can be found, the planner would restart with the next best body-positioning candidate. Our goal is to avoid generating difficult, unlikely and unnatural solutions, but favor starting over from a different initial body position that may lead to more human-like solutions.

7.8 Coordination Planner Module

The motion sequence of walk-then-action is ubiquitous in daily tasks and may seem trivial and effortless. However it is orchestrated with high level of complexity and coordination, and serves as a vital part of the PLACE planner. This section is focused on modeling the transition period between locomotion and action, which consists of 3 aspects:

(A) length of the blending window for transitions: the action motion is temporally aligned with the end of the locomotion based on the arm swing patterns as explained above. If adjustments are made to the blending window, we make sure it remains between $1/3$ to $4/3$ seconds, following the observations from the user study on blending artifacts and just-noticeable differences [WB04].

(B) starting time and length of the transition: please refer Section 7.9 and Fig 7.12.

(C) arm swing patterns modeling dur transition: there are two arm swings within the transition period of walk-then-action (See Fig 7.11): locomotion arm swing (S_l, blue) at the end of walk sequence, and action arm swing (S_a, red) at the beginning of upper-body action. Based on observations of motion capture from human subjects, S_l naturally transitions into S_a following specific patterns. As an example, it is assumed that upper-body action is executed using the right arm.

In Fig 7.11-(a), S_a naturally carries on the swing of S_l, which yields desired humanlike result. The case illustrated in Fig 7.11-(b) is not humanlike because S_l and S_a

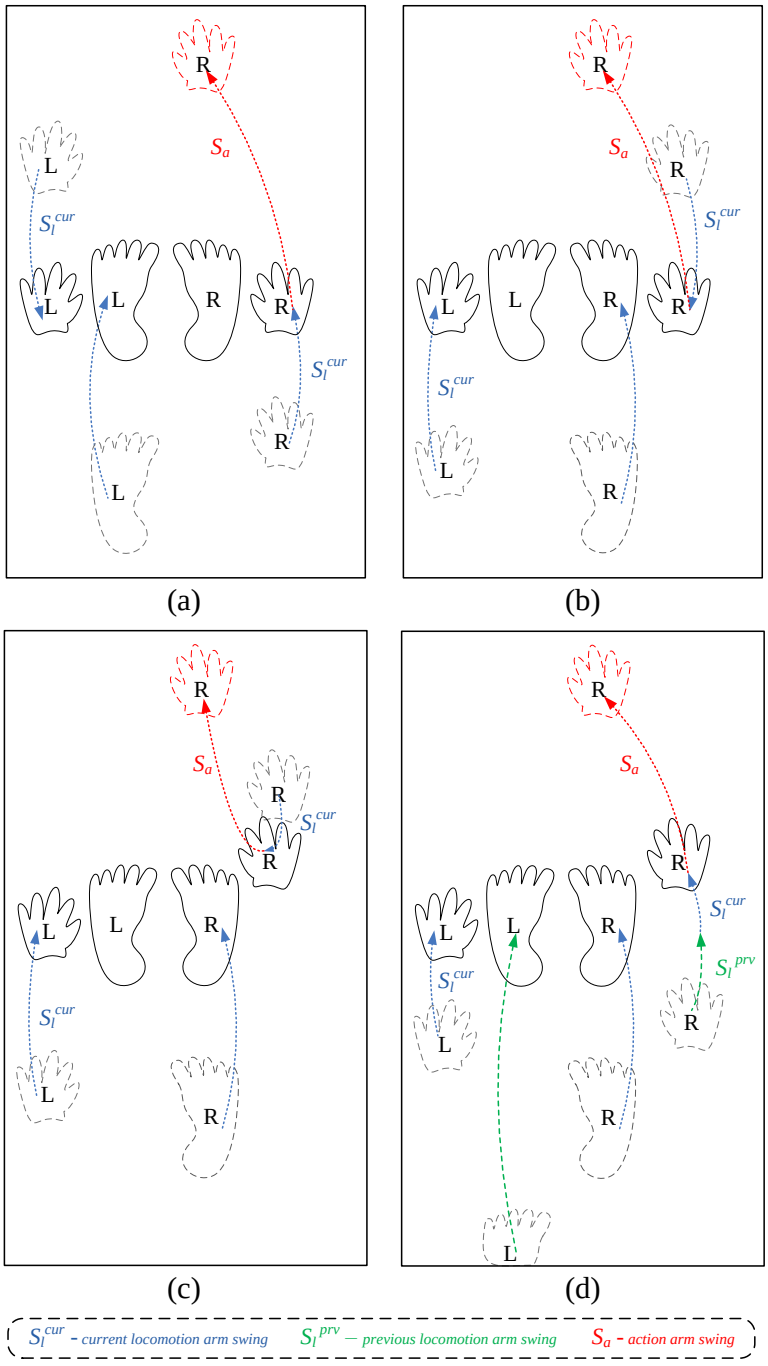


Figure 7.11: An illustration of the arm swing patterns within the transition period as part of the coordination model. The goal is to smooth the arm swing transition from end of locomotion S_l into beginning of upper-body action S_a, by either delay the arm from returning to neutral position as shown in (c), or delay crossing the neutral position shown in (d).

are in opposite directions. Instead, the natural transition lies in one of the following two options in order to avoid sudden changes of arm swing direction:
(1) S_l is shortened and blended into S_a without the right arm returning to neutral position, shown in Fig 7.11-(c).
(2) S_l is extended from the previous swing cycle and overrides the current swing cycle, then blended into S_a, shown in Fig 7.11-(d).

Within the transition, the planner smoothly blends the whole-body motion from locomotion into action, taking into account the blending parameters described in (A) and (B). Inverse Kinematics is applied to fix both feet on the floor while movements on the hips joint are allowed by the action in case the character needs to bend the knees to reach targets close to the floor, generating a final coherent whole-body action (this is due to our action motion is also full-body). Arm swings are handled as described in (C), and unnatural swings shown in Fig 7.11-(b) is corrected with either (c) or (d) depending on the duration of the final walk cycles. As a result, the transition appears smooth and human-like, with better visual quality than simple full-body blending between two sequences. More results of the smooth transition are presented in Figure 7.16 and 7.17.

7.9 Engagement Planner Module

The visual engagement (i.e. gaze behaviors) was modeled as part of the overall planning result and applied on top of the whole-body synthesis. Realistic engagement modeling contributes greatly to the human-likeness of the final synthesis due to its ability to aid with basic two-way communications by directing attention, and also to disambiguate for listeners [Ken90b]. In the experimental settings, higher-level gaze behaviors are analyzed and the synchronized with the stroke point of the actions addressing specified target object in the workspace. It could be observed that each trial typically consists of a series of largely consistent gaze and gaze-related events, as listed below:

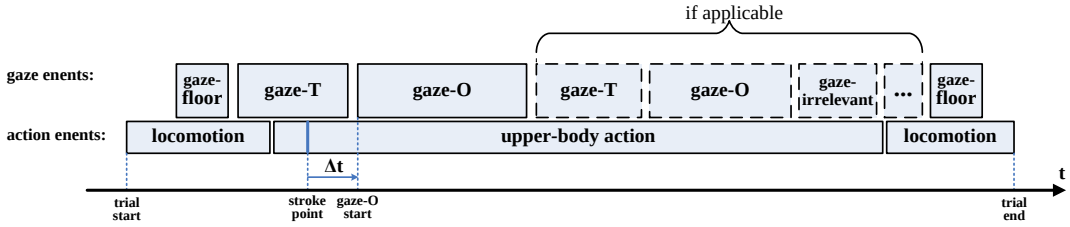


Figure 7.12: This figure illustrates a typical serious of events occurred in one captured trial. Drawing is not to scale.

1. $gaze\text{-}floor$: participant gazes at the floor when walking towards T;
2. $gaze\text{-}T$: participant gazes at T while gesticulatating;
3. stroke point of the action (here we only focus on the first stroke);

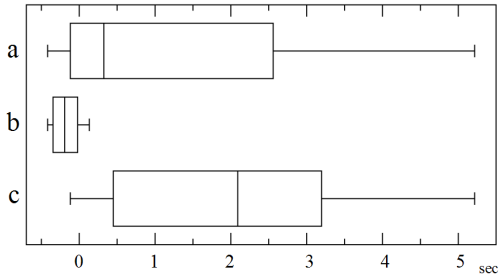


Figure 7.13: Temporal delay Δt dictates if the start of $gaze$-O occurs before ($\Delta t < 0$) or after ($\Delta t > 0$) the action stroke point: (a) box plot of Δt across all trials. (b) box plot of Δt for only out-of-FoV observer positions O_{P1} and O_{P5}. (c) box plot of Δt for only inside-FoV observer positions O_{P2}, O_{P3} and O_{P4}.

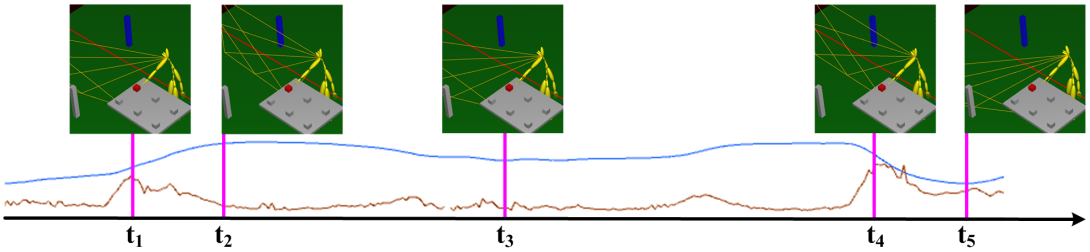


Figure 7.14: The velocity profile for participant's head rotations observed from captured gaze behavior (unfiltered). The lower brown line logs the angular accelerations of the head rotations. The upper bell-shaped blue line (filtered) reflects the head rotation angle pattern used to simulate the natural gaze behavior. Along the time line: t_1 is start of $gaze$-O; t_2 is the apex of $gaze$-O; t_3 shows a $gaze$-T during target demonstration; t_4 is the end of $gaze$-O; t_5 is the start of another $gaze$-T.

4. $gaze$-O: participant gazes at O while describing T;
5. $gaze$-T: participant again gazes at T during action, if applicable;
6. $gaze$-O: participant again gazes at O during action, if applicable;
7. $gaze$-$irrelevant$: participant gazes at additional locations, if applicable;
8. $gaze$-$floor$: participant gazes at the floor when stepping back to initial location.

The temporal parameters for gaze behavior modeling have been specifically studied mainly in three aspects. First aspect is the temporal delay (Δt) between the action stroke point and the start of $gaze$-O event, see Figure 7.12. Annotation results show that when O is positioned within participant's field-of-view (FoV) (i.e. O_{P2}, O_{P3}, O_{P4} in Fig 7.3-left), $gaze$-O immediately follows the stroke point, resulting in $\Delta t > 0$. By contrast, when O is outside of FoV (i.e. O_{P1} and O_{P5}), due to the large gaze-shift required to visually engage with O, $gaze$-O starts ahead of the stroke point, and in this

case $\Delta t < 0$. This temporal delay extracted from the trials (measured in seconds) is plotted in Figure 7.13. The second aspect is from our pioneering study of the gaze durations for repeated upper-body actions. It was observed that the gaze durations tend to decline over time, and they tend to follow certain surge-then-decline patterns each time the observer switches to a new location. One possible cause for such patterns is the familiarity established over time between the human subjects and the observer, please refer [HKM11] for details.

Lastly, to simulate humanlike head rotations for the gaze, a velocity profile similar to [YKH04] is extracted from captured data and used to dictate the head rotations for the virtual agent, see Figure 7.14. The bell-shaped blue line reflects the angular pattern of smooth filtered head rotations, which is used to simulate the natural gaze behavior.

These temporal parameters (refer [HKM11] for details)) are used to schedule gaze behaviors to add on top of the full-body motion generated by the other modules. The maximum gaze-shift angle ϕ is computed with the body positioning module from Chapter 7.5. Together with the bell curve extracted from motion capture for synthesis of the actual head rotations, our engagement module is able to generated human-like gaze behaviors closely resemble those observed from the real captures.

7.10 Results and Discussion

This chapter presents the **PLACE** planner, a novel whole-body motion planner for demonstrative agents that incorporates five aspects: 1) optimal body positioning, 2) locomotion, 3) upper-body action planning, 4) transition/coordination between locomotion and action, 5) visual engagement modeling. The proposed planner is original in the sense that it offers a solution to tackle the whole-body planning tasks that involve all five aspects with high-level coordinations. Results from the overall planning are presented in Figure 7.16 and 7.17. The main contribution is that, to the best of our knowledge, our work is the first whole-body planner to incorporate precise body positioning models in agent-agent interactions, and also the first to simulate transitions and coordinations between locomotion and action sequences. The overall task is broken down into individual parts, each solved by a sub-module of the planner. Certain modules are derived and modeled using experimental data collected from human subjects. As a result, behaviors of the human participants are correctly simulated in similar virtual settings with the ability to produce high-fidelity motion synthesis. The proposed planner is capable of synthesizing the whole sequence with the computation time range from 100 to 400 milliseconds depending on the complexity of the environment (100 ms when no upper-body collision occurs, and about 400 ms when PF-InvBld is involved in the planning process), ideal for interactive training and simulation applications.

Due to the nature of the problem modeled, there are several limitations of the proposed planner:

(1) the visibility constraint between observer and target is not modeled as it is the

observer's responsibility to maintain such visibility, so is agent-target visibility due to the proximity of the two;

(2) the planner assumes the agent is able to interact with the target from all directions when searching for optimal body positioning described in Section 7.5;

(3) the experimental data is collected in one-on-one interaction scenarios only, hence at this point we do not attempt to extend this model to address audience with multiple observers, but it certainly is an interesting direction for future work.

Figure 7.15: Sequenceof a long-range locomotion planning. First row shows the beginning of the walk with a very sharp turning, and the last two rows focuses on the transition into arriving steps.

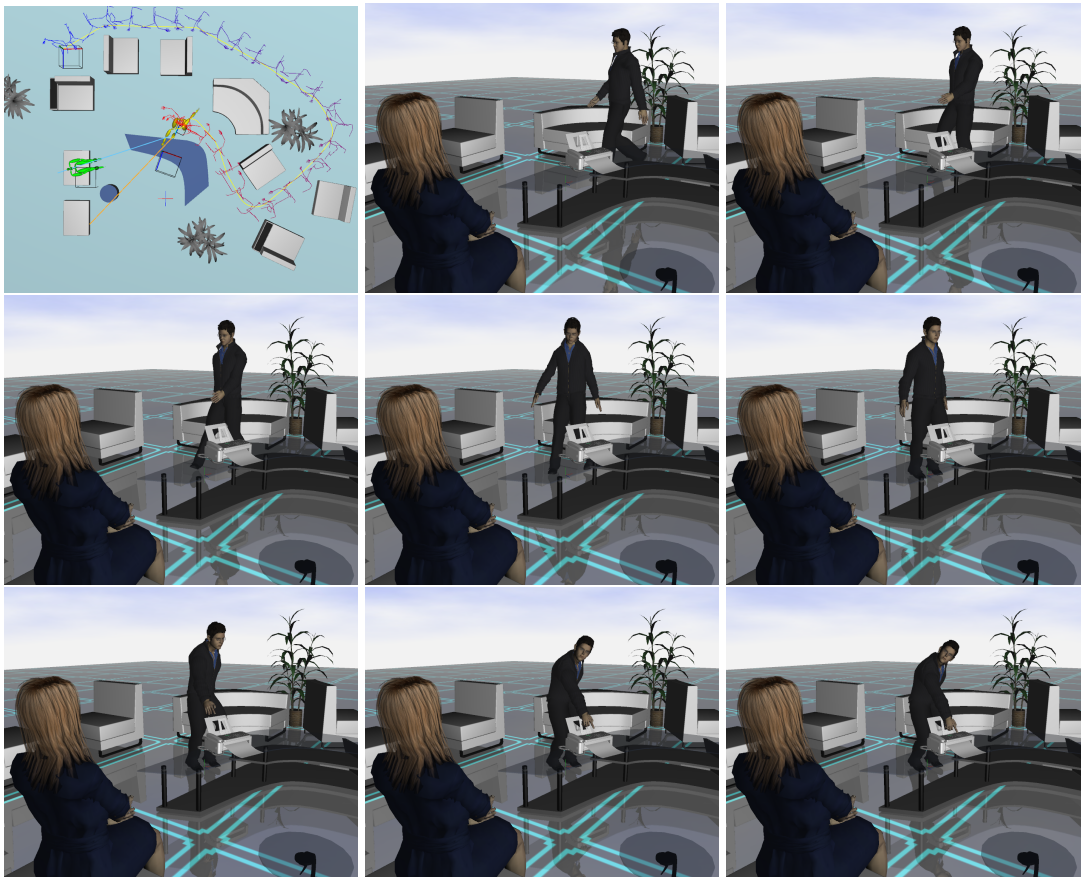


Figure 7.16: Example of a solution produced by PLACE. The top-left image shows the planning scenario and the solution placement for execution of the demonstration. The following sequence of snapshots shows the arrival locomotion seamlessly transitioning into the demonstration action pointing at the fax machine with coordinated gaze towards the observer. Second row shows the arriving steps with coordinated arm swing blended into the beginning of the upper-body action.

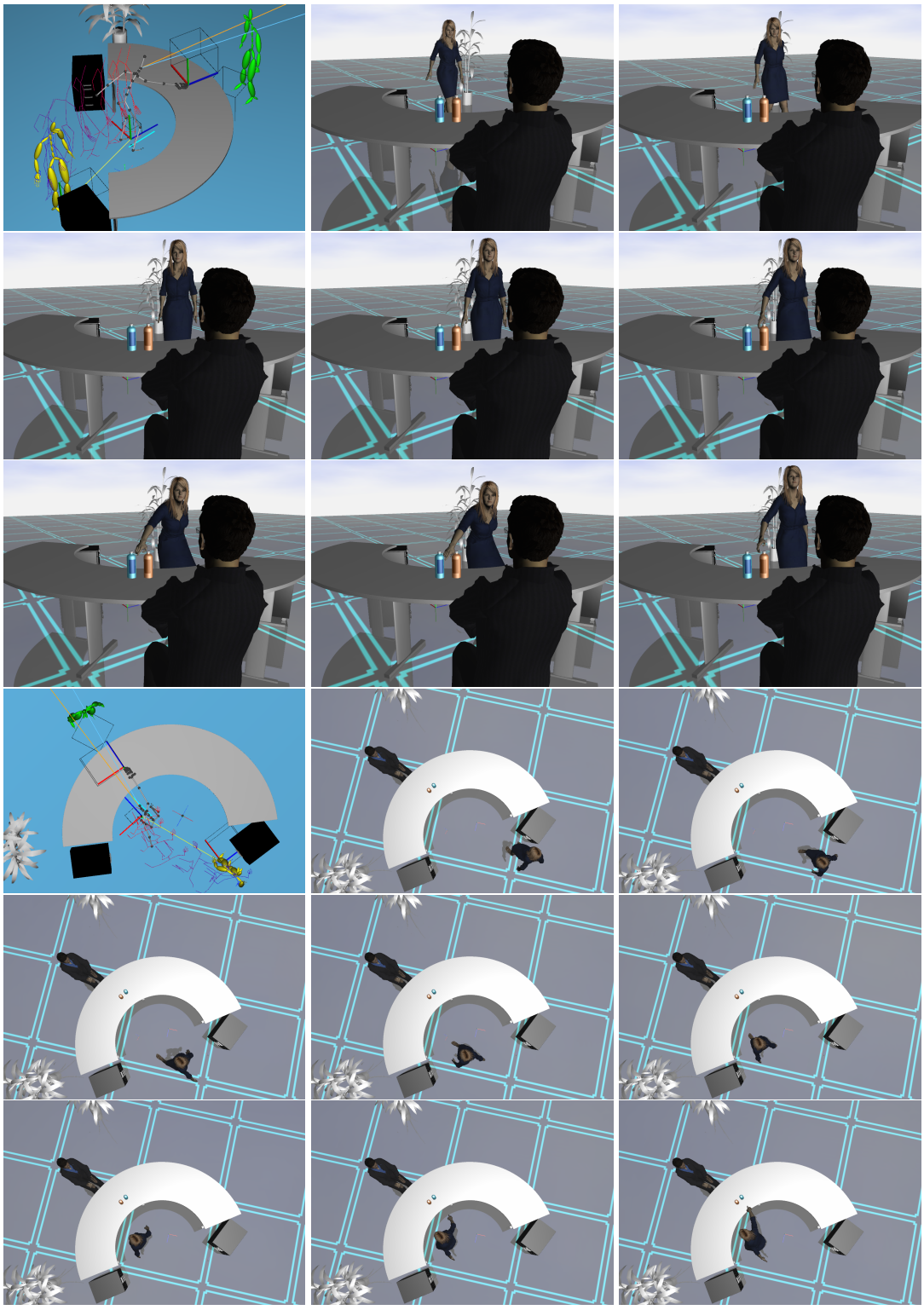


Figure 7.17: This solution shows a short-range locomotion towards a placement suitable for pointing and describing the blue bottle. Bottom three rows are taken from top view.

CHAPTER 8

Automatic Retargeting

Humanoid 3D models can be easily acquired through various sources, including online. The use of such models within a game or simulation environment requires human input and intervention in order to associate such a model with a relevant set of motions and control mechanisms. This chapter demonstrates a pipeline where humanoid 3D models can be incorporated within seconds into an animation system, and infused with a wide range of capabilities, such as locomotion, object manipulation, gazing, speech synthesis and lip syncing. A set of heuristics are offered to associate arbitrary joint names with canonical ones, and describe a fast retargeting algorithm that enables us to instill a set of behaviors onto an arbitrary humanoid skeleton. We believe that such a system will vastly increase the use of 3D interactive characters due to the ease that new models can be animated.

8.1 Motivation

3D characters are commonly seen in video games, feature films, mobile phone applications and web sites. The generation of an expressive 3D characters requires a series of stages, including the generation of a character model, specifying a skeleton for that model, deforming the model according to the movement of the skeleton, applying motion and control algorithms under a framework, and finally instructing the character to perform. Each of these processes requires a different skillset. For example, 3D models are generated by digital modelers or through hardware-based acquisition, while animators create or apply motion to the characters.

Thus, while many high quality assets such as humanoid models or motion capture data can be readily and inexpensively acquired, the integration of such assets into a working 3D character is not automated and requires expert intervention. For example, after motion capture data is acquired, it then needs to be retargeted onto a specific skeleton. An acquired 3D humanoid model needs a skeleton that satisfies the constraints of a real-time game system, and so forth. Modern game engines provide a means to visualize and animate a 3D character, but require assembly by a programmer or game designer. The complexity of animating 3D virtual characters presents an obstacle for the end user, who cannot easily control a 3D character without the assistance of specialists, despite the broad availability of the models, assets and simulation environments.

This chapter presents a system to address this problem, which facilitates the rapid incorporation of high-fidelity humanoid 3D models into a simulation. Characters intro-

duced to our system (SmartBody [USC12]) are capable of executing a wide range of common human-like behaviors. Unlike a traditional pipeline, SmartBody requires no intervention from artists or programmers to incorporate such characters after the assets have been generated. Our pipeline relies upon two key automated processes:

1) An automated skeleton matching process; skeletons are examined to find a match between the new skeleton, and one recognized by the simulation. Such a process looks for similarly named joints, as well as relies on expected topology of humanoid in order to recognize similarly functioning body parts.

2) A retargeting process that can transfer high quality motion sets onto a new character without user intervention.

In addition, the virtual character's capabilities are generally based on two different sources:

A) A set of controllers that can generate motion by means of rules, learned models, or procedurally-based methods, and

B) A set of behaviors generated from animation data that can be parameterized across various dimensions, such as running speed for locomotion, or reaching location for interaction with other 3D objects.

8.2 Related Work in Motion Retargeting

The first stage of SmartBody utilizes an automated mapping process which uses a set of heuristics for mapping an arbitrarily named humanoid skeleton onto a common skeleton with familiar names. To our knowledge, no such algorithm has been previously published. Many other methods for registering skeletons require matching names or manual annotations [MAF11]. At the time of this writing, [Uni12] demonstrates a process by which a skeleton can be automatically registered, but no technical details are provided regarding underlying algorithms and robustness. In addition, systems such as [AI11] attempt to automate the acquisition of motion and models, but have not seen any details regarding the skeleton rig recognition step.

The second stage of SmartBody utilizes a fast, but offline retargeting system to generate animations appropriate for a particular skeleton. Retargeting has been an area of much research in the animation community since Gleicher [Gle98]'s work which uses optimization and low-pass filtering to retarget motion. Many other retargeting algorithms use various approaches: Kulpa [KMA05] retargets motion by using angular trajectories, and then solve several body areas, Less [LS99] uses a hierarchical approach to retargeting, Mozani [MBB00] uses an intermediate skeleton and IK to handle retargeting between skeletons with different topologies. Kulpa [KMA05] retargets motion through a morphology-independent representation by using angular trajectories, and then solving several body areas. Ho et al. [HKT10] uses spatial relationships for motion adaptation which can handle many contact-based motion retargeting problems. Zordan [ZV03] retargets optical data directly onto a skeleton via a dynamics-based

method. Shin [SLS01] uses an online retargeting method via an analytical IK method that prefers the preservation of end effector values. Choi [CK99] uses a Jacobian-based IK method for online retargeting.

Our retargeting system attempts to find footplants in order to better retarget movement. An online footplant detection and enforcement method is presented in Glardon's work[GBT06]. By contrast our retargeting method enforces footplants offline, and doesn't modify the length of limbs as in [KSG02] so as to be compatible with many game and simulation skeleton formats. Similar to our goals, the work in [HRE08] is focused on retargeting to creatures with a varying morphology, such as differing number of legs, tails or the absence of arms. The system described in that work relies heavily on inverse kinematics in performing online retargeting based on semantic descriptions of movement. By contrast, what SmartBody needed is offline but relatively fast retargeting of high quality motions onto humanoid characters that cannot be achieved via simple walk cycles and reaching constraints. [MAF11] develops a system to automatically assemble a best-fitting rig for a skeleton. By contrast, SmartBody assumes the skeleton and model have already been connected, and focus on the use of such skeleton in real time simulations.

The characters in SmartBody can be instructed to perform certain behaviors using the Behavioral Markup Language (BML) [KKM06]. BML provides a high-level XML-based description of a set of tasks that can be synchronized with each other. Many systems have been developed to utilize BML, such as EMBR [HK09a], Elckerlyc [WRR10], Beat [CVB01] and Greta [NBM09] in addition to SmartBody [TMM08, Sha11]. However, to our knowledge, no other BML-based systems besides our own have implemented extensive character locomotion capabilities or generic capabilities such as object manipulation [FXS12] which are associated with large sets of behaviors. Since the BML specification emphasizes speech, head movements and gestures, most BML-compatible systems emphasize only those features.

8.3 Automatic Skeleton Joint Mapping

One of the challenges of using an off-the-shelf character model is that the user has to first set up a joint mapping table to comply with the skeletal system and motion data used for the target system/application. This step is critical for many motion parameterization procedures like retargeting, and although being a trivial task, it is commonly done by hand. Existing commercial tools and applications (e.g. MotionBuilder [Aut]) have incorporated "automatic mapping" features where such mapping is estimated for skeletons not following MotionBuilder naming conventions, providing a starting point for manual While the underlying mechanism for this MotionBuilder automatic mapping is unknown, it is believed that from the generated mapping result it could be base on joint name searching, similar to Animeeple [AI11] which uses regular expression joint name searching.

In this chapter a heuristic-based automatic skeleton joint mapping has been pro-

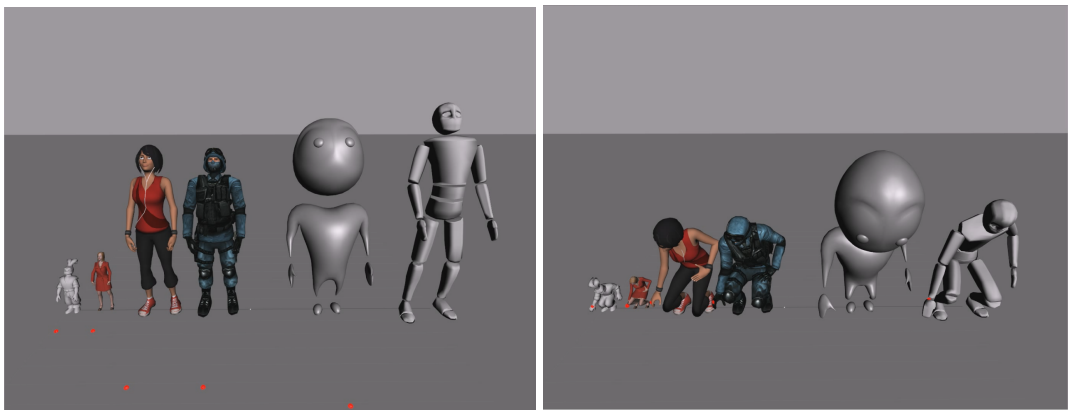


Figure 8.1: A set of characters from many different sources are automatically retargeted and registered into SmartBody. The characters can now perform a number of tests with controllers and parameterized motions in order to insure that the behavior has been properly transferred: gazing, object manipulation, locomotion, head nodding, etc.

posed. This method utilizes the skeleton hierarchy structure and symmetries, combined with keyword searching to help determine certain key joints in the hierarchy. The automatic mapping method has been successfully validated using character skeletons from various popular sources (mixamo.com, rocketbox-libraries.com, turbosquid.com, axyz-design.com, 3DSMax, MotionBuilder), results shown in Fig 8.1, 8.6.

The goal is to map a list of arbitrary joints from any user-defined biped skeleton to the set of canonical joints on the target skeleton inside our character animation system. Fig 8.2 shows the final mapping result to be achieved from left side mapped to the right side. Left side as an example follows MotionBuilder[Aut] standard skeleton joint naming convention, and right side is the corresponding names in the SmartBody standard skeleton. SmartBody does not intend to map all the joints, and in many cases not all joints can be mapped easily. Only a basic set of joints are mapped which would enable most of our controllers to drive user-defined characters for behaviors like gaze, reaching and locomotion.

Algorithm 4 Search routine for arm joint-chain.

```
1.  J ← acromioclavicular (AC)
2.  while J ← J.child() do
3.      if J has 5 children then
4.          MAP(wrist, J)
5.      else if J.num_children() = 0 then
6.          J ← J.parent()
7.          MAP(wrist, J)
8.      end if
9.  end while
10. if not wrist.mapped() then
11.     return WRIST_NOT_FOUND
12. end if
13. J1 ← shoulder ; J2 ← wrist
14. switch (J2.depth − J1.depth)
15. case 2:
16.     uparm ← shoulder
17.     MAP(uparm);MAP(elbow)
18. case 3:
19.     MAP(uparm);MAP(elbow)
20. case 4:
21.     MAP(uparm);MAP(elbow)
        if forearm then MAP(forearm)
22. case 5:
23.     MAP(uparm);MAP(elbow);
        MAP(forearm)
24. end switch
```

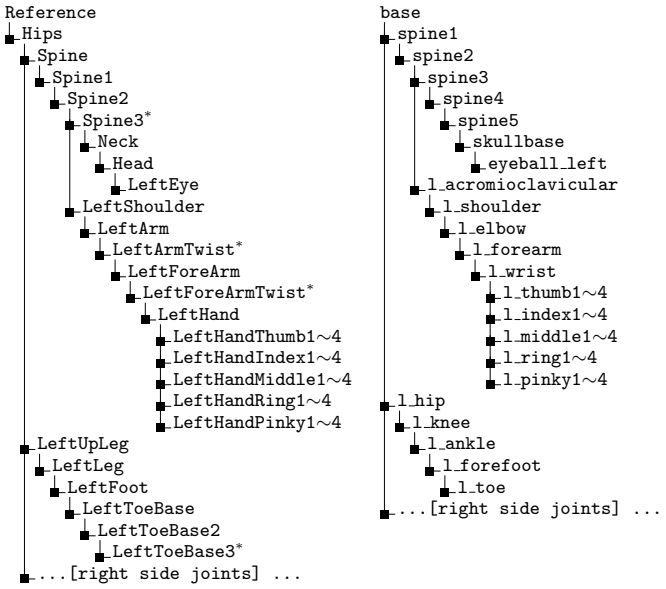
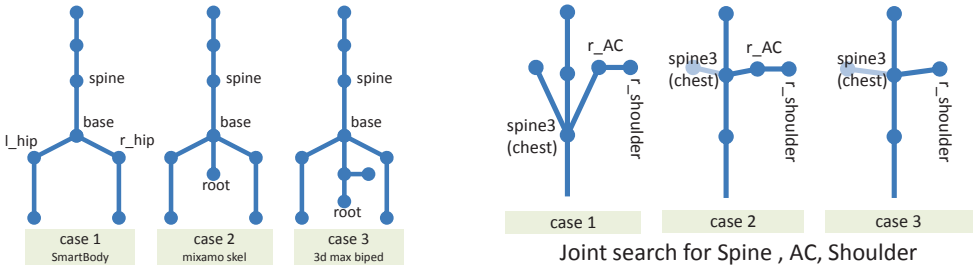
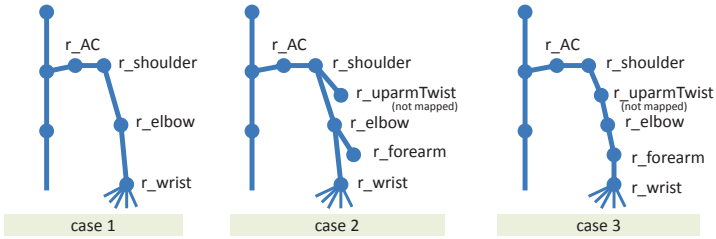


Figure 8.2: Final mapping result achieved by SmartBody, left side is given as an example following MotionBuilder naming convention, right side is the corresponding joint names in SmartBody. * denotes joint (if exists) is skipped as it's not handled by SmartBody.

The mapping is largely based on heuristics and is specifically adapted to our system. The first step is to find the character's base joint. The situation is considered only where the input skeleton is biped, in which case the base is usually defined as the parent of spine and two legs. Fig 8.3 generalizes some of the variations found in our testing skeletons, and the routine is partially outlined in Algorithm 2. Once the base joint is found, our algorithm tries to map the remaining key joints based on the symmetry/hierarchy of the canonical target skeleton and the assumption that source skeleton will have similar properties. A portion of this procedure is shown in this chapter due to the redundancies in Fig 8.3, Algorithm 3 and 4 outline part of the search routines for spine/chest and arm joint-chain respectively, however more complicated cases are also handled. As an example, if two joints are found sharing the same parent joint (chest), both have the same depth also the same number of children joints, the algorithm will assume they are either acromioclavicular or shoulder, and then determine left/right using their joint names. Another example is that based on the depth of shoulder and wrist in the hierarchy, the heuristic determines if twist joints are present in-between and estimates the mapping accordingly. In certain cases the heuristics may rely on keyword search inside joint names to determine the best mapping, but switches to purely hierarchy-based mapping





Joint to search	Search keyword		Alternatives
skullbase	head	skull	
l_wrist	hand	wrist	wrist
l_ankle	ankle	foot	
l_thumb	thumb	finger0	pollex
l_index	index	finger1	pointer, forefinger
l_middle	middle, mid,	finger2	medius
l_ring	ring	finger3	fourth
l_pinky	pinky	finger4	little
eyeball_left	eye (but not lid, brow, lash)		

Keywords used in joint search

Figure 8.3: An illustration of various configurations generalized from testing skeletons for certain key joints and joint-chain mapping using heuristics.

when not successful. Characters with uncommon hierarchy/joint names may break the heuristics, such as with the presence of extra joints (wings, tails, etc) or asymmetrical hierarchy, in which case the user needs to manually complete the mapping starting with the partial mapping table generated by the algorithm.

8.4 Retargeting

The motion retargeting process works by transferring an example motion set from our canonical skeleton to a custom skeleton provided by the user. The retargeting process can be separated into two stages. The first stage is to convert the joint angles encoded in a motion from our canonical skeleton to the custom skeleton. The second stage is to enforce various positional constraints such as foot positions to remove motion artifacts such as foot sliding.

Algorithm 2 Search routine for base joint.

```
1.  while i ≤ max_search_depth do
2.      J ← skeleton.joint(i)
3.      switch (J.num_children())
4.      case 2:
5.          if J has 2 symmetrical children then
6.              return MAP(base, J)
7.          end if
8.      case 3:
9.      if J has 2 symmetrical children then
10.             return MAP(base, J); MAP(spine)
11.     end if
12.     end switch
13. end while
14. return BASE_NOT_FOUND
```

Algorithm 3 Search routine for spine, chest, acromioclavicular and head joints.

```
1.  J ← base
2.  while J ← J.child() do
3.      if J.num_children() ≥ 2 then
4.          MAP(Spine4, J) {chest joint}
5.          break
6.      else
7.          MAP(spine#, J)
8.      end if
9.  end while
10. if J has 2 symmetrical children then
11.     MAP(AC, J.child())
12. end if
13. if J.child().name() = "Head" then
14.     MAP(skellbase, J.child())
15. end if
```

8.4.1 Motion Data Transfer

The goal of this stage is to transfer the motion data such as joint angles from a source skeleton to a target skeleton. Joint values can be directly copied over for skeletons with aligned local frames and initial T-poses. However in most cases, a skeleton provided by the user tends to have different setup and default pose from our canonical skeleton. Therefore the procedure first needs to align the default pose between the target skeleton and our canonical skeleton. This is done by recursively rotating each bone segment in target skeleton to match the global direction of that segment in source skeleton at default pose (Fig 8.4 left) so that the target skeleton is adjusted to have the same default pose as the source skeleton.

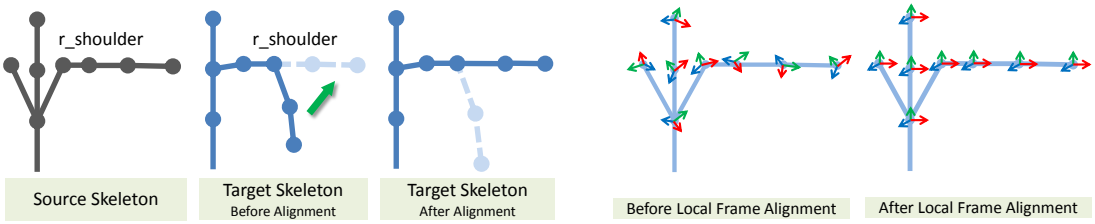


Figure 8.4: Left side shows alignment of a bone segment between two skeletons so that target skeleton matches the pose of source skeleton. Right side shows re-orientation of joint local frames so that they align with the canonical world frame, which enables straightforward transfer of motion data from source to target skeleton.

Once the default pose is matched, the method addresses the discrepancy between their local frames by adding suitable pre-rotation and post-rotation at each joint in target skeleton. Specifically, given a joint b_i, with its global rotation R^G and initial local rotation q^{init} when in default T-pose, its local frame is re-oriented as

$$q' = q^{init}\ R^{G^{-1}}\ q\ R^G$$

, where q' is the actual local rotation after re-orientation, and q is the standard rotation that complies with the default global frame. In other words, the original local frame of b_i is re-oriented to align with the default canonical global frame as shown in Fig 8.4 right, e.g. a left 30° turn around Y-axis in Y-Up global frame simply means setting $q = quat((vec(0,1,0),30)$ without considering the initial rotation of b_i. Since our canonical skeleton already has all of its joint local frames aligned with the global frame, this in turn aligns joints in both skeletons into the same local frames. Therefore the motion data transfer can now be done trivially by copying the joint rotations to the target skeleton. Similarly, the root translation p_r can also be transferred to the target skeleton by scaling it according to the length of legs between two skeletons. The scale factor s_r is computed as $s_r = \frac{l_t}{l_s}$, where l_t is the leg length of target skeleton and l_s is that of source skeleton. For motions created specifically for skeletons with non-canonical alignments, the re-orientation process is reversed as

$$q = R^G \; q^{init^{-1}} \; q' \; R^{G^{-1}}$$

to make these motions become aligned with default global frame, which can be directly applied to any skeleton after realignment in a very straightforward fashion.

8.4.2 Constraint Enforcement

Once motion data is transferred, they would serve as a rough approximation to enable the target skeleton with various behaviors such as locomotion. However the transferred motion may not work perfectly on the target skeleton due to different limb lengths, which may result in foot sliding artifacts, etc. This problem could be seen in many kinds of motions after naive data transfer but is mostly visible among locomotion sets. In order to alleviate these artifacts, inverse kinematics is applied to enforce the foot plant constraint in the transferred motions. The inverse kinematic method used in the system is based on Jacobian pseudo-inverse,

$$\Delta\Theta = J^+\Delta\mathbf{x} + (I - J^+J)\Delta\mathbf{z}$$

, where $J^+ = J^T(JJ^T)^{-1}$ is the pseudo-inverse of Jacobian matrix J, $\Delta\mathbf{x}$ is the offset from current end effector coordinates to target coordinates $\mathbf{x_r}$, and $\Delta\mathbf{z}$ is the desired joint angle increments toward target pose $\mathbf{z} = \tilde{\Theta}$. The above IK method deforms an input pose to satisfy the end effector constraint, while maintaining the target pose $\mathbf{z}$ as much as possible. This IK method is applied at each motion frame in the locomotion sequences to ensure the foot joint is in the same position during the foot plant stage.

Previous methods exist for detecting and fixing foot sliding [KSG02, GBT06]. They mostly work by finding a time range over which the foot plant occurs, and enforce the foot plan during that period. Additional smoothing is usually required to ensure that the constraint enforcement does not create popping artifacts in the motion. Experiments show that it is difficult to robustly detect foot plant range across different type of motions. Also, without careful consideration, smoothing may create more motion artifacts

if foot plant is not correctly found. Since the original motion is assumed to be smooth and free of feet-sliding, we chose to warp the original motion trajectory and enforce constraints over the whole trajectory. Let $p_s(t), p_d(t)$ be the foot position trajectory for source and target skeleton. A new trajectory is created for target skeleton by warping the original trajectory using the following equation,

$$\begin{aligned} p'_d(0) &= p_d(0) \\ p'_d(t+\delta t) &= p'_d(t) + s_r(p_s(t+\delta t) - p_s(t)) \end{aligned}$$

, where p'_d is the new target trajectory, and s_r is the scaling factor based on leg length from the previous section. The above equation warps the foot trajectory from original skeleton based on the scale of target skeleton. This was proven to work well during our experiments on various skeletons with different limb lengths and proportions. Good robustness was shown on different motion styles with no additional smoothing needed.

8.5 Discussion and Conclusion

The pipeline described in this chapter is designed for incorporating high-quality humanoid assets into a virtual character and quickly infuse that character with a broad set of behaviors that are common to many games and simulations. We believe that by automating the incorporation of models, we are lowering the barrier to entry for end users and potentially increasing the number and complexity of simulations that can be generated.

Our skeleton mapping algorithm is limited to humanoid or mostly humanoid forms. It assumes that characters have human-like structure: two arms, two legs, shoulders, elbows, knees and so forth. In addition, many controller-based behaviors require a minimum configuration of joints in order to be fully-realized. For example, the gaze control requires a number of joints, stretching from the lower back to the eyes in order to gaze while engaging several body parts at once. Also, the behavior sets that rely on single or parameterized motion sets require a reasonable match between the original motion subject on which the data was captured, and the targeted skeleton. If the skeleton topology or bone proportions fall too far outside of normal human limits, the appearance quality of the behavior will be deteriorated.

By providing an automated means to transfer a set of motions, and potentially, a set of behaviors, onto a character, we envision the widespread development of behavior libraries separate from a particular game or simulation. As digital artists create libraries of models for generic use, so too can motion developers capture or design a library of animations for generic use as well. Thus, experts in crafting motion can create both stylized or context-specific motion sets. Game or simulation designers can then choose from a set of motions in the same way that they can choose from a set of models. By loosening the bond between the motion and the model, the use and reuse of digital assets can thus be greatly increased.

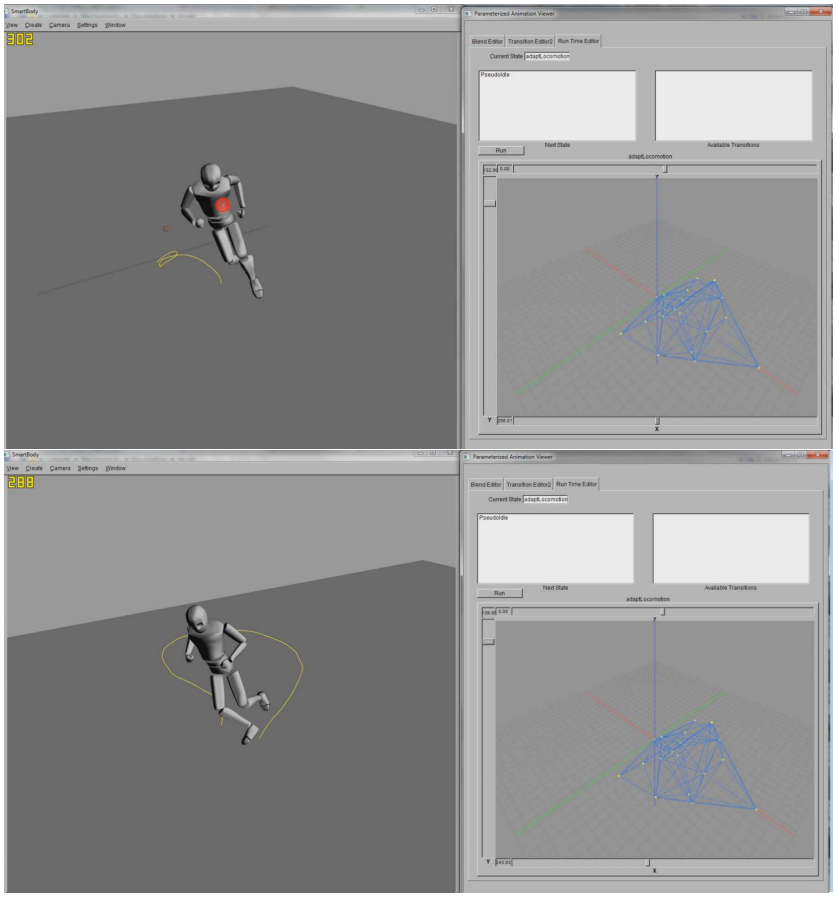


Figure 8.5: In the figures above, a set of 20 motion captured locomotion animations are mapped to drive an arbitrary character. The motion captured locomotion data set is of much higher visual quality than can be generated via procedural techniques such as through the use of IK or footstep models.

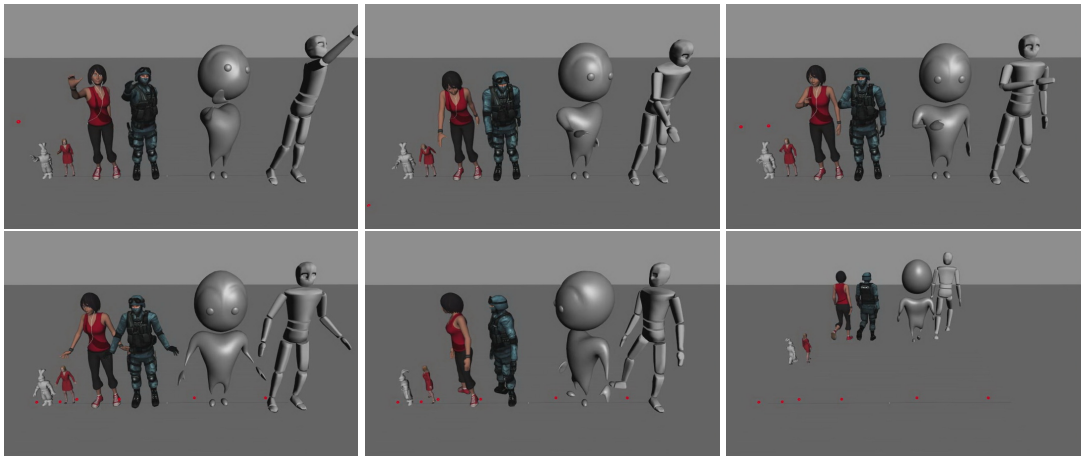


Figure 8.6: More result showing the automatic retargeting on different character models from various sources.

CHAPTER 9

GestureVest - A Portable Motion Capture Solution

9.1 Motivation and Background

One of the key problems for achieving real-time motion modeling platforms is that most of the animation systems are mainly based on pre-defined motion dataset. Building such database required for given scenarios, for example inside a specific training environment, remains a complex and time consuming work. It is usually done by skilled animators specifically for each scenario, and needs to be re-built when switching to a new environment. In order to improve this situation, this chapter describes a wearable low-cost motion capture device *GestureVect*. This vest, along with the framework described in Chapter 5, could capture the advantages of both approaches. A real-time motion blending techniques has been proposed [HK10] in Chapter 3 in order to achieve realistic motion synthesis that can be adapted and parameterized. Also taken into account is the on-line motion demonstrations for the interactive modeling of new gestures and action motions. Together as a framework, it presents a new gesture modeling paradigm based on interactive motion demonstrations. Our framework can be seen as an imitation-based approach [BBG05, SIB03] and is especially relevant for controlling and programming tasks for humanoid agents [SYK08, ONL05, NM03, RM02, BM01]. It also represents a natural approach to human-computer interaction by analogy to the way humans naturally interact with each other. This Chapter describes our results towards achieving an intuitive interface based on low-cost wearable motion sensors for programming and customizing demonstrative gestures and actions. Figure 9.1 shows the user captures a pouring action wearing the vest on our motion modeling platform.

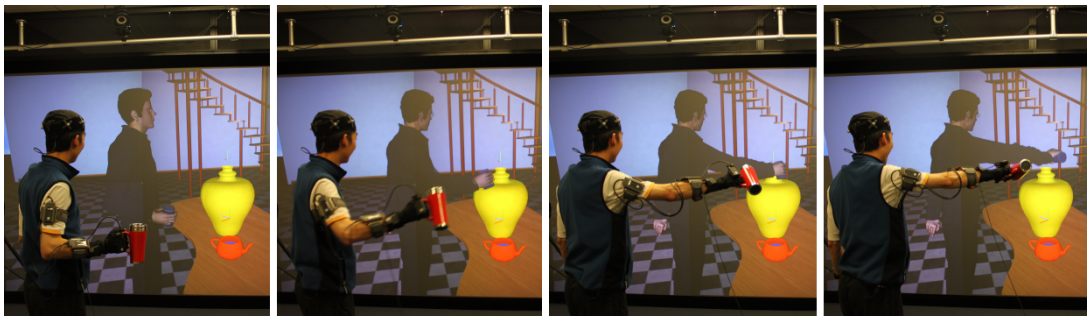


Figure 9.1: This image shows the user captures a pouring action on our motion modeling platform wearing the *GestureVect*.

9.2 System Overview

Our framework models each demonstrative gesture with a cluster of example gestures of the same type but with variations for the different locations being demonstrated. In the pointing gestures modeled in this work the different locations are the locations being pointed at. For example a gesture cluster for a certain way of pointing consists of several examples of similar pointing gestures but each pointing to a different location. Gestures in a cluster are time-aligned in order to allow correct blendings and Inverse Kinematics corrections to be performed for computing a final gesture motion able to achieve given specific target pointing locations. This process is referred here as Inverse Blending.

We have developed for our system a *GestureVest* for achieving a mocap-based human-computer interface for on-line motion demonstrations. Example motions can be demonstrated interactively and converted to new examples to be added in the database, in order to enable the interactive customization of gestures. This interactive interface allows seamless user interactions for programming gestures. A concrete example utilizes the vest is described in Chapter 5 as a motion modeling platform.

The system maintains a gesture database on-the-fly by storing selected demonstrations from the *GestureVest*. The database can also be initialized with basic gestures.

The virtual character can then reuse the demonstrated motions in order to point to any desired location and the resulted motion will exhibit the same characteristics of the demonstrated examples. If suitable candidate example motions are not available in the database, or if the user is not satisfied with the aspect of the output motion, new demonstrations can be interactively given by the user via the vest. The system can therefore learn user-specific (e.g. pointing) tasks in different scenarios through on-line demonstrations, and at the same time preserve the characteristics of the demonstrated motions as much as possible.

9.3 Wearable Motion Capture *GestureVest*

We have built a wearable motion capture vest for achieving a suitable human computer interface for our system. The interface uses five InnaLabs AHRS sensors [Inn08] to capture the orientation changes of performer's spine, head and a full arm. In addition, a 5DT data glove is used to capture hand shapes, and a Nintendo WiiMote controller is used for providing basic instructions (record, play, delete, etc) during demonstration of new example gesture motions. As illustrated in Figure 9.2-right, the sensors are attached on a detachable sleeve, in the form of an easy-to-wear *GestureVest*. The vest is connected to a computer via wired USB-RS 485 converters, optionally connected to a wireless USB hub. For specifications of the miniAHRS sensor, please refer Appendix 11.2.

Each sensor measures its orientation in global coordinates based on tri-axial gyro,

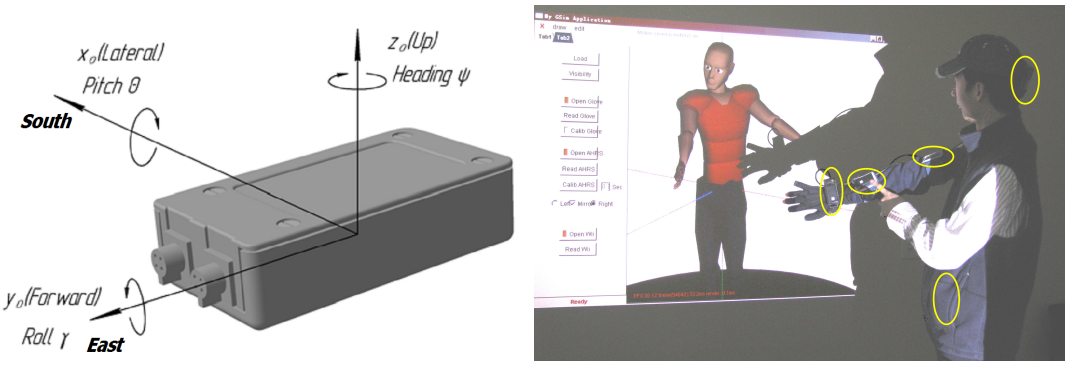


Figure 9.2: Left: miniAHRS m2 sensor used in our system. Right: Our *GestureVest*, with a data glove and 5 miniAHRS sensors placed on spine, head, right upper-arm, right forearm and right hand. The locations are highlighted with yellow rings.

accelerometer and magnetometer. Each measured rotation $q_{reading}$ is represented in quaternion form in respect to sensor's canonical reference frame with X axis pointing South, Y axis pointing East, and Z axis pointing upwards, as shown in Figure 9.2-left. The maximum update rate is 120 Hz in quaternion mode. These commercially-available sensors provide good results but different sensors and technologies can also be used, such as ultrasonic triangulation [VAV07] and accelerometer-based [SH08].

Although it is possible to infer and reconstruct whole-body motions with only a limited number of sensors through models trained on full-range motion capture database [MLC10, SH08], the focus of the vest is to capture high quality gesture motions with enough sensors covering the joints of interest, just like the commercially available systems [Xse07] but more cost-effective.

9.3.1 Calibration

The sensors use magnetometers to acquire absolute orientation based on the earth's magnetic field. We perform an initial orientation calibration by standing in a T-pose facing North to match the zero-rotation of the sensor with the negative X pointing North, as shown in Figure 9.3-(2). Since the sensors are mounted at slightly different positions every time the performer puts on the vest, a calibration is needed before each new capture session. In the calibration process, a reference rotation q_{calib} is record for each sensor.

9.3.2 Skeleton Mapping

In order to map the sensed rotations to our skeleton representing the character to be animated we have to transform the rotations to suitable frames in local coordinates. First we transform $q_{reading}$ to a Z-up coordinate system, producing a rotation compatible with the sensor's native coordinate system (Z-up, see Fig 9.2-left). In this way, for

example, a rotation $q_{reading}$ along South axis will produce the corresponding quaternion rotation q_{z-up} along X axis after applying the calibration quaternion. Rotation q_{z-up} is obtained with:

$$q_{z-up} = q_{reading} \cdot q_{calib}^{-1}$$

The virtual character used follows a Y-up coordinate system as shown in Fig 9.3-(1). In order to drive our character, we use pre-rotations and post-rotations that transform the rotation from miniAHRS Z-up coordinate to our Y-up coordinate:

$$q_{y-up} = q_{preRot} \cdot q_{z-up} \cdot q_{postRot},$$

where q_{preRot} is 120° rotation along axis (-1, 1, 1) in quaternion form that rotates the character from the Y-up frame to the Z-up frame, see Fig 9.3. Rotation $q_{postRot}$ produces the opposite rotation which is -120° rotation along the same axis of (-1, 1, 1).

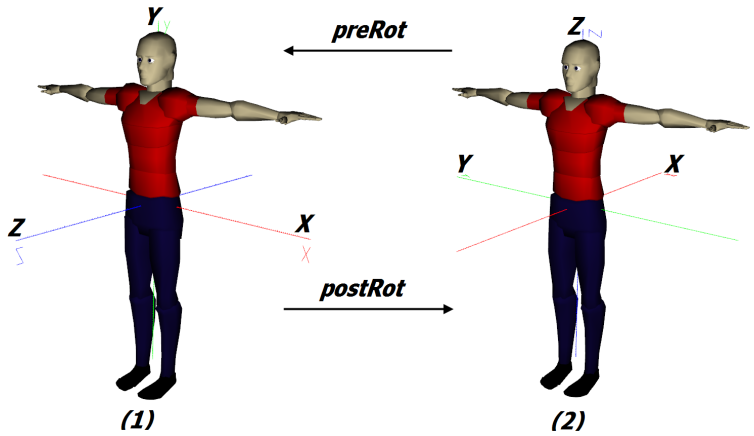


Figure 9.3: This image shows two different coordinate systems: (1) is the default coordinate system used by the AHRS sensor and (2) is the standard Y-up coordinate system used in our setup.

Before applying q_{y-up} to the corresponding joint, it has to be first transformed to the local coordinates frame of that joint. Every time the character skeleton is updated with sensors' readings, the rotation of each joint in global coordinates is computed by Forward Kinematics, i. e. by recursively multiplying the local rotations of all parent joints, starting from the root joint. Global rotations can then be transformed to the local coordinates frame with:

$$q_{local} = q_{parent}^{-1} \cdot q_{y-up},$$

where q_{local} is the final rotation to be sent to the joint being mapped, and q_{parent} is the rotation of the parent joint in global coordinates.

9.3.3 Quality Evaluation

We have compared the results achieved by our *GestureVest* against results recorded with the Vicon optical-based motion capture system, which captures positions in global coordinates with precision generally better than 4 millimeters. Motions were captured simultaneously with the two systems, wearing the vest underneath the Vicon suit with the reflective markers placed following the Vicon system manual.

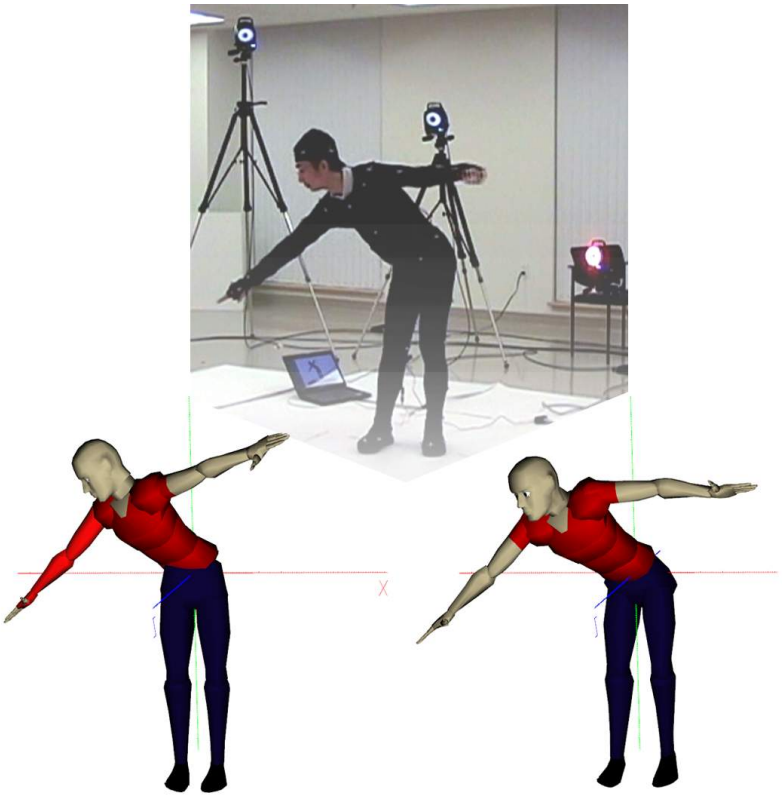


Figure 9.4: Reconstruction quality comparison between our system and Vicon. Top: one captured posture from the analysis. Left: our reconstruction. Right: Vicon's reconstruction.

The sensor readings of the vest were acquired at a rate of 30 frames per second and were directly mapped to our skeleton. Vicon Blade software was used to fit the same skeleton used by our system to the optical markers. Both systems initialized the reconstruction process with identical initial T-poses. The global reference frame of Vicon's reconstruction was also transformed to be the same as in our system, i.e. at the root joint of the character. Since our vest does not capture lower body motions, the performer tried to maintain the waist at the initial position to avoid unnecessary variations of the magnetic field after initial calibration which might affect the reconstruction quality of our system. Although neither reconstruction is 100% precise, Vicon matches the original motions visibly better and is considered the industry standard for motion

capture, thus we treat it as ground truth in our comparison analysis. Evaluation is done by comparing the rotations in-between corresponding joint pairs on both skeletons.

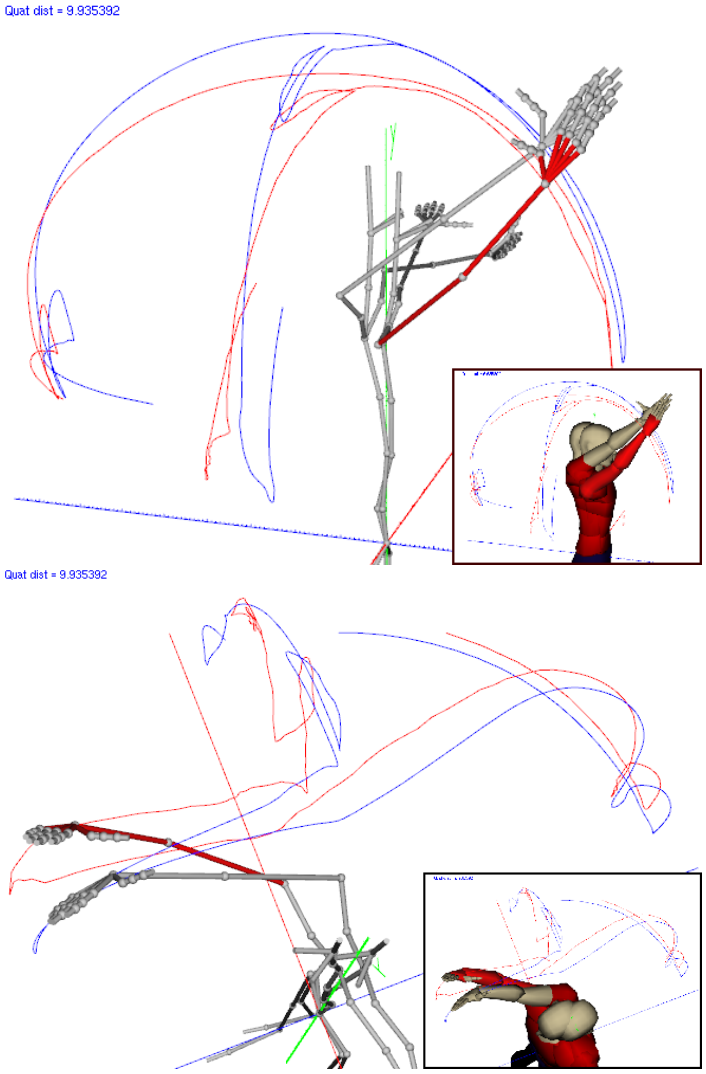


Figure 9.5: Trajectories of the right hand joint with our reconstruction (in red) and Vicon's (in blue) over certain duration where the arm reached extremities towards various directions. The two images are different views of same result, and the small plot within each image shows the same comparison with a skinned character. The entire arm is marked in red on the skeletons/characters with our reconstruction.

Figure 9.4 shows one frame of the reconstruction from the analysis, which consists 85-seconds range-of-motion captured with the right arm covering most of the reachable space of a normal human-being. The reconstruction of the vest is shown on the left (with entire arm marked in red color), and Vicon's result is shown on the right. Figure 9.5 shows the trajectories of the right hand joint traversed over certain period of the captured motion where the arm reached extremities towards various directions, with our reconstruction's trajectory in red and Vicon's in blue. Two plots are showing

different views of the same result. Vicon's reconstruction matches the original motion slightly better as it captures more joints than our vest, though our vest is able to better reconstruct subtle motions such as slight wrist flicks and more precise head orientation as shown in Fig 9.4-left. Note that our vest does not capture the left arm, hence the left arm maintains its initial orientation throughout the reconstruction.

Figure 9.7 Euler angle plots for a set of 3-DOF joints (neck, right shoulder, spine and right hand) over 50 seconds of reconstruction result from both systems. The top three rows in each plot visualize the decomposed Euler angles (along X, Y and Z axis respectively) of each joint measured by the vest (dash red lines) and by Vicon (solid black lines). The bottom row in each plot shows the rotational difference between our reconstruction and that of the Vicon system. The rotational difference is computed using the orientation distance [Han05] between pairs of corresponding joints from the two systems measured in quaternion, defined as $\theta = arccos(q_1 \cdot q_2)$.

As can be seen in Fig 9.6, the rotational differences are generally below $10 \sim 20^{\circ}$. Joint rotations from our mocap system could mostly follow that of Vicon, except for certain periods where the rotational difference slightly increases, particularly when angular acceleration of the motion greatly increases or decreases, and in extreme cases such difference could briefly reach 50°, see Fig9.7. This is due to the characteristics of the miniAHRS sensor that the orientation sensing relies more on gyroscope and accelerometer to achieve fast responses when angular acceleration increases, and depends more on magnetometer when movement becomes slower for fairly accurate measurement.

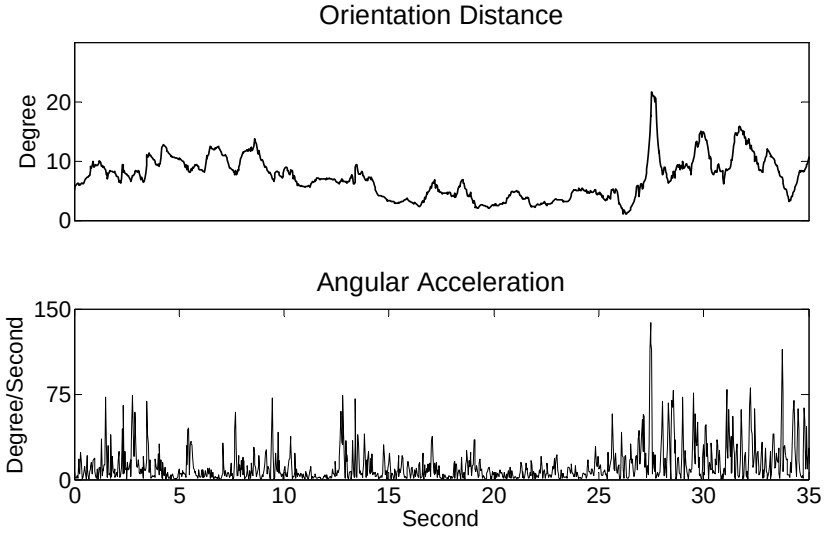


Figure 9.6: Top: orientation distance between corresponding one pair of shoulder joints from the two systems. The plot covers over 35 seconds of reconstruction. Botom: the angular acceleration of the orientation distance shown above, which shows that the orientation distance increases with the angular acceleration.

It is worth noting that two years after the material in this chapter was published, Cockcroft presented an in-depth evaluation and analysis of the inertial motion capture technology in his thesis work [Coc11].

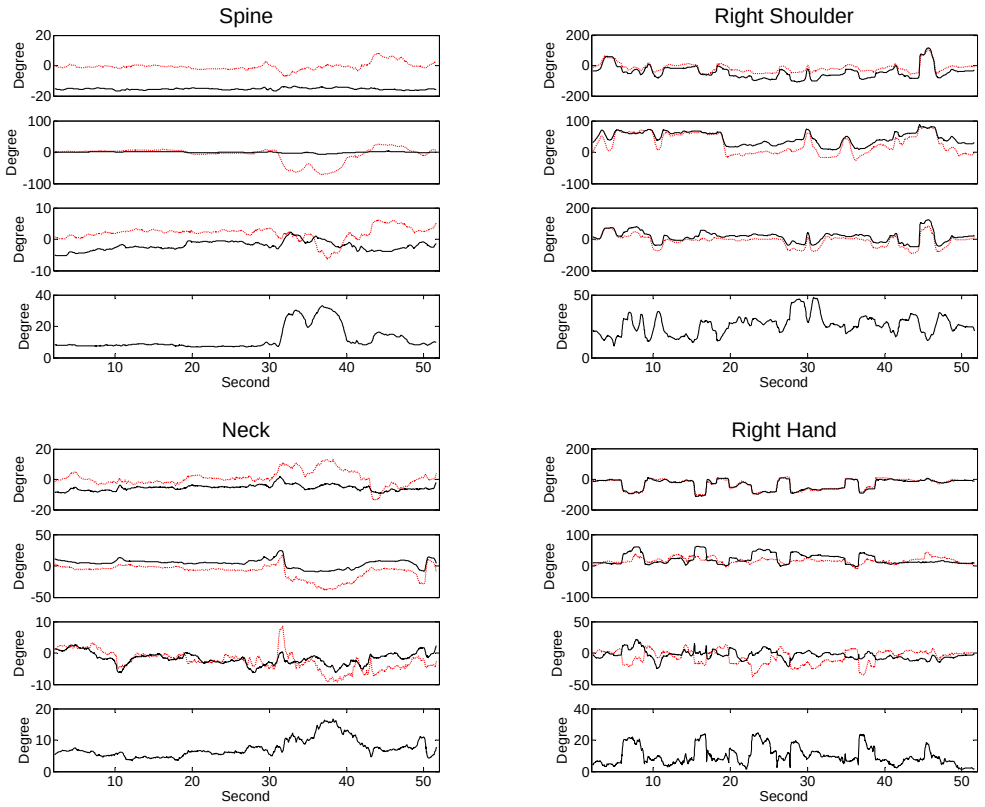


Figure 9.7: Euler angle plots for a set of 3-DOF joints (spine, shoulder, hand and neck) over 50 seconds of reconstruction result from both systems. The top three rows in each plot illustrate the Euler angles (along X, Y and Z axes) reconstructed by the vest (dash red lines) and by Vicon (solid black lines). The bottom row in each plot shows the orientation difference between the two reconstructions from the joint pair. Due to the nature of the miniAHRS sensor, such difference may briefly reach 50° in extreme cases.

9.4 Discussion

Our *GestureVest* system is specifically built for on-line motion acquisition with fairly good results. Our single-arm configuration for capturing single arm gestures with 5 sensors provides enough flexibility for on-line motion demonstration of demonstrative gestures, where example motions will be demonstrated interactively and converted to new gesture examples to populate clusters in the database. After a few demonstrations are performed, object targets to point to can be given in any scenario. Whenever a new

target in the scene is selected, the appropriate cluster is identified and new motions are synthesized by the inverse blending algorithm. The mocap system is robust, easy to wear, intuitive to use, and also relatively cheap in cost.

The motion stream produced by the sensors is continuously monitored during performances and annotated by simple commands given by the user through the WiiMote controller (Fig 9.8) such as: create new example motion, refine existing motion, replace current candidate motion and delete from database. These simple commands achieve a seamless and intuitive human-computer interface for informing how to interactively segment, classify and cluster the input motion into meaningful gesture segments.

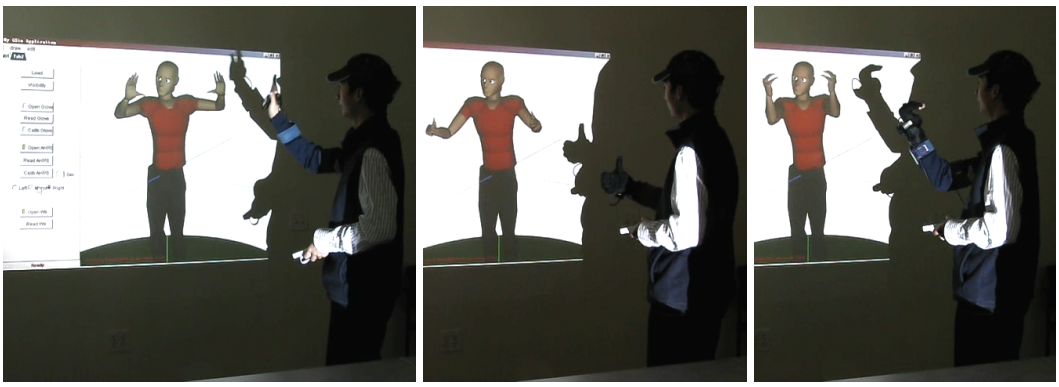


Figure 9.8: The mirroring feature is enabled and the left arm is mirroring the movement of the right arm which is captured with 3 miniAHRS sensors. This frees up the left hand to hold a WiiMote as system controller. Two images on the right show that the vest is being used with a data glove capturing the finger movements. This minimal setup is ideal for capturing gestures and simple upper-body actions.

The system automatically segments and organizes the collected gestures into gesture database to be used by the Inverse Blending. For our current experiments with pointing gestures, segmentation is implemented in a straightforward way simply by observing the zero crossing of the velocity vector of the hand. This works well as pointings often have a unique maximal stroke point before returning to a rest posture. Sliding-window filters are applied to eliminate false detections so that the segmentation is immune to jitters and noises in the captured motions. Parameters of the filters are fine tuned by hand. Auto time-alignment of the segments is applied before storing the new segments in the database.

We have also implemented mirroring of gestures from the captured arm to the other arm. This enables the capturing of symmetric full-torso gestures as in several two-handed gestures used in speeches, conversations and in sign languages [XQM02]. The mirroring is done by negating both the x and w components of the original quaternion. Also a data glove can be added to capture the movement of the fingers. The vest presents an ideal solution for low-cost motion capture systems that target at gestures and simple upper-body action modeling and so on. These two features are shown in Figure 9.8.

9.5 Conclusion

We have presented a low-cost wearable motion capture solution that designed for real-time immersive motion modeling systems. This setup has been integrated with our motion modeling platform described in Chapter 5. Figure 9.1 as an example shows the user interactively teach virtual agents in generic training scenarios all in real-time with the portable vest solution.

The vest has also been used for several of our experimental research recordings. Figure 9.9 shows a scenario where pointing motions from human subjects were collected in one of our pioneering studies [HHK11a]. It was easy to use and calibrate, not affected by problems like occlusion, all made it perfect to handle a large group of human subjects.

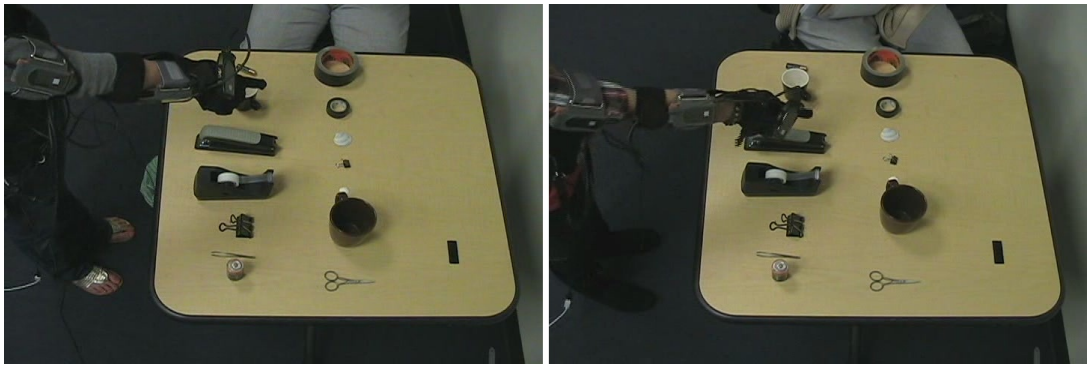


Figure 9.9: The vest was used in an experiment on pointing gestures, where the setup does not allow optical motion capture solutions due to occlusions.

CHAPTER 10

Final Conclusions

Throughout this dissertation we have developed whole-body motion planning approaches for the real-time synthesis of coordinated behaviors specifically for demonstrative settings. In this final chapter we will discuss some of the key points of our work with concluding remarks.

The main goal of this dissertation is to achieve an autonomous character animation system that combines realism, flexibility, precise control and ability to adapt to the cluttered environments. Additionally, the system is connected to higher-level constraints with task-oriented specifications, and produces real-time synthesis within interactive applications like virtual demonstrator system.

10.1 Summary of Contributions

The approach we have taken can be largely summarized as a combination of three parts: (1) motion capture based motion parametrization, (2) randomized human-like motion planning in blending spaces, and (3) coordinated behavior modeling in demonstrative scenarios.

10.1.1 Motion Parametrization

We have addressed the motion synthesis problem with an example-based motion parameterization algorithm *Inverse Blending* specifically designed to achieve satisfying generic spatial constraints at interactive frame rates. The proposed approach is based on blending of a set of consistent time-aligned example motion sequences. The example motions are carefully built for the consistency. They could be collected immersively inside the environment. To achieve precise motion control, blending weights are directly optimized until the specified constraints are best met. This process in general takes a few milliseconds depending on the dataset and the complexity of the character's joint hierarchy.

Inverse Blending has been analyzed and evaluated against three existing motion blending techniques, and the result reflects what we have claimed as contributions. The analysis shows that, while our approach might be less smooth than RBF interpolation only when the high-level constraints are continuously changed, it excels in accuracy with practically no pre-computation time. In situations where constraints are only en-

forced at certain key times, the synthesis results are smooth and closely resemble the original example motions, which is considered humanlike.

Our approach has been integrated on an immersive motion modeling platform. The user easily built training motion dataset via direct demonstration inside the virtual environment. New motions were interactively generated for teaching and demonstrative purposes in real-time as an effective way to facilitate the virtual training applications.

10.1.2 Planning in Blending Spaces

We have investigated in the topic of motion planning in an attempt to plan collision-free natural-looking action motions inside cluttered environment. Humanlikeness is hard to achieve with traditional sampling-based approaches inside the configuration space. For this reason, we have introduced Blending Space as a novel search space for human-like motions that in a sense integrates *Inverse Blending* with randomized planning. What we have achieved is a bi-directional time-synchronized sampling-based planner capable of exploring the weight-time blending space of example motions. The randomized planning allows the final synthesis to have the flexibility going around obstacles, while sampling inside the weight-time blending space gives the benefit of maintaining the original quality of the example motions. The planner is coupled with a motion graph based locomotion planner, and this bi-modal system could plan for realistic whole-body sequences inside cluttered environments.

While the randomized sampling generates natural-looking results, it only explores inside the blending space representing a subset of the full configuration space. Depending on the size of example motions in the dataset, it may take longer time to find solutions in a cluttered environment. In such cases, the locomotion planner continues to expand towards additional body placement candidates until the action can be planned and executed, or returns failure until a maximum time limit is reached, a trade-off between humanlikeness and solvability.

10.1.3 Coordinated Behavior Modeling

We have also proposed a whole-body planner that addresses the high-level coordinations specifically for demonstrative tasks. The planner breaks down the overall planning problem into body positioning, locomotion, action execution and gaze synthesis, each solved with a sub-module. The motion coordinations have been addressed in the overall synthesis. Human participants were invited to perform pointing tasks, and the data collected was used to fine tune the parameters of several sub-modules. As a result, an autonomous virtual agent planner is achieved that can engage and demonstrate towards observers in various scenarios.

Due to the nature of the problem modeled, the proposed planner aims to simulate only one demonstrative agent towards one observer at a given time. Certain visibility constraints such as observer-target and agent-target are not addressed in this model.

10.2 Future Research

The whole-body coordinated planning for demonstrative virtual agents remains a complex and challenging problem. The motion capture-based motion planning and synthesis methods presented in this dissertation demonstrated the ability in solving several aspects of the problem modeled, however this only covers part of the various possible demonstrative settings, and there still is room for improvement in the following directions.

10.2.1 Motion Synthesis with Dynamics

While fully dynamic control of an articulated character still remains a very difficult problem, the benefit of introducing dynamic models can already be seen in many of the recent works. A typical solution is to track motion capture data with physics for added responsiveness to exterior disturbance. Similar approaches have been used to help determine the best plausible re-entry into motion library play-back following unexpected impacts.

The integration of dynamic models into proposed motion parametrization approach would open up many possibilities. For example, if the character is carrying a heavy object, the added impact on the body posture could be easily handled by the dynamics, without the need to capture a set of example motions specifically for carrying heavy objects. Similar cases include collision among the body, the object with obstacles, or the object has been dropped or picked up, enabling the simulation of various scenarios with limited motion capture examples.

10.2.2 Making the Planning Faster

Longer planning time is the bottle-neck of randomized planning algorithms. As future work, to maximize the possibility of solution finding, the algorithm could adapt to the environment by automatically switching to the best sampling strategy. Another improvement is that the search trees of motion clips could be pre-computed for arbitrary environment and saved for later reuse, avoid planning from scratch each time. Other learning mechanisms could also be introduced to potentially speed up the planning process.

10.2.3 Expanding the Behavior Models and Evaluations

What we have presented is a very specific behavior model for simulating demonstrative agents in a dyadic setting, i.e. one demonstrator with one observer. There are many possibilities as future work. For example, the body positioning module could be expanded to allow the demonstrator performing toward multiple observers, or to demonstrate in front of a vertical screen such as a white board or a projection screen.

In essence, our model helps the demonstrator gain more attention from the observers. However, the reversal of such model could also be used where the agent tries to avoid attentions, for example finding the least attention-grabbing way to enter an on-going presentation without disturbing the audience and the speaker. Additional models could be integrated to address other social activities and interactions among multiple agents, such as conversations and seat selections in a public place.

Due to the importance and difficulty of evaluating the quality of the result, we also plan to evaluate and analysis the overall planning results by comparing simulated synthesis against real captures. Environment mimicking the simulated planning problems would be set up and new captures would be collected from human participants in the form of either motion capture or videotaping. Our synthesis results would be presented side by side with the new captures in the form of an online survey and rated on the similarities between our synthesis and the ground truth.

CHAPTER 11

Appendix

11.1 Body-positioning Model Fitting Parameters and Goodness of Fit

Details of fitting parameters and goodness of fit for the body-positioning model introduced in Chapter 7 Section 7.5. Polynomial (cubic and quadratic) functions were chosen over other types of fitting functions such as Gaussian and Fourier, for the ability to extrapolate at both ends ($\alpha > 150$ and $\alpha < -150$), and also for the lower computational cost at run-time.

$\beta = f(\alpha) = p_1\alpha^3 + p_2\alpha^2 + p_3\alpha + p_4$
Coefficients (with 95% confidence bounds)
$p_1 = 2.392e^{-006}(-6.74e^{-006}, 1.152e^{-005})$
$p_2 = 0.0003056(-0.0004444, 0.001056)$
$p_3 = 0.1145(-0.04067, 0.2697)$
$p_4 = -6.062(-15.42, 3.294)$
Goodness of fit:
$SSE = 5386$
$R^2 = 0.6156$
$AdjustedR^2 = 0.5713$
$RMSE = 14.39$

$\theta = f(\alpha) = p_1\alpha^2 + p_2\alpha + p_3$
Coefficients (with 95% confidence bounds)
$p_1 = 0.0006228(0.000262, 0.0009837)$
$p_2 = 0.3267(0.2991, 0.3542)$
$p_3 = 11.29(6.564, 16.02)$
Goodness of fit:
$SSE = 1441$
$R^2 = 0.9635$
$AdjustedR^2 = 0.9608$
$RMSE = 7.304$

$\phi = f(\alpha) = p_1\alpha^2 + p_2\alpha + p_3$
Coefficients (with 95% confidence bounds):
$p1 = 0.0006673(0.0001145, 0.00122)$

$p2 = 0.6736(0.6315, 0.7158)$
$p3 = 2.073(-5.167, 9.312)$
Goodness of fit:
SSE : 3381
R^2 : 0.9785
$AdjustedR^2$: 0.9769
$RMSE$: 11.19

11.2 Specifications of miniAHRS sensor in *GestureVest*

This table lists the specifications of the miniAHRS tri-axial orientation sensor used in our *GestureVest* described in Chapter 9.

SPECIFICATIONS

Parameter	Unit	Value
Update Rate	Hz	1.....100 (user settable)
Start-up Time	sec	< 1
Full Accuracy Data (Warm-up Time)	sec	< 60
Heading		
Range	deg	0 to 360
Static Accuracy at Normal Conditions (Tc=25°C)	deg RMS	0.25 (1)
Static Accuracy in Temperature Range (Tc=-40°C to +70°C)	deg RMS	0.4 (1)
Dynamic Accuracy	deg RMS	0.7 (2)
Noise (at 100Hz output)	deg RMS	0.03
Resolution	deg	0.01
Attitude		
Range: Pitch, Roll	deg	±90, ±180
Static Accuracy at Normal Conditions (Tc=25°C)	deg RMS	0.04
Static Accuracy in Temperature Range (Tc=-40°C to +70°C)	deg RMS	0.1
Dynamic Accuracy	deg RMS	0.4 (2)
Noise (at 100Hz output)	deg RMS	0.02
Resolution	deg	0.01
Angular Rate		
Input Range: Yaw, Pitch, Roll	deg/sec	±300
Bias	deg/sec RMS	0.02
Scale Factor Accuracy	%	0.2
Non-Linearity	% FS	0.1
Random Walk	deg/sqrt(hr)	6
Resolution	deg/sec	0.01
Bandwidth	Hz	40
Linear acceleration		
Input Range: X/Y/Z	g	±2
Bias	mg RMS	0.3
Scale Factor Accuracy	%	< 0.1
Non-Linearity	% FS	0.2
Random Walk	m/s/sqrt(hour)	0.06
Resolution	mg	< 10
Bandwidth	Hz	40
Environment		
Operating Temperature	deg C	–40 to +70
Non-Operating Temperature	deg C	–55 to +85
Electrical		
Input Voltage	VDC	+5.5 to +6.5
Input Current	mA	< 110
Power Consumption	W	0.66 (at 6V powering)
Digital Output Format		RS-232
Physical		
Size	mm	109 x 31 x 29 (case) 127 x 31 x 29 (with mounting lugs and connector)
Weight	kg	0.19 / 0.16 (3)
Connector		5-Pin Binder 719 (Male)

(1) root mean square error (1 sigma) in homogeneous magnetic environment, for latitude up to ±65 deg;
(2) may depend on type of motion;
(3) depends on material of the AHRS case.

REFERENCES

[AF02] Okan Arikan and David A. Forsyth. "Synthesizing Constrained Motions from Examples." *Proceedings of SIGGRAPH*, **21**(3):483–490, 2002.

[AFO03] Okan Arikan, David A. Forsyth, and James F. O'Brien. "Motion synthesis from annotations." *ACM Transaction on Graphics (Proceedings of SIGGRAPH)*, **22**(3):402–408, 2003.

[AI11] Okan Arikan and Leslie Ikemoto. "Animeeple character animation tool.", 2011.

[ALP04] Yeuhi Abe, C. Karen Liu, and Zoran Popović. "Momentum-based parameterization of dynamic character motion." In *SCA '04: 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 173–182, Aire-la-Ville, Switzerland, 2004. Eurographics Association.

[AN99] Y. Aydin and M. Nakajima. "Database Guided Computer Animation of Human Grasping using Forward and Inverse Kinematics." *Computer & Graphics*, **23**:145–154, 1999.

[Ani97] Animazoo. "Gypsy electromechanical motion capture systems.", 1997.

[Arb81] M. A. Arbib. "Perceptual Structures and Distributed Motor Control." In V. B. Brooks, editor, *Handbook of Physiology, Section 2: The Nervous System Vol. II, Motor Control, Part 1*, pp. 1449–1480. American Physiological Society, Bethesda, MD, 1981.

[Asc91] Ascension Technology Corporation. "Flock of Birds motion trackers.", 1991.

[AST08] Edilson de Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. "Performance capture from sparse multi-view video." *ACM Trans. Graph.*, **27**(3):98:1–98:10, August 2008.

[Aut] Autodesk, Inc. "MotionBuilder real-time 3D character animation software.".

[BBG05] C. Breazeal, D. Buchsbaum, J. Gray, D. Gatenby, and D. Blumberg. "Learning from and about Others: Towards Using Imitation to Bootstrap the Social Understanding of Others by Robots." *Artificial Life*, **11**:1–2, 2005.

[BH00] Matthew Brand and Aaron Hertzmann. "Style machines." In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pp. 183–192, 2000.

[BKD06] Dominik Bertram, James Kuffner, Ruediger Dillmann, and Tamim Asfour. "An Integrated Approach to Inverse Kinematics and Path Planning for Redundant Manipulators." In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1874–1879. IEEE, May 2006.

[BLC02] Christoph Bregler, Lorie Loeb, Erika Chuang, and Hrishi Deshpande. "Turning to the masters: motion capturing cartoons." In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pp. 399–407, 2002.

[BM01] A Billard and M. J. Matarić. "Learning human arm movements by imitation: Evaluation of a biologically inspired connectionist architecture." *Robotics and Autonomous Systems*, **37:2-3**:145–160, November, 30 2001.

[Bos05] Boston Dynamics. "BigDog, a dynamically stable quadruped robot.", 2005.

[Bre06] Timothy Bretl. "Motion Planning of Multi-Limbed Robots Subject to Equilibrium Constraints: The Free-Climbing Robot Problem." *International Journal of Robotics Research*, **25**(4):317–342, 2006.

[BW95] Armin Bruderlin and Lance Williams. "Motion signal processing." In *SIGGRAPH '95*, pp. 97–104, New York, NY, USA, 1995. ACM.

[BWA10] Nikolaus Bee, Johannes Wagner, Elisabeth André, Thurid Vogt, Fred Charles, David Pizzi, and Marc Cavazza. "Discovering eye gaze behavior during human-agent conversation in an interactive storytelling application." In *Int'l Conference on Multimodal Interfaces and Workshop on Machine Learning for Multimodal Interaction*, ICMI-MLMI '10, pp. 9:1–9:8, New York, NY, USA, 2010. ACM.

[CBY08] Stelian Coros, Philippe Beaudoin, Kang Kang Yin, and Michiel van de Pann. "Synthesis of constrained walking skills." *ACM Trans. Graph.*, **27**(5):1–9, 2008.

[CCL10] Benjamin J. Cohen, Sachin Chitta, and Maxim Likhachev. "Search-based Planning for Manipulation with Motion Primitives." In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 2902–2908, 2010.

[CGV11] Elin Carstensdottir, Kristin Gudmundsdottir, Gunnar Valgardsson, and Hannes Vilhjalmsson. "Where to sit? the study and implementation of seat selection in public places." In *Proceedings of the 10th international conference on Intelligent virtual agents*, IVA'11, pp. 48–54, 2011.

[CHB04] Kathleen E. Cullen, Marko Huterer, Danielle A. Braidwood, and Pierre A. Sylvestre. "Time Course of Vestibuloocular Reflex Suppression During Gaze Shifts." *Journal of Neurophysiology*, **92**(6):3408–3422, 2004.

[CHK10] Carlo Camporesi, Yazhou Huang, and Marcelo Kallmann. "Interactive Motion Modeling and Parameterization by Direct Demonstration." In *Proceedings of the 10th International Conference on Intelligent Virtual Agents (IVA)*, 2010.

[CHP07] Seth Cooper, Aaron Hertzmann, and Zoran Popović. "Active learning for real-time motion controllers." *ACM Trans. Graph.*, **26**(3), July 2007.

[CK99] Kwang jin Choi and Hyeong seok Ko. "On-line Motion Retargetting." *Journal of Visualization and Computer Animation*, **11**:223–235, 1999.

[CL03] Min Gyu Choi and Jehee Lee. "Planning biped locomotion using motion capture data and probabilistic roadmaps." *ACM Transactions on Graphics*, **22**:182–203, 2003.

[CLC05] Joel Chestnutt, Manfred Lau, Kong Man Cheung, James Kuffner, Jessica K Hodgins, and Takeo Kanade. "Footstep Planning for the Honda ASIMO Humanoid." In *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2005.

[CNK07] J. Chestnutt, K. Nishiwaki, J.J. Kuffner, and S. Kagami. "An adaptive action model for legged navigation planning." In *Proceedings of the IEEE/RAS International Conference on Humanoid Robotics*, 2007.

[Coc11] Stephen John Cockcroft. *An evaluation of inertial motion capture technology for use in the analysis and optimization of road cycling kinematics*. PhD thesis, University of Stellenbosch, 2011. http://hdl.handle.net/10019.1/6760.

[CVB01] Justine Cassell, Hannes Högni Vilhjálmsson, and Timothy Bickmore. "BEAT: the Behavior Expression Animation Toolkit." In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pp. 477–486, New York, NY, USA, 2001. ACM.

[Dil98] Robert Dilts. *Modeling With NLP (Neuro-Linguistic Programming)*. Meta Publications, Capitola, CA, 1998.

[DK08] Rosen Diankov and James Kuffner. "randomized statistical path planning." In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 1–6, May 19-23 2008.

[DKM04] Evan Drumwright, Marcelo Kallmann, and Maja Matarić. "Towards Single-Arm Reaching for Humanoid Robots in Dynamic Environments." In *Proceedings of the IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*, Santa Monica, CA, November 2004.

[DLN05] Zhigang Deng, J.P. Lewis, and U. Neumann. "Automated eye motion using texture synthesis." *Computer Graphics and Applications, IEEE*, **25**(2):24 – 30, 2005.

[DN06] Evan Drumwright and V. Ng-Thow-Hing. "Toward Interactive Reaching in Static Environments for Humanoid Robots." In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, October 2006.

[EAP06a] Claudia Esteves, Gustavo Arechavaleta, Julien Pettré, and Jean-Paul Laumond. "Animation planning for virtual characters cooperation." *ACM Transaction on Graphics*, **25**(2):319–339, 2006.

[EAP06b] Claudia Esteves, Gustavo Arechavaleta, Julien Pettré, and Jean-Paul Laumond. "Animation planning for virtual characters cooperation." *ACM Trans. Graph.*, **25**(2):319–339, April 2006.

[EMM04] Arjan Egges, Tom Molet, and Nadia Magnenat-Thalmann. "Personalised Real-Time Idle Motion Synthesis." In *Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, PG '04, pp. 121–130, 2004.

[EMO10] Cathy Ennis, Rachel McDonnell, and Carol O'Sullivan. "Seeing is believing: body motion dominates in multisensory conversations." *ACM Trans. Graph.*, **29**:91:1–91:9, July 2010.

[EON09] EON Reality, Inc. "Virtual Reality Training System (VRTS).", 2009.

[For98] Forterra Systems, Inc. "On-Line Interactive Virtual Environment (OLIVE).", 1998.

[FXS12] Andrew W. Feng, Yuyu Xu, and Ari Shapiro. "An example-based motion synthesis technique for locomotion and object manipulation." In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '12, pp. 95–102, 2012.

[GBT06] Pascal Glardon, Ronan Boulic, and Daniel Thalmann. "Robust on-line adaptive footplant detection and enforcement for locomotion." *Vis. Comput.*, **22**(3):194–209, March 2006.

[GG92] H. L. Galiana and D. Guitton. "Central Organization and Modeling of Eye-Head Coordination during Orienting Gaze Shifts." *Annals of the New York Acd. of Sci.*, **656**(1):452–471, 1992.

[GKK03] Patrick Gebhard, Michael Kipp, Martin Klesen, and Thomas Rist. "What Are They Going to Talk About? Towards Life-Like Characters that Reflect on Interactions with Users." In *Proc. of the 1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment (TIDSE'03)*, 2003.

[Gle98] Michael Gleicher. "Retargetting motion to new characters." In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pp. 33–42, New York, NY, USA, 1998. ACM.

[Gle08] Michael Gleicher. "More Motion Capture in Games – Can We Make Example-Based Approaches Scale?" In Arjan Egges, Arno Kamphuis, and Mark Overmars, editors, *Proceeding of the International Conference on Motion in Games*, pp. 82–93. Springer-Verlag, Berlin, Heidelberg, 2008.

[GMH04] Keith Grochow, Steven Martin, Aaron Hertzmann, and Zoran Popović. "Style-Based Inverse Kinematics." *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, **23**(3):522–531, 2004.

[Goe03] Jennifer Goetz. "Matching robot appearance and behavior to tasks to improve human-robot cooperation." In *Proceeding of the Workshop on Robot and Human Interactive Communication*, pp. 55–60, 2003.

[GSK03] Michael Gleicher, Hyun Joon Shin, Lucas Kovar, and Andrew Jepsen. "Snap-together motion: assembling run-time animations." In *Proceedings of the symposium on Interactive 3D graphics (I3D)*, pp. 181–188, New York, NY, USA, 2003. ACM Press.

[Han05] Andrew J. Hanson. *Visualizing Quaternions (The Morgan Kaufmann Series in Interactive 3D Technology)*. Morgan Kaufmann, December 2005.

[HBH06] K. Hauser, T. Bretl, K. Harada, and J.C. Latombe. "Using Motion Primitives in Probabilistic Sample-Based Planning for Humanoid Robots." In *Workshop on Algorithmic Foundations of Robotics (WAFR)*, pp. 2641–2648, July 2006.

[HBH08] Kris Hauser, Timothy Bretl, Kensuke Harada, and Jean-Claude Latombe. "Using Motion Primitives in Probabilistic Sample-Based Planning for Humanoid Robots." In *Algorithmic Foundation of Robotics VII*, volume 47 of *Springer Tracts in Advanced Robotics*, pp. 507–522. 2008.

[HBL05] K. Hauser, T. Bretl, and J.C. Latombe. "Non-Gaited Humanoid Locomotion Planning." In *Humanoids*, pp. 2641– 2648, December 2005.

[HCK12] Yazhou Huang, Carlo Camporesi, and Marcelo Kallmann. "Immersive Interfaces for Building Parameterized Motion Databases." In *Proceedings of the 11th International Conference on Intelligent Virtual Agents (IVA)*, pp. 474–476. Springer Berlin / Heidelberg, 2012.

[HG07] Rachel Heck and Michael Gleicher. "Parametric motion graphs." In *Proceedings of the symposium on Interactive 3D graphics and games (I3D)*, pp. 129–136, New York, NY, USA, 2007. ACM Press.

[HHK11a] Stephanie Huette, Yazhou Huang, Marcelo Kallmann, Teenie Matlock, and Justin Matthews. "Gesture Variants and Cognitive Constraints for Interactive Virtual Reality Training Systems." In *International Conference on Intelligent User Interfaces (IUI)*, pp. 351–354, 2011.

[HHK11b] Stephanie Huette, Yazhou Huang, Marcelo Kallmann, Teenie Matlock, and Justin L. Matthews. "Gesture variants and cognitive constraints for interactive virtual reality training systems." In *Proceeding of 16th International Conference on Intelligent User Interfaces (IUI)*, pp. 351–354, 2011.

[HK09a] Alexis Heloir and Michael Kipp. "EMBR - A Realtime Animation Engine for Interactive Embodied Agents." In *In Proceedings of the 9th International Conference on Intelligent Virtual Agents (IVA)*, volume 5773 of *Lecture Notes in Computer Science*, pp. 393–404. Springer Berlin / Heidelberg, 2009.

[HK09b] Yazhou Huang and Marcelo Kallmann. "Interactive Demonstration of Pointing Gestures for Virtual Trainers." In *Proceedings of 13th International Conference on Human-Computer Interaction*, San Diego, CA, 2009.

[HK10] Yazhou Huang and Marcelo Kallmann. "Motion Parameterization with Inverse Blending." In *Proceedings of the Third International Conference on Motion In Games*, Berlin, 2010. Springer.

[HKG06] Rachel Heck, Lucas Kovar, and Michael Gleicher. "Splicing Upper-Body Actions with Locomotion." In *Proceedings of Eurographics 2006*, september 2006.

[HKM11] Yazhou Huang, Marcelo Kallmann, Justin L Matthews, and Teenie Matlock. "Modeling Gaze Behavior for Virtual Demonstrators." In *Proceedings of the 11th International Conference on Intelligent Virtual Agents (IVA)*, 2011.

[HKT10] Edmond S. L. Ho, Taku Komura, and Chiew-Lan Tai. "Spatial relationship preserving character motion adaptation." *ACM Trans. Graph.*, **29**(4):33:1–33:8, July 2010.

[HL09] Kris Hauser and Jean-Claude Latombe. "Multi-modal Motion Planning in Non-expansive Spaces." *The International Journal of Robotics Research*, 2009.

[HMK11] Yazhou Huang, Mentar Mahmudi, and Marcelo Kallmann. "Planning Humanlike Actions in Blending Spaces." In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[HN10] Kris Hauser and Victor Ng-Thow-Hing. "Randomized multi-modal motion planning for a humanoid robot manipulation task." *The International Journal of Robotics Research*, 2010.

[HNG07] K. K. Hauser, V. Ng-Thowhing, Gonzalez-Baos, H.Tomohiko Mukai, and Shigeru Kuriyama. "Multi-Modal Motion Planning for a Humanoid Robot Manipulation Task." In *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2007.

[HNS04] Hartwig Holzapfel, Kai Nickel, and Rainer Stiefelhagen. "Implementation and evaluation of a constraint-based multimodal fusion system for speech and 3D pointing gestures." In *Proceedings of the 6th international conference on Multimodal interfaces*, ICMI '04, pp. 175–182, New York, NY, USA, 2004. ACM.

[Hon00] Honda Motor Co., Inc. "ASIMO, Advanced Step in Innovative Mobility.", 2000.

[HRE08] Chris Hecker, Bernd Raabe, Ryan W. Enslow, John DeWeese, Jordan Maynard, and Kees van Prooijen. "Real-time motion retargeting to highly varied user-created morphologies." In *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, pp. 27:1–27:11, 2008.

[IAF09] Leslie Ikemoto, Okan Arikan, and David Forsyth. "Generalizing motion edits with Gaussian processes." *ACM Trans. Graph.*, **28**(1):1:1–1:12, February 2009.

[Inn08] InnaLabs. "InnaLabs miniAHRS m2 User's Manual.", 2008.

[JK07] Xiaoxi Jiang and Marcelo Kallmann. "Learning Humanoid Reaching Tasks in Dynamic Environments." In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, San Diego CA, 2007.

[Joh09] Rune Skovbo Johansen. *"Automated Semi-Procedural Animation for Character Locomotion."*. Master's thesis, Aarhus University, the Netherlands, 2009.

[JYL09] Sumit Jain, Yuting Ye, and C. Karen Liu. "Optimization-based interactive motion synthesis." *ACM Trans. Graph.*, **28**(1):10:1–10:12, February 2009.

[KAA03] Marcelo Kallmann, Amaury Aubel, Tolga Abaci, and Daniel Thalmann. "Planning Collision-Free Reaching Motions for Interactive Object Manipulation and Grasping." *Computer graphics Forum (Proceedings of Eurographics'03)*, **22**(3):313–322, September 2003.

[Kal08] Marcelo Kallmann. "Analytical Inverse Kinematics with Body Posture Control." *Computer Animation and Virtual Worlds*, **19**(2):79–91, 2008.

[Kal10] Marcelo Kallmann. "Shortest Paths with Arbitrary Clearance from Navigation Meshes." In *Proceedings of the Eurographics / SIGGRAPH Symposium on Computer Animation (SCA)*, 2010.

[Ken76] Adam Kendon. “Social Perception and Steering for Online Avatars.” In *Man Environment Systems*, volume 6, pp. 291–296, 1976.

[Ken90a] A. Kendon. *Conducting Interaction: Patterns of Behavior in Focused Encounters*. Studies in Interactional Sociolinguistics. Cambridge University Press, 1990.

[Ken90b] A. Kendon. “Some Functions of Gaze Direction in Two-Person Conversation.” *Conducting Interaction: Patterns of Behavior in Focused Encounters*, 1990.

[KG04] Lucas Kovar and Michael Gleicher. “Automated extraction and parameterization of motions in large data sets.” *ACM Transaction on Graphics (Proceedings of SIGGRAPH)*, **23**(3):559–568, 2004.

[KGP02] Lucas Kovar, Michael Gleicher, and Frederic H. Pighin. “Motion graphs.” *Proceedings of SIGGRAPH*, **21**(3):473–482, 2002.

[KHB10] Marcelo Kallmann, Yazhou Huang, and Robert Backman. “A Skill-Based Motion Planning Framework for Humanoids.” In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2010.

[KHK75] Adam. Kendon, Richard M. Harris, and Mary R. Key. *Organization of Behavior in Face-To-Face Interaction*. World Anthropology. Mouton, 1975.

[KHK09] Manmyung Kim, Kyunglyul Hyun, Jongmin Kim, and Jehee Lee. “Synchronized multi-character motion editing.” *ACM Trans. Graph.*, **28**(3):79:1–79:9, July 2009.

[KKK94] Yoshihito Koga, Koichi Kondo, James J. Kuffner, and Jean-Claude Latombe. “Planning Motions with Intentions.” In *Proceedings of SIGGRAPH*, pp. 395–408. ACM Press, 1994.

[KKM06] Stefan Kopp, Brigitte Krenn, Stacy Marsella, Andrew Marshall, Catherine Pelachaud, Hannes Pirker, Kristinn Thórisson, and Hannes Vilhjálmsson. “Towards a Common Framework for Multimodal Generation: The Behavior Markup Language.” In Jonathan Gratch, Michael Young, Ruth Aylett, Daniel Ballin, and Patrick Olivier, editors, *Proceedings of the International Conference on Intelligent Virtual Agents (IVA)*, volume 4133 of *Lecture Notes in Computer Science*, pp. 205–217. Springer Berlin / Heidelberg, 2006.

[KKN03] S. Kagami, J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue. “Humanoid arm motion planning using stereo vision and rrt search.” *Journal of Robotics and Mechatronics*, April 2003.

[KL00a] James J. Kuffner and Jean-Claude Latombe. "Interactive manipulation planning for animated characters." In *Proceedings of Pacific Graphics*, Hong Kong, October 2000. poster paper.

[KL00b] James J. Kuffner and Steven M. LaValle. "RRT-Connect: An Efficient Approach to Single-Query Path Planning." In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, CA, April 2000.

[KMA05] R. Kulpa, F. Multon, and B. Arnaldi. "Morphology-independent representation of motions for interactive human-like animation." *Computer Graphics Forum, Eurographics 2005 special issue*, **24**:343–352, 2005.

[KNK03] James Kuffner, Koichi Nishiwaki, Satoshi Kagami, Masayuki Inaba, and Hirochika Inoue. "Motion Planning for Humanoid Robots." In *Proceedings of the 11th International Symposium of Robotics Research (ISRR)*, November 2003.

[KNK07] Michael Kipp, Michael Neff, Kerstin H. Kipp, and Irene Albrecht. "Towards Natural Gesture Synthesis: Evaluating Gesture Units in a Data-Driven Approach to Gesture Synthesis." In *Proceedings of the 7th international conference on Intelligent Virtual Agents*, IVA '07, pp. 15–28, 2007.

[KS05] Taesoo Kwon and Sung Yong Shin. "Motion modeling for on-line locomotion synthesis." In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA)*, pp. 29–38, New York, NY, USA, 2005. ACM Press.

[KSG02] Lucas Kovar, John Schreiner, and Michael Gleicher. "Footskate cleanup for motion capture editing." In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA)*, pp. 97–104, New York, NY, USA, 2002. ACM Press.

[KSL96] Lydia Kavraki, Petr Svestka, Jean-Claude Latombe, and Mark Overmars. "Probabilistic Roadmaps for Fast Path Planning in High-Dimensional Configuration Spaces." *IEEE Transactions on Robotics and Automation*, **12**:566–580, 1996.

[KW04] Stefan Kopp and Ipke Wachsmuth. "Synthesizing multimodal utterances for conversational agents: Research Articles." *Computer Animation and Virtual Worlds*, **15**(1):39–52, 2004.

[Lat90] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publisher, December 1990.

[Lau98] Jean-Paul P. Laumond. *Robot Motion Planning and Control*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[LaV98] Steven LaValle. “Rapidly-Exploring Random Trees: A New Tool for Path Planning.” Technical Report 98-11, Iowa State University, Computer Science Department, October 1998.

[LaV06] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press (available on-line), 2006.

[Law03] Neil Lawrence. “Gaussian Process Latent Variable Models for Visualization of High Dimensional Data.” In *In Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, p. 2004, 2003.

[LBR92] Philippe Lefevre, Ignace Bottemanne, and Andre Roucoux. “Experimental study and modeling of vestibulo-ocular reflex modulation during large shifts of gaze in humans.” *Experimental Brain Research*, **91**:496–508, 1992.

[LCR02] Jehee Lee, Jinxiang Chai, Paul Reitsma, Jessica K Hodgins, and Nancy Pollard. “Interactive Control of Avatars Animated with Human Motion Data.” *Proceedings of SIGGRAPH*, **21**(3):491–500, July 2002.

[Lin03] Linden Research, Inc. “Second Life.”, 2003.

[LK05] Manfred Lau and James J. Kuffner. “Behavior planning for character animation.” In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA ’05, pp. 271–280, 2005.

[LK06] Manfred Lau and James J. Kuffner. “Precomputed search trees: planning for interactive goal-driven animation.” In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA)*, pp. 299–308, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[LKT10] Sergey Levine, Philipp Krähenbühl, Sebastian Thrun, and Vladlen Koltun. “Gesture controllers.” *ACM Trans. Graph.*, **29**(4):124:1–124:11, July 2010.

[LLK11] Sergey Levine, Yongjoon Lee, Vladlen Koltun, and Zoran Popović. “Space-time planning with parameterized locomotion controllers.” *ACM Trans. Graph.*, **30**(3):23:1–23:11, May 2011.

[LM07] Brent Lance and Stacy Marsella. “Emotionally Expressive Head and Body Movements During Gaze Shifts.” In *7th Int’l Conference on Intelligent Virtual Agemts (IVA)*, pp. 72–85, 2007.

[LS99] Jehee Lee and Sung Yong Shin. “A hierarchical approach to interactive motion editing for human-like figures.” In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH

'99, pp. 39–48, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[LS01] Jehee Lee and Sung Yong Shin. "A coordinate-invariant approach to multiresolution motion analysis." *Graph. Models*, **63**(2):87–105, March 2001.

[LWH12] Sergey Levine, Jack M. Wang, Alexis Haraux, Zoran Popović, and Vladlen Koltun. "Continuous character control with low-dimensional embeddings." *ACM Trans. Graph.*, **31**(4), July 2012.

[LWS02] Yan Li, Tian-Shu Wang, and Heung-Yeung Shum. "Motion texture: a two-level statistical model for character motion synthesis." *Proceedings of SIGGRAPH*, **21**(3):465–472, 2002.

[MAF11] C. Miller, O. Arikan, and D. Fussell. "Frankenrigs: Building Character Rigs from Multiple Sources." *Visualization and Computer Graphics, IEEE Transactions on*, **17**(8):1060 –1070, aug. 2011.

[MBB00] Jean-Sébastien Monzani, Paolo Baerlocher, Ronan Boulic, and Daniel Thalmann. "Using an Intermediate Skeleton and Inverse Kinematics for Motion Retargeting." *Computer Graphics Forum*, **19**(3), 2000.

[MDT09] Hunter A. Murphy, Andrew T. Duchowski, and Richard A. Tyrrell. "Hybrid image/model-based gaze-contingent rendering." *ACM Trans. Appl. Percept.*, **5**:22:1–22:21, February 2009.

[MHF06] Bilge Mutlu, Jessica K Hodgins, and Jodi Forlizzi. "A Storytelling Robot: Modeling and Evaluation of Human-like Gaze Behavior." In *Proceedings of HUMANOIDS'06, 2006 IEEE-RAS International Conference on Humanoid Robots*. IEEE, December 2006.

[Mic10] Microsoft. "Kinect motion tracking device for games.", 2010.

[MK05] Tomohiko Mukai and Shigeru Kuriyama. "Geostatistical motion interpolation." In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pp. 1062–1070, New York, NY, USA, 2005. ACM.

[MK11] Mentar Mahmudi and Marcelo Kallmann. "Feature-Based Locomotion with Inverse Branch Kinematics." In *Proceedings of the 4th International Conference on Motion In Games (MIG)*, 2011.

[MLC10] Jianyuan Min, Huajun Liu, and Jinxiang Chai. "Synthesis and editing of personalized stylistic human motion." In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, I3D '10, pp. 39–46, New York, NY, USA, 2010. ACM.

[MMK12] M. Mori, K.F. MacDorman, and N. Kageki. "The Uncanny Valley [From the Field]." *Robotics Automation Magazine, IEEE*, **19**(2):98–100, june 2012.

[Moo65] Gordon E. Moore. "Cramming more components onto integrated circuits." *Electronics*, **38**(8), April 1965.

[Mor70] Masahiro Mori. "The Uncanny Valley (originally in Japanese: Bukimi no tani)." *Energy*, **7**(4):33–35, 1970.

[Mot98] Motek Medical BV. "V-Gait.", 1998.

[MPP11] Uldarico Muico, Jovan Popović, and Zoran Popović. "Composite control of physically simulated characters." *ACM Trans. Graph.*, **30**(3), May 2011.

[MRC05] Meinard Müller, Tido Röder, and Michael Clausen. "Efficient content-based retrieval of motion capture data." In *Proceedings of SIGGRAPH*, pp. 677–685, New York, NY, USA, 2005. ACM Press.

[MZS09] Adriano Macchietto, Victor Zordan, and Christian R. Shelton. "Momentum control for balance." *ACM Trans. Graph.*, **28**(3):80:1–80:8, July 2009.

[Nat05] NaturalMotion Ltd. "Dynamic Motion Synthesis (White Paper).", 2005.

[NBM09] Radoslaw Niewiadomski, Elisabetta Bevacqua, Maurizio Mancini, and Catherine Pelachaud. "Greta: an interactive expressive ECA system." In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '09, pp. 1399–1400, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems.

[NF02] Michael Neff and Eugene Fiume. "Modeling tension and relaxation for computer animation." In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '02, pp. 81–88, 2002.

[NI10] Yukiko I. Nakano and Ryo Ishii. "Estimating user's engagement from eye-gaze behaviors in human-agent conversations." In *Proceedings of the 15th international conference on Intelligent user interfaces*, IUI '10, pp. 139–148, 2010.

[NKA08] Michael Neff, Michael Kipp, Irene Albrecht, and Hans-Peter Seidel. "Gesture modeling and animation based on a probabilistic re-creation of speaker style." *ACM Trans. Graph.*, **27**(1):5:1–5:24, March 2008.

[NM03] M. N. Nicolescu and M. J. Matarić. "Natural Methods for Robots Task Learning: Instructive Demonstration, Generalization and Practice." In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Melbourne, Australia, 2003.

[ONL05] A. Olenderski, M. Nicolescu, and S Louis. "Robot Learning by Demonstration Using Forward Models of Schema-Based Behaviors." In *Proceedings, International Conference on Informatics in Control, Automation and Robotics*, pp. 14–17, Barcelona, Spain, September 2005.

[Org] Organic Motion. "OpenStage marker-less motion capture system.".

[PB02] Katherine Pullen and Christoph Bregler. "Motion Capture Assisted Animation: Texturing and Synthesis." *Proceedings of SIGGRAPH*, pp. 501–508, 2002.

[PK81] Alan Pease and Jacqueline Kent. *Body Language: How to Read Others' Thoughts by Their Gestures*. Camel Pub., 1981.

[PKL08] Julien Pettré, Marcelo Kallmann, and Ming C. Lin. "Motion planning and autonomy for virtual humans." In *ACM SIGGRAPH 2008 classes*, SIGGRAPH '08, pp. 42:1–42:31, 2008.

[PL06] Julien Pettre and Jean-Paul Laumond. "A motion capture-based control-space approach for walking mannequins." *Computer Animation Virtual Worlds*, **17**(2):109–126, 2006.

[POO09] Julien Pettré, Jan Ondřej, Anne-Hélène Olivier, Armel Cretual, and Stéphane Donikian. "Experiment-based modeling, simulation and validation of interactions between virtual walkers." In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, pp. 189–198, New York, NY, USA, 2009. ACM.

[PPU88] Denis Pelisson, Claude Prablanc, and Christian Urquizar. "Vestibuloocular reflex inhibition and gaze saccade control characteristics during eye-head orientation in humans." *Journal of Neurophysiology*, **59**:997–1013, 1988.

[PSS02] Sang Il Park, Hyun Joon Shin, and Sung Yong Shin. "On-line locomotion generation based on motion blending." In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA)*, pp. 105–111, New York, NY, USA, 2002. ACM Press.

[PTV07] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge Univ. Press, New York, NY, USA, 2007.

[PV08] Claudio Pedica and Hannes Vilhjálmsson. "Social Perception and Steering for Online Avatars." In *Proceedings of the 8th international conference on Intelligent Virtual Agents*, IVA '08, pp. 104–116, 2008.

[PV09] Claudio Pedica and Hannes Högni Vilhjálmsson. "Spontaneous Avatar Behavior for Human Territoriality." In *Proceedings of the 9th International Conference on Intelligent Virtual Agents*, IVA '09, pp. 344–357, 2009.

[PZL10] Jia Pan, Liangjun Zhang, Ming C. Lin, and Dinesh Manocha. "A hybrid approach for simulating human motion in constrained environments." *Comput. Animat. Virtual Worlds*, **21**(34):137–149, May 2010.

[RBC98] Charles Rose, Bobby Bodenheimer, and Michael F. Cohen. "Verbs and adverbs: Multidimensional motion interpolation." *IEEE Computer Graphics and Applications*, **18**:32–40, 1998.

[RM02] A. Ramesh and M. J. Matarić. "Learning Movement Sequences from Demonstration." In *Proceedings of the International Conference on Development and Learning (ICDL)*, pp. 302–306, MIT, Cambridge, MA, 2002.

[RPB03] I. Rodriguez, M. Peinado, R. Boulic, and D. Meziat. "Bringing the human arm reachable space to a virtual environment for its analysis." In *IEEE International Conference on Multimedia and Expo*, 2003.

[RSC01] Charles F. RoseIII, Peter-Pike J. Sloan, and Michael F. Cohen. "Artist-Directed Inverse-Kinematics Using Radial Basis Function Interpolation." *Computer Graphics Forum (Proceedings of Eurographics)*, **20**(3):239–250, September 2001.

[SA76] Albert E. Scheflen and Norman Ashcraft. *Human territories: how we behave in space-time*. Prentice-Hall, Englewood Cliffs, NJ, USA, December 1976.

[Saf06] Alla Safonova. *Reducing the search space for physically realistic human motion synthesis*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, September 2006.

[SDO04] Matthew Stone, Doug DeCarlo, Insuk Oh, Christian Rodriguez, Adrian Stere, Alyssa Lees, and Chris Bregler. "Speaking with hands: creating animated conversational characters from recordings of human performance." *ACM Transactions on Graphics*, **23**(3):506–513, 2004.

[SH07] Alla Safonova and Jessica K. Hodgins. "Construction and optimal search of interpolated motion graphs." *ACM Transactions on Graphics (SIGGRAPH 2007)*, **26**(3), August 2007.

[SH08] Ronit Slyper and Jessica Hodgins. "Action Capture with Accelerometers." In *2008 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, July 2008.

[Sha11] Ari Shapiro. "Building a Character Animation System." In *Proceedings of the Fourth International Conference on Motion In Games*, Berlin, 2011. Springer.

[SHP04] Alla Safonova, Jessica K. Hodgins, and Nancy S. Pollard. "Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces." In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pp. 514–521, 2004.

[SIB03] S. Schaal, A. Ijspeert, and A. Billard. "Computational approaches to motor learning by imitation." *The Neuroscience of Social Interaction*, **1431**:199–218, 2003.

[Sie80] Siemens. "product lifecycle management (PLM).", 1980.

[SKF07] Ari Shapiro, Marcelo Kallmann, and Petros Faloutsos. "Interactive Motion Correction and Object Manipulation." In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*, Seattle, 2007.

[SLS01] Hyun Joon Shin, Jehee Lee, Sung Yong Shin, and Michael Gleicher. "Computer puppetry: An importance-based approach." *ACM Trans. Graph.*, **20**(2):67–94, April 2001.

[SM01] Harold C. Sun and Dimitris N. Metaxas. "Automating gait generation." In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pp. 261–270, 2001.

[SM04] R. William Soukoreff and I. Scott MacKenzie. "Towards a standard for pointing device evaluation: Perspectives on 27 years of Fitts' law research in HCI." *International Journal of Human-Computer Studies*, **61**:751–789, 2004.

[SPF03] Ari Shapiro, Fred Pighin, and Petros Faloutsos. "Hybrid Control for Interactive Character Animation." In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, PG '03, 2003.

[ST05] Wei Shao and Demetri Terzopoulos. "Autonomous pedestrians." In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '05, pp. 19–28, New York, NY, USA, 2005. ACM.

[SYK08] Wael Suleiman, Eiichi Yoshida, Fumio Kanehiro, Jean-Paul Laumond, and André Monin. "On Human Motion imitation by Humanoid Robot." In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2697–2704, May 19-23 2008.

[SZG05] Robert W. Sumner, Matthias Zwicker, Craig Gotsman, and Jovan Popović. "Mesh-based inverse kinematics." *ACM Trans. Graph.*, **24**(3):488–495, 2005.

[TLM09] Marcus Thiebaux, Brent Lance, and Stacy Marsella. "Real-time expressive gaze animation for virtual humans." In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 321–328, 2009.

[TLP07] A. Treuille, Y. Lee, and Z. Popović. "Near-optimal Character Animation with Continuous Control." In *Proceedings of ACM SIGGRAPH*. ACM Press, 2007.

[TMK99] Daniel Thalmann, Soraia Musse, and Marcelo Kallmann. "Virtual Humans' Behaviour: Individuals, Groups, and Crowds." In *Proceedings of the International Conference on Digital Media Futures*, Bradford, UK, April, 13-15 1999.

[TMM08] Marcus Thiebaux, Stacy Marsella, Andrew N. Marshall, and Marcelo Kallmann. "SmartBody: behavior realization for embodied conversational agents." In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, AAMAS '08, pp. 151–158, 2008.

[TS00] K. A. Thoroughman and R. Shadmehr. "Learning of action through combination of motor primitives." *Nature*, **407**:742–747, 2000.

[UAT95] Munetoshi Unuma, Ken Anjyo, and Ryozo Takeuchi. "Fourier principles for emotion-based human figure animation." In *SIGGRAPH '95*, pp. 91–96, New York, NY, USA, 1995. ACM.

[Uni12] Unity3D cross-platform game engine. "version 4.0 pre-release new feature demo.", June 2012.

[USC12] USC Institute for Creative Technologies (ICT). "SmartBody animation framework.", 2012.

[VAV07] Daniel Vlasic, Rolf Adelsberger, Giovanni Vannucci, John Barnwell, Markus Gross, Wojciech Matusik, and Jovan Popović. "Practical motion capture in everyday surroundings." *ACM Trans. Graph.*, **26**(3):35, 2007.

[Vic95] Vicon, formerly known as Oxford-Metrics, Ltd. "Vicon 370e optical tracking system.", 1995.

[VSC08] Marion R. Van Horn, Pierre A. Sylvestre, and Kathleen E. Cullen. "The Brain Stem Saccadic Burst Generator Encodes Gaze in Three-Dimensional Space." *J. of Neurophysiology*, **99**(5):2602–2616, 2008.

[WB04] Jing Wang and Bobby Bodenheimer. "Computing the duration of motion transitions: an empirical approach." In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '04, pp. 335–344. Eurographics Association, 2004.

[WH97] D. J. Wiley and J. K. Hahn. "Interpolation Synthesis of Articulated Figure Motion." *IEEE Computer Graphics and Applications*, **17**(6):39–45, 1997.

[WRR10] Herwin van Welbergen, Dennis Reidsma, Zsófia M. Ruttkay, and Job Zwiers. "Elckerlyc: A BML Realizer for continuous, multimodal interaction with a virtual human." *Journal on Multimodal User Interfaces*, **3**(4):271–284, 2010.

[XQM02] Yingen Xiong, Francis Quek, and David McNeill. "Hand Gesture Symmetric Behavior Detection and Analysis in Natural Conversation." *Multimodal Interfaces, IEEE International Conference on*, **0**:179, 2002.

[Xse07] Xsens Technologies. "Moven inertial motion capture system.", 2007.

[YKH04] Katsu Yamane, James J. Kuffner, and Jessica K. Hodgins. "Synthesizing Animations of Human Manipulation Tasks." *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, **23**(3):532–539, 2004.

[YLP07] KangKang Yin, Kevin Loken, and Michiel van de Panne. "SIMBICON: Simple Biped Locomotion Control." In *Proceedings of ACM SIGGRAPH*, San Diego, CA, 2007.

[YSA08] Jingzhou Yang, Tariq Sinokrot, Karim Abdel-Malek, Steve Beck, and Kyle Nebel. "Workspace zone differentiation and visualization for virtual humans." *Ergonomics*, **51**(3):395–413, 2008.

[ZH02] Victor Brian Zordan and Jessica K. Hodgins. "Motion capture-driven simulations that hit and react." In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '02, pp. 89–96, 2002.

[ZHH10] Franziska Zacharias, Ian S. Howard, Thomas Hulin, and Gerd Hirzinger. "Workspace comparisons of setup configurations for human-robot interaction." In *IROS*, pp. 3117–3122, 2010.

[ZLM09] Liangjun Zhang, Steven M. LaValle, and Dinesh Manocha. "Global vector field computation for feedback motion planning." In *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, pp. 3065–3070, 2009.

[ZMC05] Victor Brian Zordan, Anna Majkowska, Bill Chiu, and Matthew Fast. "Dynamic response for motion capture animation." In *Proceedings of ACM SIGGRAPH*, pp. 697–701, New York, NY, USA, 2005. ACM Press.

[ZNK09] Liming Zhao, Aline Normoyle, Sanjeev Khanna, and Alla Safonova. "Automatic construction of a minimum size motion graph." In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, pp. 27–35, 2009.

[ZV03] Victor Brian Zordan and Nicholas C. Van Der Horst. "Mapping optical motion capture data to skeletal motion using a physical model." In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03, pp. 245–250, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.