DATA-DRIVEN DERIVATION OF SKILLS FOR AUTONOMOUS HUMANOID AGENTS

by

Odest Chadwicke Jenkins

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(COMPUTER SCIENCE)

December 2003

Acknowledgements

My experiences leading to the completion of this dissertation have been extremely rewarding, but also a tremendous amount of work. Along the way, I have benefited from the guidance, support, and camaraderie of many individuals, whom I now acknowledge and thank.

I would like to first thank my parents, Odest Charles Jenkins and Dr. Nadine Francis Jenkins, and family for raising me to be a conscientious individual and for being a continual source of encouragement in my endeavors. Thanks for being patient with me. It is difficult for me to imagine how I could have come this far without the help of my wife Sarah. She has been an invaluable companion and partner who has supported and tolerated me throughout my time as a graduate student, especially near deadlines. Thanks for buffering me from the rigors of daily life. Love you.

I consider myself fortunate for having had Professor Maja Matarić serve as my advisor and collaborator over the past four years. I thank Maja for providing me with direction and inspiration in my research while remaining open to my ideas and encouraging me to speak my mind. I have appreciated Maja setting a high bar in her expectations of me, even when they are the source of our "discussions".

I am grateful to the members of my defense and proposal committees for their time, consideration, and constructive feedback towards developing my dissertation. Specifically, I thank Professor Stefan Schaal for being an excellent teacher and exposing me to the benefits of machine learning, Professor Carolee Winstein for helping me relate my work to fields outside of robotics and computer science, Professor Ulrich Neumann for insights into evaluating my methodology, and Professor William Swartout for valuable comments on improving the presentation of my work and its relationship to computer animation.

My time at USC would not have been the same without the various friends (and accomplices) that I have met along the way. While a plethora of great work is produced in our robotics lab, it would not be possible without the underappreciated and less glamorous (if that is possible) work of our administrative staff, Kusum Shori, Angela Megert, and Aimee Barnard. I particularly thank Angie for sharing a common vision and the frustrations that come with trying to improve the status quo.

Thanks to a selfless friend, Monica Nicolescu, who has helped me throughout my time in the lab and shared in the work and frustrations of graduate student life. I appreciate that Monica has always been available for a spontaneous venting session. Thanks to Brian "Beef" Gerkey for all his help over the years as our de facto sys admin, for engaging conversations about the state of the world (keep fighting The Man, brother), and for giving me a reason to brave the west side. Thanks to Andrew Howard for being a continual source of valued criticism, a role model for me as a scientist, and a really smart dude. Thanks to a great friend and my favorite sounding board for ideas, Gabriel Brostow, whose enthusiasm and drive has been infectious. Thanks to one of the most prepared people that I know, Dani Goldberg, for showing me how to be a successful

Ph.D. student while maintaining a sense of fun. I am grateful to have collaborated with excellent humanoid robotics researchers in our lab, including: Stefan Weber whose free-spirited energy helped establish the fun atmosphere of the lab, Ajo Fod who was the other original inhabitant of the Beefy Lounge, Chi-Wei (Wayne) Chu who has braved the inner depths of my programming and survived, Evan Drumwright whose sense of humor is like multiple pieces of genius. Thanks also to fellow humanoids Amit Ramesh and Marcelo Kallmann.

I have benefited from my interactions with the faculty and postdocs in the USC Robotics Labs including Professor Gaurav Sukhatme, Richard Vaughan, Paolo Pirjanian, Ash Tews, and Torby Dahl. The Robotics Lab has been a fun environment to work in because of people such as Jacob Fredslund, Kasper Stoy, Esben Ostergaard, the ultimate hoop warrior Chris Vernon Jones, Jens Wawerla, Helen Yan, the always dapper, witty, and "rispiktfil" Dylan "Lord Flexington" Shell, Nate Koenig, Kale Harbick, and Gabe Sibley.

I also thank Doug Fidaleo and the people of CGIT who showed me why an RV is not a good idea for going to conferences, Mircea Nicolescu for making soccer enjoyable, Alex Francois for introducing me to rugby, Kaz Okada for invaluable pointers to various dimension reduction techniques, and Didi Yao and other my colleagues in the CSGO.

I am grateful to Jessica Hodgins for giving me my start as a robotics researcher in the Animation Lab while at Georgia Tech and providing motion data that were critical in evaluating my work. Thanks to the Mathematics and Computer Science Department at Alma College for the personal attention and interest in my academic development as an undergraduate and for giving me the confidence to pursue graduate studies. While

at Alma, I was fortunate to have met my friend James Blum, whose constant drive for innovation and finding solutions has been an example for my direction as a computer scientist and a leader.

Big ups to Roger the Sealion.

The research reported in this dissertation was conducted at the Robotics Research Laboratory in the Computer Science Department at the University of Southern California and supported in part by the USC All-University Predoctoral Fellowship, DARPA MARS Program grants DABT63-99-1-0015 and NAG9-1444 and ONR MURI grants N00014-01-1-0354 and SA3319.

Contents

List Of Tables

List Of Figures

Abstract

This dissertation addresses the problem of modularizing the capabilities of a *humanoid agent* into *skill level behaviors*. Our approach to this problem is to derive the skill level behaviors in a data-driven fashion from human demonstration. The humanoid agent is provided with a repertoire of basic skills by leveraging underlying behaviors in observed human movement. These skills serve as a foundation for endowing a humanoid agent with the ability to act *autonomously*. Given such a repertoire, a humanoid agent can autonomously perform functions such as control for various tasks, classification of human motion, and learning by imitation. Additionally, a repertoire of skills provides a common vocabulary for human-agent interaction and interface for non-technical users.

We developed *Performance-Derived Behavior Vocabularies (PDBV)*, an automated data-driven methodology for deriving a vocabulary of skill level behaviors from human motion data. PDBV assumes as input an unlabeled kinematic time-series of joint angle values, acquired from human performance demonstrative of multiple activities. We present *spatio-temporal Isomap* as an unsupervised dimension reduction technique for uncovering underlying spatio-temporal structure in kinematic motion. Using spatio-temporal Isomap, demonstrated motion data are clustered into groups of *exemplars*, where each group contains exemplars of an underlying primitive behavior. Exemplars

of a behaviors can be generalized and realized as *forward models* that encode the nonlinear dynamics of the underlying behaviors in the joint angle space of the agent. Skills as forward models can be used in a variety of functions, including control and perception.

We validate and evaluate the above approach to automated skill derivation in several ways. First, the methodology is empirically evaluated on multiple sources of time-series data, spanning scripted activities such as dancing and athletics, in order to validate input motion preprocessing, the structure of derived behavior vocabularies, realizing each behavior as forward models, and humanoid agent control. Second, we analyze and discuss our approach for deriving capabilities with respect to our empirical results. Lastly, we illustrate the utility of derived behavior vocabularies for use in movement imitation, addressing two subproblems: humanoid motion synthesis and human movement classification.

Chapter 1

Introduction

Robots are emerging as a ubiquitous and crucial component in our society. Robotic technologies are now used in a variety of contexts, including manufacturing, entertainment, education, medical domains, space exploration, and emergency response. These technologies have much promise for other applications such as assistants for the elderly, training/rehabilitation, homeland security, and environmental monitoring/cleanup. In addition to *physically embodied* robotic systems, *artificially embodied* robots, situated in virtual environments with simulated physics, are increasingly used for computer animation, interactive entertainment, and digital special effects. Both physically and virtually embodied robots fall under the category of *robotic agents*.

This dissertation focuses on *autonomous humanoid agents*, a subset of robotic agents with the embodiment characteristics of a human and the ability to act without external supervision. Examples of physically embodied and artificially embodied humanoid robotic agents are shown in Figure 1.1. Many variations of humanoid agents have been successfully designed and implemented. Endowing such agents with the ability to act

autonomously remains an open problem, due to the complex challenges of *autonomous control*, *external interaction*, and *task specification*.

Challenges in control of robotic agents:

- Autonomous control: How do we endow a humanoid agent with autonomous control?
- External interaction: How do we communicate our intentions to a humanoid agent?
- Task structure: How do we represent tasks in a manner amenable to autonomous humanoid control?

One of the promising approaches to dealing with the complex problems outlined above is through the use of *primitives* [93, 120]. These modules provide a substrate or repertoire of skills for the agent that are meant to replace otherwise intractable on-line trajectory planning. Using an example from [120] , the number of possible actions for a 30-*degree-of-freedom (DOF)* humanoid agent with 3 possible values for each DOF is $30^3 > 10^{14}$. This exponential explosion in the space of actions limits the applicability of search-based trajectory planning techniques for the selection of humanoid actions. By representing the capabilities of a humanoid as primitives, the space of possible actions for the agent can be modularized into a more compact and accessible interface. Furthermore, such modularization enables a variety of approaches to autonomous control to be employed, including reactive, behavior-based, and hybrid (for a complete survey of relevant architectures see [92]). Finally, primitives have been employed as a foundation for learning by demonstration, a technique for humanoid control that is growing in popularity [120, 93].

Even though modular skills can serve as a useful action *repertoire*, it remains unclear what constitutes a set of primitive modules suitable for autonomous humanoid agents.

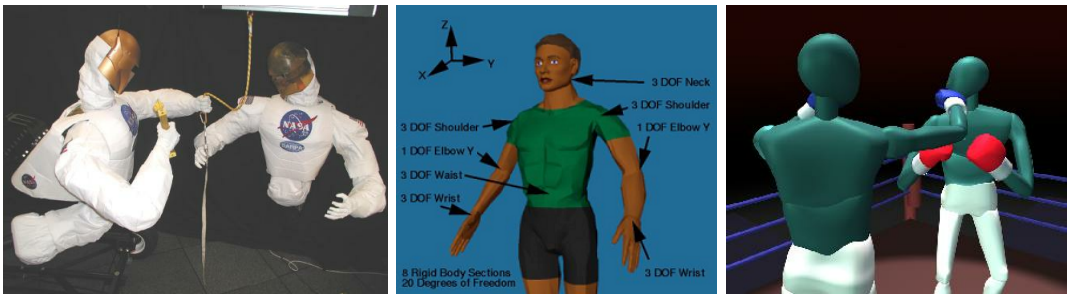


Figure 1.1: *Humanoid agents can be physically embodied, as with the NASA Robonauts [1] (left), or virtually embodied in physical simulations, as with Adonis [91] (center) and Zordan's boxing simulation [148] (right).*

Such a repertoire could be *model-based*, created according to models of human motor control or domain-specific behaviors (e.g., dancing, athletics). In contrast, this dissertation addresses the problem of skill acquisition in a *data-driven model-free* fashion, through *learning from demonstration*. We propose a methodology to automatically derive a repertoire of skill-level behaviors for a humanoid agent from motion demonstrated by a human. More specifically, our methodology is aimed at extracting behaviors that underlie unlabeled time-series data of kinematic motion consisting of unsegmented performances of multiple activities. By deriving skills in this manner, we leverage the inherent human capabilities for structuring and modularizing motion, and attempt to uncover that structure automatically and apply it toward humanoid agent control. By using a data-driven approach, we seek to remove bias from the modularization by avoiding as many explicit assumptions as possible.

Dissertation goal: Development of a methodology for endowing a humanoid agent with a modular repertoire of skills automatically derivated from kinematic human motion.

1.1 Aims and Motivation

In order to realize the goal of this dissertation, we must address two fundamental problems regarding modularization and learning.

Problems addressed in this dissertation:

- Capability modularization: how to modularize or subdivide the functionality of a system into a repertoire of basic capabilities (e.g., behaviors or procedures)?
- Spatio-temporal clustering: how to uncover underlying structure in spatio-temporal data (data whose points have both spatial and temporal relationships)?

The ability to modularize the functionality of a system is a problem encountered throughout the field of Computer Science. This problem of capability modularization is an aspect of system construction that aims to define and implement the basic procedures used by a system. Programmers and researchers typically must select and then implement a set of basic and modular procedures (or functions) that can be invoked by other programs for various purposes. For example, a module for sorting data could be used in one program to alphabetize a list of names, another program to search more efficiently for relevant information, and still another program to help construct an indexing structure for a database. Thus, such modular routines serve as a foundation for creating new programs and algorithms.

Additionally, modularity can also be realized as a set of *primitives*. The notion of a primitive refers to a module that cannot be further subdivided and can be combined using defined operations with other primitives to create more intricate modules. Consider, for example, constructive solid geometry [45], where geometric structures are created from a

set of chosen primitive modules, including a rectangle, a sphere, and a cylinder module. New geometries in this paradigm are created by instantiating primitive modules to create shapes. More intricate shapes are formed by combining previously created shapes through Boolean operations.

In this dissertation, we aim to provide a method for automatically modularizing humanoid agent control as a set of motion primitives. We assume that control for humanoids is structured by a set of motion primitives that can be sequenced and/or superimposed. Each primitive is a controller for performing a general capability, such as "reaching" or "punching". We assume the presence of a high level controller that sequences the primitives to achieve the agent's goals. Our focus is on how to derive such modules or primitives automatically, and express them as nonparametric, exemplar-based memory models. Each module or primitive is a model that represents a family of motions which, collectively, express a single capability. Similar to a manually-intensive approach proposed by Rose et al. [116], each module is described as a potentially sparse set of exemplars, where exemplars are motion examples of a single capability being performed. These exemplars are generalized to express an infinite family of motions that perform a capability. A family of motions serves as a memory that can be used to instantiate (or recall) motions that perform a particular capability. For example, a family of reaching motion can be instantiated or recalled to produce a reach to a particular location. Described at length in Chapter 4, representing capabilities in this way, as memory models, allows for them to be used as predictors, which enables the invocation of these models in a variety of functions.

Automated construction of capabilities as primitives is relevant to several active areas of research, including artificial intelligence, autonomous agents, humanoid robotics, and computer animation. As discussed in Section 1.3, we can avoid problems of manual design of primitives by automatically deriving them from human motion. However, the use of automatic derivation is contingent upon having a mechanism capable of extracting structure from unlabeled spatio-temporal movement data (in our case, kinematic joint angle data). We must be able to extract and cluster exemplars of capabilities from such data as an unsupervised machine learning problem. We address the problem of handling the spatio-temporal nature of motion data and the development of spatio-temporal clustering, described at length in Chapter 3. We use the combination of our spatio-temporal method for dimension reduction accompanied by a simple clustering procedure.

In summary, our goal is to develop a methodology for modularizing control of humanoid agents. Through automatic derivation of modular capabilities, we aim to endow a humanoid agent with autonomy by having it use those capabilities as a repertoire for control. To avoid manual development and bias, we use an automated approach based on capabilities demonstrated by humans.

1.2 General Approach to Autonomous Humanoid Control

We use a general structure for humanoid control consisting of four levels: plant, motor, skill, and task, as shown in Figure 1.2.

- The *plant level* is the lowest, and contains the embodiment of the humanoid agent. It provides the agent's basic interface to the world through sensing and actuation.

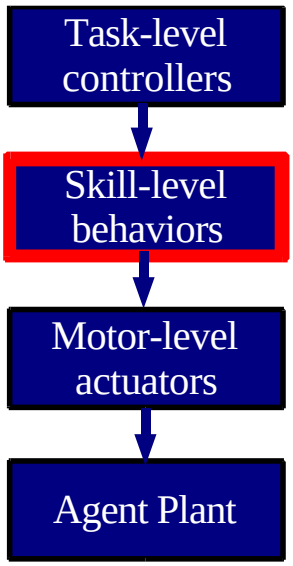


Figure 1.2: *Our general approach to autonomous humanoid control consists of: i) an agent plant as the embodied interface to the world, ii) motor level sensing and actuation for achieving desired static configurations, iii) skill level capabilities for setting configurations over time according to a motor program, and iv) task level controllers for directing skills to achieve the objectives of the agent.*

- Above the plant, the *motor level* contains controllers that produce actuation forces to the plant level. These forces are determined based on the difference between the agent's current and desired configurations.

- Above the motor level is the *skill level*, which contains a set of behaviors used to determine desired configurations for the agent over time. A behavior at the skill level is the realization of some *motor program* defining a capability for the agent. A set of skill level behaviors comprise a *capability repertoire* for that agent. Through arbitration or fusion [108], the outputs of all skill level behaviors are combined to continuously provide desired configurations to the motor level.

- Finally, the *task level* contains controllers constructed from a skill level repertoire to provide control for robot tasks in general. These controllers have objectives with respect to the world. Task level controllers accomplish their objectives by arbitrating, fusing, and indexing into skill behaviors. The work in this dissertation is focused on constructing behaviors at the skill level.

Skill level control can be effectively modularized through a variety methodologies, as discussed in Section 2.2.2. Our approach to modularizing skills for humanoid agents is inspired by the notion of primitives and behaviors, and is data-driven. Underlying behaviors are derived from time-series of kinematic data captured from human performance. Dimension reduction and clustering techniques are used to group motions within the resulting time-series that are indicative of a common underlying behavior. Using interpolation and eager evaluation, clustered groups of motion are generalized to realize behaviors.

Such derived behaviors can be used as skill level primitives in a variety of functions, including motion prediction, motion synthesis, motion classification, and imitation. Additionally, our method for deriving behaviors can be applied to produce the behaviors and skills assumed in a variety of control architectures for reactive, behavior-based, and hybrid systems [92].

Developing skill level behaviors through our methodology helps to address the problems of autonomy, interaction, and task structuring for humanoid agents we outlined above. First, skill behaviors *abstract* the details of motor level control, which otherwise typically requires desired robot configurations to be specified. Configurations for hu-

manoid robots are likely to be high dimensional (in the tens of DOF). Control in the configuration space of DOF can be both unintuitive and tedious for expressing autonomous control. This is further evidenced by the amount of detail, effort, and infrastructure required for procedures such as teleoperation and keyframe animation. Skill behaviors provide context with respect to a particular basic capability of an agent. The details of a specific motor level sequence can thus be abstracted. For instance, a reaching motion can be identified as a variation on a "reach" capability instead of a collection of parameter values specifying DOF values over time.

Second, because skill behaviors abstract low level control, these behaviors serve as *building blocks* for structuring autonomous control for robot tasks. Skills do not necessarily incorporate robot goals. Instead, skills contain preconditions and postconditions that are specific to a particular capability. In contrast, task level control explicitly incorporates objectives for the robot. Task level controllers perform sequencing and/or superposition of skill level behaviors in order to achieve the agent's objectives. Therefore, skill behaviors as abstract building blocks provide a means for addressing the questions of autonomy and task structuring for humanoid agents.

Finally, a capability repertoire provides a shared *grounding* between humans and robots to address the question of interaction. A skill behavior embodies a capability of the humanoid agent. Together, a set of skill behaviors form a *vocabulary* of capabilities for the agent. By using this vocabulary, humans can communicate directly with the robots, by translating the vocabulary into control or objectives for the robot.

In addition to modularity, behaviors express general capabilities of the humanoid agent, regardless of the specific function being performed. Borrowing inspiration from

concepts in neuroscience, interactions between skill level behaviors and a function specific mechanism (e.g., perception, control) can be viewed as functionally similar to interactions between *motor primitives* [11] and *mirror neurons* [115]. Motor primitives have been proposed as a biological model for the structure of the motor system in various animals, including humans [132]. Mirror neurons are hypothesized to map observed movement onto motor programs, possibly onto a set of motor primitives [93]. This mapping allows for movement to be executed from observation and provides a guiding principle for imitation learning. For the purposes of this dissertation, the biological concepts of motor primitives and mirror neurons serve as inspiration for the derivation of behaviors that are not specific to a particular function. Analogous to motor primitives, skill behaviors should allow an agent to perform functions for control, perception, or internal modeling using common underlying mechanisms and provide building blocks for task level control. In this sense, skill behaviors fit within the model of *perceptual-motor primitives* [93] that links perception and motor functions into a common structure.

1.3 Issues in Developing Humanoid Capabilities

Several approaches to autonomous humanoid control and learning utilize skill level capabilities [21, 90, 105, 59]. Yet, the creation and maintenance of a repertoire of capabilities for an agent is subject to several complexities. Fully automated or *tabula raza* approaches to learning capabilities are appropriate for learning mappings such as in inverse kinematics (mapping end-effector coordinates to joint angles) and inverse dynamics (mapping actuation forces to control commands). For modularization of general capabilities, how-

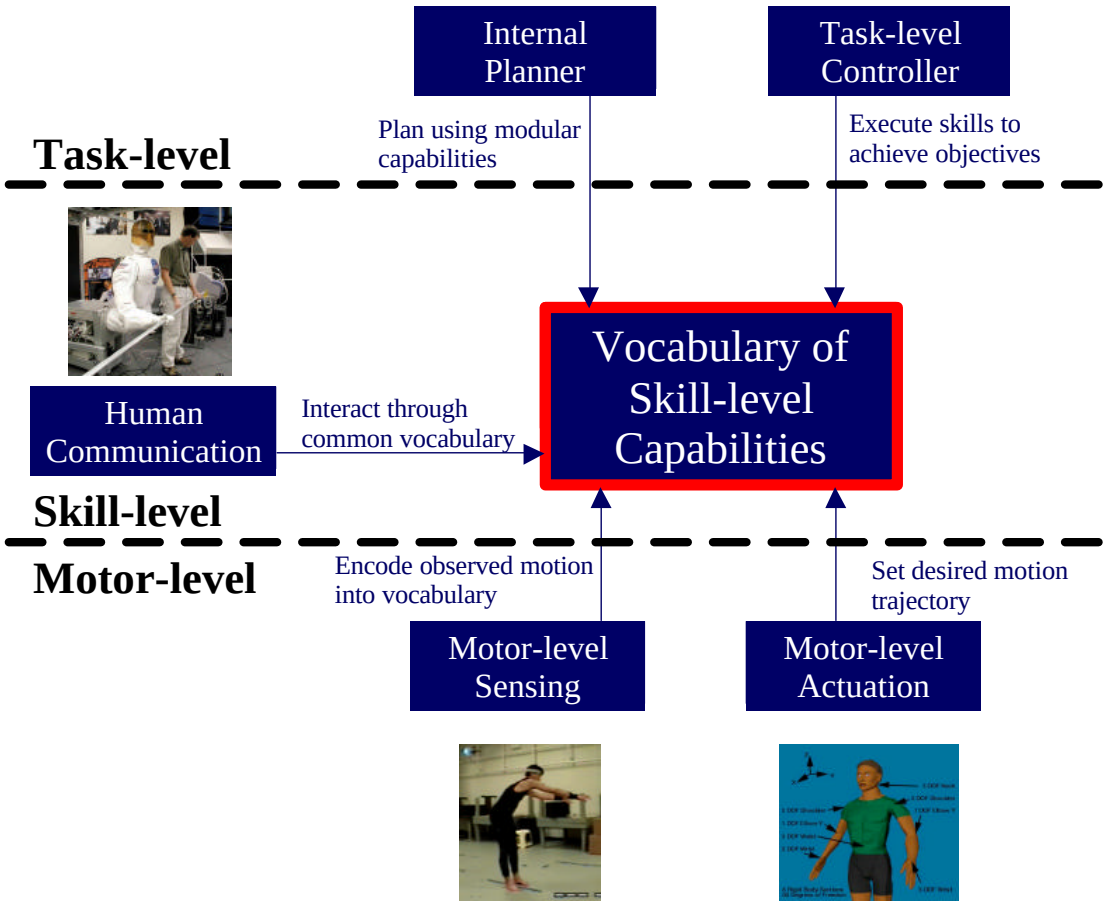


Figure 1.3: *Examples of functionality modes for interfacing with a vocabulary of skill level capabilities. These modes include abstracting motor level functions, supporting task level functions, and encoding skill level interactions. Regardless of functionality, the underlying skill behaviors should not change.*

ever, a fully automated learning approach would need both a means to explore the space of possible modularizations and judge the quality of each modularization. Given the complexity of humanoid agents, exhaustive exploration of module space is likely to be intractable. More importantly, the metric used to judge the quality of a given modularization is probably based on a hypothesis of capabilities inherent to human beings. Assuming no definitive "ground truth" knowledge about capabilities inherent to humans is available, such a modularization metric is specific to a particular domain or biased towards a particular hypothesis of human structure.

Manual development of a skill repertoire is a valid approach, especially when domain knowledge is available. While manual development appears straightforward, several issues remain unaddressed. A repertoire developer must *design* a specification of behaviors in the repertoire and *implement* the behaviors as controllers. The design of a repertoire must consider factors such as the selection of necessary behaviors, the scope and complexity of each behavior, the potential need for mutual exclusion between behaviors, and the parsimony of the repertoire as a whole. We assume that no definitive knowledge about the inherent capabilities of human beings is currently available. Consequently, a manually designed repertoire is likely to be specific to a certain domain or hypothesis about human capabilities.

Once designed, conceptualized behaviors are manually implemented as controllers for a humanoid agent. Depending on complexity, proper implementation of a controller realizing a behavior can be tedious and time consuming. In addition, manual design and implementation is susceptible to errors in human judgment and execution. Any method (automatic or manual) is susceptible to producing a repertoire of with errors. Although

errors in a repertoire are inevitable, *scalable* approaches to modularization allow such errors to be corrected through manual or automatic refinement without complication. Scalability factors include the relative ease in adding new capabilities, subtracting unnecessary capabilities, modifying existing capabilities, and rebuilding controllers from altered designs.

Within our general methodology, we utilize automated approaches to capability development to serve as an "initialization" for a scalable repertoire amenable to manual refinement. Our approach to skill acquisition avoids biasing a repertoire towards a certain model of humans by using motion capture data from human performance. Capabilities are produced in a data-driven fashion from motion data exhibiting desired capabilities, directly or indirectly. From this data-driven perspective, we avoid the issues that plague manual development by leveraging abilities demonstrated through human performance. Derived behaviors are exemplar-based and are amenable to manual or automatic modification through these exemplars, similar to Verbs and Adverbs [116]. Automatic reimplementation for controllers of modified behaviors is performed through interpolation and eager evaluation.

1.4 Dissertation Contributions

The goal of this dissertation is to provide a method for automatically deriving skills from a human subject through demonstration. These skills serve to enable and enhance other methods for task-level control and motor-level actuation, as well as the sensing and actuation capabilities of various humanoid agent platforms. As a whole, our general approach

aims to enable control through demonstration rather than explicit programming. Endowing autonomy in this manner provides an agent with the ability to collaborate, compete, learn, and train with humans other agents.

Primary dissertation contribution: The primary contribution of this dissertation is a methodology for automatically deriving skill-level behaviors from kinematic motion data collected from human performance.

This contribution can be viewed as the automated "compilation" of behavior vocabularies from existing motion data. The compiled behavior vocabulary has several desirable properties. First, each behavior is defined by a set of clustered exemplars generalized through interpolation and transition probabilities to other behaviors. From this representation, the exemplars and transitions can be modified to manually refine the vocabulary. Vocabulary compilation then serves as vocabulary initialization. Second, motion data may be perceptually meaningful to a non-specialist user but not intuitive in the space of an agents' DOF. Automated compilation serves to handle the low-level details of modularization while leaving the source and target observably intuitive. Even for users experienced with kinematic motion, automated derivation serves to reduce the amount of time and effort spent towards skill generation. Lastly, a compiled behavior vocabulary reduces the space of control into modules, providing a more accessible agent interface for non-specialist users, task-level controllers, and for human-agent interaction.

The primary contribution of this dissertation is further divided into several subcontributions:

Secondary dissertation contributions:

- automated grouping of free-space motion data into exemplars of underlying behaviors
- extension of geodesic-based multidimensional scaling for clustering behavior exemplars with spatio-temporal structure
- expression of exemplar-based behaviors as flowfields encapsulating nonlinear dynamics of an underlying skill
- using the dynamics of skill behaviors for motion synthesis, classification, and imitation

The basic representation for skill behaviors is exemplar-based. To produce exemplar-based behaviors, our methodology segments a continuous input motion and groups segments with common spatio-temporal signatures. We assume motion segments with similar spatio-temporal signatures are instances of variations on a common underlying behavior. For finding clusters of motions, we developed a spatio-temporal extension to Isomap [131, 34], a geodesic distance technique for nonlinear dimension reduction. Interpolation is used to generalize exemplars to an infinite family of variations of an underlying behavior. We developed a method to pre-evaluate a behavior for variations off-line. By performing pre-evaluation, desired variations of a behavior can be quickly indexed, eliminating the need for exhaustive search. In addition, a behavior can be used as an indivisible dynamical process, requiring neither indexing nor searching. Furthermore, behaviors as dynamical processes can be used as predictors. These predictors can be used for functions such as control, perception, and planning, without modifying the underlying behavior.

1.5 Dissertation Outline

The remainder of this dissertation is organized into six main chapters and three appendix chapters, as follows.

Chapter 2 (Background) will survey work related to learning structure from data, building agent capabilities for kinematic motion and robotics, and neuroscience inspirations for mechanisms that combine perception and motor control.

Chapter 3 (Spatio-Temporal Isomap) presents our technique for learning spatio-temporal structure and its relationship to other methods for linear, nonlinear, and spatio-temporal dimension reduction.

Chapter 4 (Performance-Derived Behavior Vocabularies) describes the application of our approach to spatio-temporal dimension reduction, described in Chapter 3, to human motion data to derive capabilities for humanoid agents in the form of behavior vocabularies. Our behavior derivation methodology is called *Performance-Derived Behavior Vocabularies (PDBV)*.

Chapter 5 (Evaluation) describes our results from deriving behavior vocabularies from input motions containing performances of multiple activities. Advantages and shortcomings of our methodology are discussed with respect to these results.

Chapter 6 (Movement Imitation with Behavior Vocabularies) describes applications of derived skill behaviors as perceptual-motor primitives for humanoid motion synthesis, classification, and imitation.

Chapter 7 (Conclusion) concludes this dissertation with a summary of the work and avenues for further research.

Appendix Chapter A (Collecting Natural Human Performance) describes complimentary approaches we developed to our behavior derivation methodology aimed towards collecting motion of humans in natural situations.

Appendix Chapter B (Applying Spatio-temporal Isomap to Robonaut Sensory Data) describes the application of our approach to spatio-temporal dimension reduction for finding structure in sensory data in grasping motions collected from teleoperation of Robonaut [1], a humanoid torso robot developed by NASA.

Chapter 2

Background

As stated previously, the aim of this dissertation is the acquisition of skills for a humanoid agent toward applications such as autonomous control. This objective is related to and/or motivated by previous work spanning several research areas, including autonomous humanoid control, motion (or trajectory) formation, machine learning and neural computation, and neuroscience. We survey related work in these areas in relation to our approach to endow a humanoid agent with skills. This discussion places our work *i)* within approaches to autonomous humanoid control, *ii)* in relation to other approaches to expressing humanoid motion, *iii)* in relation to approaches to modularization from unsupervised learning, and finally *iv)* with respect to inspirations from neuroscience.

2.1 Modularity for Autonomous Humanoid Control

As described by Matarić [92] and Arkin [3], the four dominant approaches to control for autonomous robotic agents are deliberative, reactive, hybrid, and behavior-based control. Behavior-based, hybrid, and (typically) reactive systems are inherently modularized into behaviors. However, none of these approaches to autonomous control indicate how to

choose or implement behaviors, humanoid or otherwise. For humanoid agents, methods for automatically deriving behaviors aptly compliment these approaches to autonomous control by providing human-demonstrated behaviors. As examples, derived behaviors could be used to provide skills assumed in several proposed task-level control techniques, such as hybrid systems proposed by Huber and Grupen [59], Bentivegna and Atkeson [7], and Faloutsos et al. [42, 43] and behavior-based systems proposed by Nicolescu and Matarić [105] and Rosenstein and Barto [117]. At a minimum, automatically derived behaviors can serve as a baseline for manual refinement or as a blueprint for manual development.

Deliberative control is the most traditional approach to humanoid control and is typically infeasible for autonomous control. The infeasibility of deliberative control is due to the necessity of search of an exponentially large space of actions. Deliberative techniques for humanoid agents are typically restricted to off-line procedures such as *path planning* [80, 74] and humanoid animation through constraint-based optimization [142, 29, 109]. By deriving a repertoire of skills, the space of action is modularized based on behaviors demonstrated by a human. These behaviors serve as a means for *symbol grounding* [51], abstracting assumed capabilities as symbols. Symbol grounding through primitive modules eliminates the need for exhaustive search by separating the vastness of possible control commands from task-level control. The resulting system is a hybrid system, a deliberative planner hierarchically interfaced with reactive behavior modules.

Described in depth by Schaal [120] and Schaal et al. [121], *imitation learning* is a means for endowing humanoid agent with autonomous control from human demonstration. Imitation learning can take several forms, such as direct control policy learning,

learning from demonstrated trajectories, and model-based imitation. One approach to imitation learning is to use demonstrated motion trajectories as constraints for optimization-based search, as in spline-based *via-points* proposed by Miyamoto et al. [97, 98]. Similar to deliberative systems, imitation using via-points is requires search of possible trajectories and is sensitive to spatial and temporal perturbations. Instead, the work in this dissertation primarily emphasizes learning from demonstration to build capabilities with a secondary interest in model-based imitation. For learning from demonstrated trajectories, our aim is to learn a repertoire of skill level behaviors from human demonstration, instead of using demonstration as a constraint. These behaviors in the repertoire can then be used as forward models for multiple-model competition for imitating previously unobserved movement. Our approach to imitation learning shares common objectives as the methodologies proposed by Demiris and Hayes [37] and Wolpert and Kawato [143]. Discussed in more detail in Section 2.2.2, these methods construct behaviors incrementally on-line using classification and sensory context in a greedy manner. Trading on-line learning for a global view of training data, our methodology is a batch procedure placing greater emphasis on extracting behaviors, with potentially large intra-behavior variations,.

Zordan and Hodgins [148] present an interaction-based variation on imitation learning. Unlike most of the methods discussed in this dissertation, their approach to autonomous control assumes a humanoid agent will be subject to strong dynamical interaction with its environment. Such dynamical interactions can be irresistible to the agent, such as being punched while boxing. Thus, a tradeoff arises between following a demonstrated trajectory containing the desired motion to be imitated and adjusting to the interaction

dynamics of the environment. For upper body motion, Zordan and Hodgins approach this tradeoff by incorporating contact and task constraints into *tracking controllers*. These controllers autonomously follow a demonstrated trajectory while attempting to maintain contact and task constraints in the face of strong dynamic disturbances. Along a similar line, Faloutsos et al. [42, 43] address the problem of autonomous control with strong dynamical interactions through supervised coordination of composable controllers. However, little insight is provided into how composable controllers should be designed or implemented.

2.2 Representing Motion Capabilities

For humanoid agents, we consider the primary form of expression to be articulated rigid-body kinematic motion, although, other modalities of expression are plausible and valid, such as speech, facial expression, and non-rigid deformation. Given this disposition, we classify methods for representing motion capabilities into three categories:

- motion graphs (e.g., probabilistic road maps) [79, 80]
- motion modules (e.g., behaviors) [72, 87, 63]
- motion mappings (e.g. inverse kinematics) [31]

Each motion representation is described in turn.

2.2.1 Motion Graphs

In a motion graph representation, a graph is constructed in an agent's configuration space (or a corresponding space) to represent possible valid motion trajectories. The

nodes of this graph are a set of valid agent configurations. These nodes are connected by edges, with each edge specifying the ability to perform a valid transition between two configurations. Once a motion graph is constructed, motion trajectories are specified as connected paths in the graphs. Traversing a path in a motion graph produces a motion trajectory consisting of valid configurations.

Motion graphs can be constructed using a variety of techniques. *Tabula raza* approaches to motion graph construction produce *probabilistic road maps* [75] or *rapidly-exploring random trees* [84]. These methods have no *a priori* knowledge of the configuration space and produce the motion graph through exploration. Exploration begins from an initial location in configuration space, from which nearby points and transitions are tested for validity. Points with valid transitions to the initial location are connected to the initial location with an edge. Exploration iteratively continues for added points with validity testing and edge connection to other valid configurations. *Tabula raza* motion graph construction is subject to the *curse of dimensionality*, exponential growth in the number of nodes with additional dimensionality, requiring clever graph pruning and exploration techniques.

Avoiding the curse of dimensionality, computer animation approaches to motion graph construction [79, 2] assume a set of nodes provided from previously acquired human motion. Based in methods for *video texturing* [124], we term this set of methods as *motion capture graph construction*. Methods for motion capture graph construction start from a database of previously collected kinematic motion. Configurations across all motions in the database provide the set of nodes for the motion graph without *tabula raza* exploration. An initial set of edges are provided to the motion graph by connecting sequentially

adjacent configurations from the individual motions of the database. The motion graph is completed by creating additional edges between unconnected configurations judged to be similar according to a given metric. Typically, such metrics determine if a smooth transition can be made by "splicing" two motions together at two potentially similar configurations. A *cluster forest*, proposed by Lee et al. [86], varies from a motion graph by introducing hierarchy for fast indexing. A cluster forest is comprised of a low-level motion model based on a first-order Markov process and a high-level cluster tree formed by a mixture model. Similar to [44], clusters formed by mixture models tend to provide spatial discretization than modularization.

From a utility perspective, motion graphs can be a useful representation for generating motion trajectories, but have some inherent limitations. First, motion graph structures typically require *search* in order to form desired trajectories. Searching is usually performed to connect starting and ending configurations or find an optimal path with respect to user constraints. The responsiveness of search on a motion graph is inversely related to the size and complexity of the graph. Second, non-*tabula raza* motion graphs generalize existing motion through *transitions*, placing edges through similar configurations. Generalization through transitions alone restricts our ability to incorporate new variations on existing motion without adding additional complexity to searching procedures. An alternative would be to initialize a *tabula raza*-based exploration procedure with a motion capture graph construction. Third and most important, motion graphs are typically constructed as a single monolithic structure in configuration space. Thus, they lack modularity. Skill-level behaviors derived through our methodology could be expressed as a

modular set of motion graphs. However, we prefer to express such behaviors as predictive modules that can be used for more functions than only motion formation.

2.2.2 Motion Modules

In contrast to monolithic structuring of motion graphs, the motion capabilities of a humanoid agent could be represented as a set of motion modules. Analogous to comparing deliberation and reaction, motion graphs are suited for top-down search and motion modules are suited for bottom-up coordination.

Several approaches have been proposed to modularizing the capabilities of an agent. We highlight a representative set of these approaches to humanoid agents in Tables 2.1 and 2.2. Each approach in the table is summarized by six characterizations (modularity scope, module form, inter-module interaction, repertoire design, repertoire construction, repertoire functionality). These characterizations are complimented by a brief comparison with our proposed approach to modularization, *Performance-Derived Behavior Vocabularies (PDBV)*. While PDBV appears similar to other approaches, there are important differences that make PDBV advantageous for certain situations. We compare the differences between PDBV and other approaches to modularization based on the following characteristics: input feature specificity, coarse modularization, overmodularization, model specificity, automation limited to a single class, heavy user supervision requirement, limited ability to correspond variations of a module, limited module interpretation.

Input feature specificity. Several efforts in the computer vision community have sought to find modular mechanisms capable of explaining events occurring in a video stream. Extracting structure from video is currently an open area of research with sev-

Modularization	Sco.[a]	Form	Int.[b]	Des.[c]	Con.[d]	Fun.[e]	Comparison
Autoregressive Processes [114, 113]	R	autoregressive processes	A	D (contours)	B (MLE)	P	feature dependent
Triangulated Graphs [145]	M	point graphs	A	D (points)	B (EM)	P	feature dependent
Movemes [18]	M	link LDS	C	D (video)	B (MLE)	P	over modularization
Postural Primitives [140]	D	key postures	F	Mo	M	C	coarse modularization
Torque-Fields [104, 94]	D	step and pulse functions	F	Mo	M	C	coarse modularization
Oscillators [141, 33]	R	oscillators	F	Mo	M	P or C	coarse modularization
Programmable Pattern Generators [122]	D, R	pattern generators	F	Mo	M	C	coarse modularization
EMOTE [25]	M	shape and effort params	C	Mo	M	C	model specificity
Fourier Emotion Models [134]	R	Fourier models	A, F	Ma	M	C	user heavy
Demonstrated Primitives [70]	M	trajectories	A	Ma	B (motion)	P, C	user heavy
Verbs and Adverbs [116]	M	exemplars	A	Ma	M	C	user heavy
Control Basis [50]	M	policies	A	Ma	M	C	user heavy

[a]Scope: M=multiple behaviors;S=single behavior;D=single discrete behavior;R=multiple rhythmic behaviors

[b]Interaction: A=arbitration,sequencing; F=fusion, superposition; C=coordination

[c]Design: D=data-driven; Mo=model-based; Ma=manually-decided

[d]Construction: I=incremental; B=batch; M=manual

[e]Function: P=perception; C=control

Table 2.1: (Part 1) A comparison of approaches to modularization of motion capabilities in relation to our proposed methodology, Performance Derived Behavior Vocabularies (PDBV).

Modularization	Sco.[a]	Form	Int.[b]	Des.[c]	Con.[d]	Fun.[e]	Comparison
Style Machines [17]	S	Gaussian SHMM	A	D (motion)	B (HMM)	C	single class automation
Parameterized Trajectories [9, 8]	M	trajectories	F	Ma	B (neural net)	P, C	single class automation
Autonomous Attractor [63, 64]	D, R	trajectory NLDS	F	Mo	B (RBF)	P,C	single class automation
Coupled HMMs [16]	M	Gaussian CHMM	A	D (points)	B (dyn. prog.)	P	variation correspondence
PCA-based Clustering [44, 100]	M	exemplars or graphs	A	D (motion)	B (PCA,K-means)	P, C	variation correspondence
Generalized Motor Schemas [62]	M	exemplars	A, F	D (motion)	I (class)	P, C	variation correspondence
Paired Forward-inverse Models [143]	M	predictor-controllers	F	D	I (clas.)	P, C	variation correspondence
Forward Models [37]	M	predictive models	A	D (motion)	I (clas.)	P, C	variation correspondence
Motion Textons [87]	M	LDS	A	D (motion)	B (MLE)	C	limited interpretation
PDBV [72]	M	exemplars	A	D (motion)	B (ST-Isomap)	P, C	**spatio-temporal correspondences**

[a]Scope: M=multiple behaviors;S=single behavior;D=single discrete behavior;R=multiple rhythmic behaviors

[b]Interaction: A=arbitration,sequencing; F=fusion, superposition; C=coordination

[c]Design: D=data-driven; Mo=model-based; Ma=manually-decided

[d]Construction: I=incremental; B=batch; M=manual

[e]Function: P=perception; C=control

Table 2.2: (Part 2) A comparison of approaches to modularization of motion capabilities in relation to our proposed methodology, Performance Derived Behavior Vocabularies (PDBV).

eral difficult subproblems, such as image segmentation and feature extraction, in addition to kinematic motion modularization. Consequently, several of these approaches to modularization utilize data representations specific to features that are practically extractable from images, such as points from corner detection [145], contours [114, 113], and color blobs [18]. Such features are not always amenable to usage with kinematic structures and typically ignore the control aspects of modularization. However, unsupervised methods for learning modular models from this area of research could potentially be applied to kinematic motion. Many of these techniques, however, emphasize motion classification rather than automated motion modularization.

Overmodularization. The concept of a *moveme*, introduced by Bregler [18], provides a means to modularize the motion of a single rigid body into a set of linear dynamical systems (LDS). These LDS moveme modules discretize movement for coordination across a set of rigid bodies with a *Hidden Markov Model (HMM)*. An HMM in this context is a high-level module that represents some class of whole-body movement. While movemes may be sufficient for gesture recognition, movemes are overmodularized for control purposes because modularization at such a fine kinematic level results in an exponential space of possible actions. The high-level HMM moveme coordinators could provide a usable modularization of skills. Underlying these coordinators are movemes that are ignorant of couplings between rigid bodies imposed by joints. As a result, such coordination modules may encode motion that violates the kinematic constraints of an agent.

Coarse modularization. All movement of a humanoid agent can be described by either *discrete* movement (i.e., reaching from one point to another) or *rhythmic* movement (i.e., continuously oscillatory motion) [122]. Taking these different perspectives, several

approaches to humanoid control have been centered around discrete key-posture control, such as *postural primitives* [140] and *torque-field control* [95], or rhythmic oscillators [141, 33]. Combining both approaches, Schaal and Sternad have proposed *programmable pattern generators* [122] that fuse together commands from discrete and rhythmic control systems. This modularization is supported by biological findings [123] from fMRI experimentation that human brains activate different structures to produce discrete and rhythmic motion. Ijspeert et al. [63, 64] have used discrete and rhythmic modularization to create nonlinear dynamical behaviors for movement imitation. Modularization into discrete and/or rhythmic components is suitable for movement imitation because the modules can describe the entire range of humanoid motion. Consequently, considering only discrete and rhythmic classes of motion produces a small number of modules with *broad* parameter spaces.

For behaviors other than reaching, using small numbers of broad modules places more burden on task-level controllers to perform trajectory planning and indexing into each module. Discrete and rhythmic modules provide good coarse categorizations of behaviors; however, skill level behaviors should provide a finer degree of modularization. Our approach to avoiding the problems of coarse modularization is to design our models based on capabilities demonstrated by humans in a data-driven fashion. By making this decision, we must be able to apply or develop unsupervised learning techniques able to estimate underlying behaviors from multi-activity demonstrations.

Model specificity. An alternative to data-driven modularization is to design and implement behaviors based on a hypothesized model of human movement. One such *model-based* approach to movement is the EMOTE system, proposed by Chi et al. [25].

EMOTE separates modular gestures from general movement using Laban Movement Analysis, which indicates gesture must have observable *effort* and *shape*. The properties of effort and shape are claimed to allow modular gesture behaviors to be created and parameterized without interpolation. However, these gesture behaviors must be manually observed and implemented based on a developer's interpretation of movement.

Heavy user supervision. Our focus is to modularize for multiple behaviors, with each behavior realizing a capability of the agent. One approach for creating such a repertoire is through manual design and implementation. As discussed in Chapter 1, fully manual development of a behavior repertoire is subject to many potential shortcomings in design and implementation. Previous techniques are suited to aid in manual development by providing a framework for developing modules, such as frameworks proposed by Unuma et al. [134] and Chi et al. [25], or automated procedures to help with implementation, as in Jenkins et al. [70] and Rose et al. [116]. However, such approaches to manual development serve to help only in creating controllers based on manual design, yielding similar problems of slightly less magnitude. From one perspective, issues related to manual design imply freedom for a developer to make custom decisions.

From our perspective, however, manual design alone is a potential source of problems that could be aided by automated initialization. In particular, we have found the *Verbs and Adverbs (V-A)* proposed by Rose et al. [116] a worthy approach to manually modularization into a "motion vocabulary". V-A vocabularies are comprised by a set of "verb" modules and a verb transition graph. Each verb represents an infinite family of trajectories in the joint angle space of a humanoid agent. A verb is defined by a set of exemplar trajectories in joint angle space with corresponding points in an "adverb space".

A new motion variation of a verb can be generated by selecting a non-exemplar point in the adverb space and interpolating. Exemplars are manually placed in the adverb space such that the axes of this space have an intuitive meaning. Verb exemplars can be modified in joint angle space or repositioned in adverb space to easily refine the family of trajectories represented by the behavior. Using a verb graph, motion can be generated autonomously at run-time by transitioning between verbs and smoothly connecting motion produced by two verbs at a transition. In Chapter 4, we describe how PDBV can derive V-A vocabularies automatically from human demonstration.

Single class automation. Depending on how modules are represented, data-driven learning of a capability repertoire may be limited to a single class of motion. For single class approaches, a developer must decide which behaviors to include, acquire demonstration for those behaviors, and derive modules for each behavior separately.

One such approach, proposed by Ijspeert et al. [63, 64], considers a primitive module as an nonlinear attractor encoding a demonstrated trajectory. Trajectory attractors are expressed so as to be spatially and temporally invariant and robust to perturbations. Combinations of attractor primitives can be weighted to produce complex movement. However, the emphasis in that work is not modularization but the encoding of primitive attractors from trajectories. Although Ijspeert et al. have demonstrated attractors can be created for discrete and rhythmic behaviors, modularization from multi-activity demonstration has not been addressed. A possible line of research could combine our methodology for modularization with their approach to building attractors. The emphasis of our methodology is grouping demonstration motion into exemplars. Nonlinear dynamical behaviors can be formed from clustered exemplars; however, we do not attempt

to provide the guarantees of formal dynamical systems. Instead, clustered trajectories could be used as the input for forming attractors grounded in the formalism of dynamical systems.

Another single class method, presented by Billard and Matarić [9] and Billard and Schaal [8], focuses on generalizing demonstration motion using associative memory. The method uses abstract versions of biological models of the brain to provide a hierarchy of neural networks for an associative memory. This associative memory is capable of imitating a large variety of movement with small error. The approach can work well for generalizing a single class of motion, but does not attempt to modularize demonstration into a capability repertoire. Instead, modularization is encoded into models comprising the associative memory.

Brand and Hertzmann [17] and Brand et al. [16] have approached the problem of endowing humanoid agent with capabilities using *Hidden Markov Models (HMMs)*. The *style machines* method of Brand and Hertzmann [17] explicitly aim to generalize a single class of motion. Style machines work by looking for a single general description and multiple specific descriptions in a database of motion. In the formalism of HMMs, *stylistic HMMs (SHMMs)* are proposed to represent a class of HMMs with a single structurally-general HMM and a set of style-specific HMMs. A style variable is created to allow a user to model a specific mixture of styles while retaining the general structure of the class. By restricting input data to a single class of motion, SHMMs separately learn style as specific HMMs and structure as a general HMM with a style variable. In addition to SHMMs, Brand et al. [16] have proposed *coupled HMMs (CHMMs)* for classifying motion with multiple underlying processes. While an interesting idea, CHMMs require

significant manual supervision and do not modularize training data. Instead, CHMMs are trained separately for each module to provide likelihoods for motion classification.

All of these approaches place their emphasis on generalization and not modularization. Modularization for single class approaches must occur externally by manual design or through a separate modularization procedure. Additionally, single class methods could potentially use our modularization method as a preprocessing step for behavior generalization.

In general, topics of modularization and single class generalization raise the issue of *front-end* versus *back-end* motion processing. Using our analogy of "compiling" behaviors from motion, modularization serves as a front-end system to process source demonstration motion into intermediate groupings of exemplar motions. Generalization mechanisms serve as back-end systems that transform on exemplars of an individual group to produce target behaviors for a humanoid agent. In considering potential back-end systems to pair with our method for modularization, we note in particular the *registration curves* approach of Kovar and Gleicher [78]. The registration curves approach focuses only on back-end problems for automatically generalizing a set of motion exemplars for general *motion blending* operations. Another potential back-end technique, proposed by Ramesh and Matarić [112], performs uses a hierarchical structure to perform on-line learning and representation of extended movement sequences.

Limited correspondence ability. In order to modularize appropriately, a data-driven method must be able to identify and correspond two motions that are variations on the same behavior. However, this correspondence can be difficult to establish for two motions that are spatially distal. For instance, consider two reaching arm motions that

begin from the same initial posture but reach two different Cartesian end locations. For simplicity, one reach is high (above the head) and one reach is low (below waist level). Although these two motions are structurally similar, their Euclidean distance is relatively large. Our methodology for modularization is geared to find *spatio-temporal correspondences* between pairs of motion in an off-line *batch* procedure. By using a batch procedure, all demonstration trajectories are analyzed together to find such correspondences. In the ideal scenario, modularization would occur *incrementally* on-line to incorporate modules as needed. Incremental approaches to modularization, such as those proposed by Iba [62], Demiris and Hayes [37], and Wolpert and Kawato [143], use classification to determine when new modules should be created. Without external information, however, these incremental approaches are less able to correspond structurally similar and spatially distal motions. Establishing spatio-temporal correspondence with incremental modularization would require a merging procedure to identify modules representing corresponding motion.

Fod, Matarić, and Jenkins [44] performed clustering of motions into behaviors using *principal components analysis (PCA)* for dimension reduction and *K-means clustering*. Other clustering based approaches have also been proposed, such as [100, 146]. Clustering motion in this manner led to modules that are little more than spatial decompositions of motion data. Additionally, modules from clustering in this manner provide little insight about the common theme of the grouped motions. This circumstance is problematic for controller design. Our methodology for modularization follows the same general approach of Fod, Matarić, and Jenkins by using dimension reduction and clustering while addressing the shortcomings of: *i)* using linear dimension reduction for data with nonlinear structure,

ii) ignoring the temporal dependencies that exist in sequentially ordered data for spatio-temporal correspondences, and *iii)* eliminating the need for the number of clusters to be specified *a priori*.

Modules with limited interpretation. Li et al. [87] have proposed *motion textons* as a basic unit for structuring motion capabilities. A capability repertoire for *motion textures* is represented as a set of motion textons, as linear dynamical systems (LDSs), and a texton distribution for inter-texton transitioning. Based on work by Schödl [124], motion textons are primarily used to capture the basic elements for synthesizing motion based on texturing these basic elements. Motion textons are learned with a batch procedure by an initial segmentation routine to find an appropriate number of textons and a maximum likelihood estimation (MLE) procedure to fit the textons. Our modularization method and texton learning both use a segmentation procedure as a starting point for processing. In texton learning, segmentation is performed based on an LDS error threshold for an initial division of an input motion and to find the module cardinality. In using an error threshold, segmentation and the entire learning procedure can be performed without biasing the modularization with heuristic procedures. Using LDS error, however, requires careful tuning to avoid overfitting with a small threshold or overgeneralization with a large threshold. Additionally, the linear limitation of textons may not encapsulate local structures in the motion such that each module has an observable meaning. To avoid problems in module interpretation in our methodology, we separate procedures for segmentation and motion grouping. Segmentation is performed heuristically to divide the input motion as a preprocessing step for modularization with dimension reduction and clustering. Unlike the LDS of textons, our segmentation and motion grouping proce-

dures can use completely different models for processing motion. More specifically, our segmentation procedures implicitly encode some model of motion and our motion grouping procedure is *model-free* by making pairwise correspondences between motions. In addition, decoupling segmentation and grouping allows for motion segments to remain nonlinear trajectories and, consequently, produce modules encoding nonlinear dynamics in joint angle space.

2.2.3 Motion Mappings

Complimentary to motion graphs and motion modules, motion mappings construct control spaces with correspondences to the joint angle space of a humanoid robot. The driving idea for control spaces is that desired motion can be easily and intuitively specified in control space and mapped back into joint angle space to produce control. The most popular motion mapping is *inverse kinematics (IK)* in which control is specified in operational space, Cartesian space of an end-effector. However, many motion mappings have been proposed, including principal component spaces [44, 100, 69], style variable spaces [17], and adverb spaces [116].

Motion mappings serve a complimentary purpose to motion graphs and motion modules. With respect to motion graphs, motion mappings can potentially provide more compact interpretations of joint angle space, allowing for feasible exploration and search. For purposes like motion graphs and IK, an assumption is made that a single transformation can be made to coordinates amenable for control and reconstruction. Instead, multiple motion mappings from joint angle space can be created, one mapping for each

module, to provide both modularity and accessibility. One example of modular motion mappings are Verbs and Adverbs motion vocabularies [116].

2.3 Unsupervised Learning

One of the goals of this dissertation is to provide a modularization of human motion data acquired from demonstration. In the context of *unsupervised learning* [10], this goal is the extraction of *features* from an unlabeled data set. Features, in this sense, represent some commonality in a subset of the data. More specifically, a feature is an underlying representation responsible for generating a subset of data. Input data to an unsupervised learning procedure are generated by one or more of these underlying features. Bishop [10] discusses several techniques for feature extraction, including *clustering*, *discriminant analysis*, *topographic maps*, *mixture models*, and *dimension reduction*. In this dissertation, we center our attention on dimension reduction techniques for easily distinguishable clusters for feature extraction.

Our use of feature extraction begs the question of what a "good" feature is. The "goodness" of a feature is dependent on what properties make different data points common. Several of the feature extraction methods provided by Bishop [10] assume that *spatial proximity*, relatively close distances between points, indicates commonality. Other feature extraction methods assume that the data are *temporally dependent*, i.e., that the sequential order of the data contributes to the meaning of the data. Points occurring close in time have a dependency that could indicate commonality. Motion data that we want to extract features from have both spatial and temporal dependencies. In other

words, if two motions appear to be similar or occur in sequence, they potentially belong to the same feature. More specifically, our aim is to find data that are spatially similar across a local temporal interval and correspond data points with common spatio-temporal characteristics across a potentially large volume in an input space.

2.3.1 Clustering

Clustering is the partitioning of a data set into groups (or clusters). Data members of a single cluster are assumed to represent or have been produced by a common underlying feature. We consider three basic types of clustering: *agglomerative* [68], *K-means* [10], and *mixture models* [10]. Agglomerative clustering works in a bottom-up fashion by iteratively merging data points into clusters. Such methods typically consider each data point to initially have its own cluster. These clusters are merged until a stopping condition is reached. In contrast, K-means is more top-down procedure that iteratively associates each data point with one of K clusters. Each cluster is initialized with a location in input space and updated with an *Expectation-Maximization*-like procedure consisting of two steps. The first step performs *hard assignments* to associate each data point with a single cluster based on proximity. The second step updates the location of the cluster to be the centroid of its associated data points. Iterations of these two steps are performed until convergence. Mixture models work in a similar fashion as K-means. The primary difference between these two methods is that *soft assignments* are used to associate data points and clusters. Soft assignments allow for a data point to have partial assignments to multiple clusters. Mixture models provide a more flexible framework for clustering than K-means.

Clustering using these methods may not yield representative features for modularization. This shortcoming is not necessarily a problem of the clustering mechanism, but rather the input space in which it is being applied. Clustering mechanisms typically assume spatial proximity indicates structural similarity. However, this *proximity-equals-similarity assumption* may not hold for motion modularization. To resolve this problem, dimension reduction can be used to transform an input space into an *embedding space* where clustering based on proximity is structurally appropriate. In addition to similarity issues, the aforementioned clustering techniques require the number of clusters to be known *priori* or estimated in some fashion. For modularization, however, we may not know how many clusters are appropriate or want to avoid cluster cardinality estimation. Although, if a dimension reduction mechanism produces separable clusters in an embedding space, cluster cardinality can be estimated automatically.

Along a similar line of though, Ben-Hur et al. [6] have proposed *support vector clustering (SVC)*. SVC is a support vector machine that performs clustering in a higher dimensional feature space. Clustering is performed in this feature space using optimization on an implicit embedding of the data from kernel functions centered at each point. Unlike SVC, our approach to modularization separates the process of embedding and clustering, partially to better visualize the procedure.

2.3.2 Hidden Markov Models

In the process for finding spatio-temporal structure, we are looking to model *time-series data*, data with both spatial and temporal dependencies. A natural choice for this problem are *Hidden Markov Models (HMMs)* [111]. HMMs provide a probabilistic framework for

learning and modeling the sequential structure of observed time-series data. However, the foundation of the HMM framework is suited for discrete data. In order to use continuous data, a mechanism must be included for discretizing the data into a finite number of states, i.e., for spatial feature extraction. This discretization is typically performed by some spatial clustering mechanism. In practice, however, discretizing using spatial clustering can be very sensitive to initialization and can provide non-intuitive clusters. Additionally, explicitly hidden states are not necessarily amenable modules for grounding a capability repertoire. In our method for feature extraction, we strive to transform the data so that spatio-temporal features can be clearly established through spatial clustering. In this respect, our methodology and HMMs are complimentary for discretizing and then sequencing. However, HMMs are not a necessary component in our derivation of action and behaviors, but allow for a probabilistic interpretation to be used instead of our largely geometric interpretation. We note that our goal is providing useful modules from a time-series, not necessarily time-series prediction, smoothing, or filtering.

2.3.3 Linear Dimension Reduction

Dimension reduction can be a useful preprocessing step in which data from different features can be discriminated more easily. By performing dimension reduction on a data set, we are finding a subspace *embedded* in our original data space. Data used to find this embedding are transformed into this subspace and are referred to as *embedded data*. There is a one-to-one correspondence between the original and embedded data. One common set of techniques performs dimension reduction through embedding a linear subspace. Examples of linear embedding techniques include *Principal Component Analysis (PCA)*

[10], *Factor Analysis (FA)* [49], and *Independent Component Analysis (ICA)* [60]. These techniques are not useful for modularization in general because they are unable to find nonlinear spatial structure, in particular spatio-temporal structure we are seeking. ICA could be of potential interest because of its properties for *blind source separation*, the separation of underlying sources from observed mixtures of these sources. At an abstract level, if we consider observed motion to be mixture of underlying source motions, then independent components should provide a useful set of basis behaviors.

2.3.4 Nonlinear Dimension Reduction

The popularity of linear PCA suggests the use of nonlinear PCA. However, several different methods exist for performing PCA-like embeddings in a nonlinear manner. One of these methods, *autoencoders* [40, 35], extends a linear PCA neural network with nonlinear hidden layers. Another technique, *Principle Curves* [54, 76], is a generalization of linear PCA in which principal components are curves instead of lines. The nonlinear embedding approach that we use in this thesis assumes that the data have an underlying spatial surface, or *manifold*.

Manifold-based dimension reduction can be performed in several ways. One popular method is the *self-organizing topographic map* [10] in which a set of vertices connected to form an manifold compete to represent a certain region of the data. The underlying manifold is specified by the placement of the vertices and their connections. Another approach, used by [136, 19], is to use locally linear models to directly fit subsets of the data and, thus, approximate the underlying manifold in a piece-wise fashion. While most topographic maps and piece-wise approaches use spatial distances, some spatio-temporal

approaches to topographic maps have been proposed by Chappell and Taylor [22] and Varsta et al. [135].

Another set of methods for performing manifold-based dimension reduction uses information from the perspective of each data point. These methods include *Locally Linear Embedding (LLE)* [118], *Manifold Charting* [15], *Kernel PCA* [125], and *Isomap* [131] As we discuss further in Chapter 3, these techniques maintain pairwise relationships between N data points, so that each column of a $N \times N$ represents a view of the data set from a single point. PCA is performed on this matrix to yield a nonlinear embedding.

The previously described methods are intended for data that are spatial in nature. Thus, the data are assumed to be *independent and identically distributed (IID)*, the data occur in no specific order but were drawn from the same underlying distribution. Because of this limitation, we are particularly interested in Isomap, in order to allow for the inclusion of pairwise temporal dependencies.

2.4 Motivation from Neuroscience

To help guide our thinking on modularization and on constructing perceptual-motor methods, we draw on evidence from neuroscience about the organization of the brain. As stated in [61] and elsewhere, neuroscience evidence suggests that a certain area of the human (and monkey) brain contain so-called *mirror neurons*. Such neurons become active when the human observes, performs, or visualizes a particular movement. Mirror neurons are thought to serve as an important mapping mechanism between functions for perception and control. They have served as inspiration and motivation for several ap-

proaches in humanoid robot control, including paired forward-inverse models [53], parallel forward models [36], connectionist [9], and dimension reduction [44] approaches, as well as in describing superposition of movement primitives [119, 132].

Motivated by mirror neurons, we aim toward deriving motion modules that should be usable in multiple contexts, namely perception and control. Furthermore, the modules should serve as an intermediate representation that can be encoded from one context, such as perception, and mapped into another context, such as motor control. Imitation is the most natural example of such a mapping, in which perceptual observations are mapped into control commands [93].

2.5 Acquisition of Human Motion

A critical point of our methodology is to endow a humanoid agent with human capabilities through demonstration. For this we need motion data to serve as input, in order to enable the derivation in data-driven manner. Therefore, our approach depends on a means for accurate capture of kinematic motion from humans performing various activities. A variety of systems for motion capture exist, including commercial systems based on optical [130], electromagnetic [67], exoskeletal [102], and fiberoptic [66] technologies. However, these technologies typically restrict the motion of a subject to a constrained environment or require the subject to wear heavy instrumentation. Instead, we are interested in acquiring motion of people engaged in natural everyday situations as a basis for deriving behaviors for natural activities. Thus, work related to motion capture of markerless [47, 38, 27, 23], or unobtrusively instrumented kinematic subjects is of par-

ticular relevance. We highlight two of these techniques in Appendix A and describe a method we developed to further enable this work. These approaches to motion capture will be especially useful for leveraging the ongoing "sensor explosion" currently occurring in commercial and consumer electronics, and extending well beyond the focus of this dissertation.

2.6 Summary

In this chapter, we placed our methodology for automatically deriving movement modules in the context of related previous work. We have described the need for data-driven extraction of spatio-temporal behaviors from demonstration in the context of unsupervised learning. The need for automatic derivation methods serves to provide a vocabulary of modular skill level behaviors for structuring a variety of functions. We also presented the biological motivation underlying the constructing perceptual-motor mechanisms from derived modules.

Chapter 3

Spatio-temporal Isomap

In this chapter, we discuss current techniques for dimension reduction and their applicability for finding spatio-temporal structure in data. We discuss the potential of using or extending current dimension reduction techniques for spatio-temporal feature extraction. We propose an extension of Isomap nonlinear dimension reduction [131] for spatio-temporal data. Our method for spatio-temporal dimension reduction will help uncover structure in kinematic motion, which is a focus of this dissertation.

Central to our methodology for deriving behaviors from motion is the ability to estimate the underlying structure of unlabeled spatio-temporal data. Kinematic motion is inherently spatio-temporal. A set of R joint angle values define the static posture of a kinematic structure at a particular instant of time. Because each joint angle value is a scalar, this posture is spatially defined by a point in a R-dimensional space, or *joint angle space*. Motion of a kinematic model is described by changing joint angle values over time, or a trajectory in joint angle space. We assume trajectories formed by kinematic motion from humans are indicative of an underlying structure that can be used to automatically

derive skill level capabilities for an autonomous humanoid agent. The detailed description our approach to automatic derivation of skills appears in Chapter 4.

Deriving skills in this data-driven manner is dependent on having an *unsupervised* procedure capable of modularizing spatio-temporal data. As discussed in Section 2.3.1, clustering techniques provide the most straightforward means for modularizing a data set, but are susceptible to problems when spatial proximity is not indicative of structural similarity. In such cases, we can use dimension reduction techniques to transform, or *embed*, a data set into an new coordinate system, or *embedding space*. With an appropriate technique for dimension reduction, proximity in the embedding space will indicate structural similarity, providing a coordinate space appropriate for clustering.

For clustering spatio-temporal data, a dimension reduction technique must uncover spatio-temporal structure by:

- *proximal disambiguation* of spatially proximal data points in the input space that are structurally different
- *distal correspondence* of spatially distal data points in the input space that share common structure

To illustrate our concepts of proximal disambiguation and distal correspondence, we revisit our reaching example described in Section 2.2.2. Restating this example, two arm reaching motions beginning from the same initial zero posture to different Cartesian end locations, one above the head and one below the waist. These two motions are distal in joint angle space and share common structure, united by the behavior of reaching. For disambiguation, the lower reaching motion and a retraction motion to the zero posture

may be proximal in joint angle space. However, these two motions are structurally different being particular instances of different behaviors, reaching and retraction. Thus, these two motions should be disambiguated into separate clusters, i.e., distal in the embedding space. In contrast, the two reaching motions are distal in joint angle space, but should be corresponded into the same cluster, i.e., proximal in the embedding space.

In the following sections, we discuss several existing approaches to dimension reduction with respect to their ability to estimate underlying structure in spatio-temporal data. We pay particular attention disambiguating proximal and corresponding distal data points with *i)* potentially high input dimensionality R, *ii)* spatial nonlinearity, and *iii)* temporal ordering. Additionally, we propose an extension of one nonlinear dimension reduction technique, Isomap [131], for spatio-temporal data. As a note, we assume the absence of significant amounts of noise in the motion data that are not indicative of is underlying structure.

3.1 Linear Dimension Reduction

3.1.1 Principal Components Analysis

Arguably the most well-known and widespread method for dimension reduction is *Principal Components Analysis (PCA)*, described in [10]. PCA is performed by applying an eigenvalue decomposition on the covariance matrix of a given set of data points. A subset of resulting eigenvectors, called *principal components (PCs)*, span some percentage of variance in the data. The PCs form a new linear subspace for representing the data set in a parsimonious manner. Equivalently, PCA could be thought of as fitting a R-

dimensional ellipsoid to the data and selecting its $R_{PC} << R$ major axes to serve as the new coordinate system.

PCA clearly introduces parsimony and provides reconstructability for a data set, but is not necessarily suitable for finding spatio-temporal structure for several reasons. First, PCA assumes that the intrinsic spatial structure of the data are linear. Unless a data set happens to be structured linearly, PCA embeddings tend to overestimate the dimensionality of the underlying structure, yielding PCs that are not indicative of how the data are structured. More specifically, each PCs does not provide enough insight toward the nonlinear process that produced the data set. Interpretation of embeddings produced by PCA for nonlinear data can be a fruitless undertaking. Additionally, methods such as PCA also assume that the elements of the input data are independent samples from an identical distribution, i.e., have no temporal or sequential ordering. Motion data, however, are temporally dependent via their sequentially ordering. Also, because we assume noise-free motion as input, we do not consider the methods such as *factor analysis* [49] that have an explicit noise model.

In our early joint work with Fod and Matarić [44], we encountered several of these shortcomings in applying PCA to motion trajectories captured from human arm movement. Similar to the aims of this dissertation, the purpose of this work was to extract features from the motion data representative of basic behaviors. However, we encountered an impasse in deciding what features in the embedding should be used to construct parameterized basis behaviors.

On one hand, we could use each PC axis as a feature. Clearly, each PC has a linear parameterization. Because PCs are spatially orthogonal, the PCs can superimposed

through vector summation. However, understanding the underlying meaning of each PC and superposition of PCs is extremely difficult. Even if we could create behavior modules for each PC, accessing or indexing into the resulting modules would be non-intuitive.

On the other hand, we could cluster the data in the PCA embedding into features. Clusters will typically provide features with a more intuitive meaning. However, the meaning provided by the clusters is little more than a spatial partitioning of motion. By performing PCA, we have changed the distances between data points, but not their relative placement. Two motions may be spatially similar and not representative of the same underlying movement. Consequently, the clusters provide a good spatial partitioning, but not necessarily better features.

Additionally, clustering methods, such as K-means [68], require the number of clusters to be found in the data to be specified *a priori*. In many useful situations, *a priori* cluster cardinality cannot be specified in a reliable manner. This ambiguity can be addressed through manual intervention or methods for cluster cardinality estimation. However, if a dimension reduction method can introduce feature separability or structurally significant spatial relationships, reliable cluster cardinality estimates could be produced automatically.

PCA provides little means for incorporating the ability to disambiguate and correspond spatio-temporal data. Disambiguation could be performed by *temporal windowing*, using an interval about each data point rather than the data point itself. Disambiguation without correspondence, however, limits our ability to modularize spatio-temporal data. Given the problems of PCA, our aim has been to provide a dimension reduction technique that yields clusterable features, with each cluster having an accessible meaning

with respect to its temporal and nonlinear spatial structuring. Later in the chapter, we address these problems by using nonlinear dimension reduction methods as a guide for extracting useful features.

3.1.2 Independent Components Analysis

Independent Components Analysis (ICA) [5, 60] is a method for finding structure in data as statistically independent features. Similar to PCA, ICA constructs a new coordinate system consisting of linear *independent components (ICs)*. In contrast to PCA, the ICs are statistically independent, while PCs are decorrelated (i.e., statistically independent up to the second order). Consequently, ICs will not necessarily be orthogonal if higher-order statistical dependencies are present within the data.

In further contrast to PCA, ICA is a method for *blind source separation*, providing an inherent meaning of the ICs with respect to the data. In the ICA formulation, each data point x is an observation of some linear combination of a set of underlying source components s. The weighting of each source component for each observed data point is specified in matrix A, giving the data generation model:

$$x = As \tag{3.1}$$

Assuming that the source components are statistically independent, finding ICs corresponds to estimating the underlying source components responsible for generating the data.

In applying the ICA concept to motion data, each observed motion is assumed to be a mixture of underlying source ICs. As with PCA, each IC feature could be used as a parameterized linear source behavior in joint angle space or all ICs can be used as an embedding space for clustering. Using individual ICs as behaviors can be useful for superposition in joint angle space through weighted vector summation. Because the ICs are independent rather than orthogonal, reconstruction of data from IC projections are not as accurate as using PCs, but are likely to provide more observable meaning.

The standard formulation of ICA, originally proposed by Bell and Sejnowski [5], has several problems when applied to motion data. As with PCA, underlying features in motion data are most likely nonlinear and will not be captured faithfully by linear components. Unlike PCA, ICA transforms data such that each new coordinate axis is independent. The IC coordinate axes are useful as a preprocessing step for clustering or features themselves. However, determining independent components is a procedure whose outcome is subject to several factors, such as initialization and mutual information metrics. Whereas PCA reliably produces similar components for the same data set through eigendecomposition, ICA is an iterative process that is subject to producing different components for the same data set at different times. The ambiguity in ICA is due to its sensitivity to initial estimates for the cardinality of the source components and the mutual information metric.

For temporal dependencies, ICA can consider intra-data point temporal properties. Given a set of audio recordings containing different mixtures of the same auditory event, for example, ICA can separate these mixtures into their component source signals. However, given audio samples of different events with inter-sample dependencies, ICA cannot

guarantee structural components because the samples are not *IID* (i.e., independent samples from an identical distribution) and are not necessarily mixtures of common underlying sources.

In summary, linear ICA does not aid in the proximal disambiguation and distal correspondence of spatio-temporal data. Additionally, nonlinear methods for ICA are typically restricted to classes of nonlinearity. Kernel methods for ICA, such as those by Bach and Jordan [4], may hold promise for modularizing spatio-temporal data, but that research avenue is outside the scope of this dissertation.

3.2 Nonlinear Dimension Reduction

Several methods exist for extending the capabilities of PCA and ICA to produce nonlinear decorrelated or independent components. However, nonlinear dimension reduction may not lead to the extraction of useful features in nonlinear data. Factors that may prohibit useful feature extraction include restrictions to classes of nonlinearity, unknown component ordering, and coordinating component models.

3.2.1 Autoencoders

From a neural networks perspective, dimension reduction is performed by an *autoencoder* [40, 57], a multi-layer neural network with hidden layers that optimizes for network weights towards an output layer to minimize a reconstruction cost metric. Using only linear neurons, an autoencoders provides embeddings equivalent to PCA. Demers and Cottrell [35] describe one type of autoencoder for nonlinear data. Like other methods for dimension reduction, autoencoders do not consider temporal dependencies that may

exist in data. However, a means for incorporating temporal dependencies into an autoencoder is not readily apparent. *Time delay neural networks* [83] have been proposed, but are suited for classification with temporal windowing. Thus, TDNNs provide proximal disambiguation, without making distal correspondences.

3.2.2 Principal Curves and Piecewise PCA

Two variations on PCA that address spatial nonlinearity are *principal curves*, originally proposed by Hastie and Stuetzle [54], and *piecewise* procedures, such as [19, 136, 133]. The principal curves approach is the geometric extension of a principal component from a line to a curve. In contrast, piecewise approaches represent the global nonlinearity in a data set through locally linear models. Locally linear models could be learned incrementally on-line [136], using a manifold assumption [19], or through applying PCA to partitioned data [133].

The principal curves methodology defines components as parameterized curves that pass through the "middle" of data point distribution, similar to the axis of a generalized cylinder [52]. The projection of a data point onto a principal curve is defined by the parameter value along the curve with the smallest Euclidean distance to the data point. Learning of principal curves in data are typically performed by two levels of iteration. The first level learns one component per iteration until convergence, i.e., all of the significant variability in the data has been spanned by the components. The second level occurs at each first level by iteratively fitting a new component to the null space of the existing components. Kegl et al. [76] have proposed a one-level method for learning principal curves through initialization and optimization of a topographic map structure.

In joint work with Chu and Matarić [27], we recently proposed a noniterative method for approximating principle curves, described in Chapter A, using Isomap to reduce point volumes into easily skeletonizable configurations.

As is the case with PCA, proximal disambiguation is possible for these approaches through temporal windowing. However, distal correspondence remains difficult to incorporate, mainly due to geometrical assumptions used in these approaches for local linearity and global structure.

3.2.3 Topographic Maps

Generative topograpic maps [10, 77] provide a unique combination of both clustering and dimension reduction. Topographic maps assume an embedding space is defined by a set of nodes connected in a fixed topology. Using these nodes as cluster centers in the input space, the input data are both assigned into clusters and mapped into an embedding space of *a priori* topology. Methods, such as Varsta et al. [135], have incorporated temporal dependencies into topological maps through leaky integrators, allowing for proximal disambiguation. The ability to perform distal correspondence could potentially be incorporated into temporal topograpic maps; however, such methods will remain sensitive to initialization and *a priori* topology specification.

3.2.4 Local Spectral Dimension Reduction

Avoiding many of the limitations of previous approaches, *local spectral dimension reduction (LSDR)* methods, such as *Locally Linear Embedding (LLE)* [118] and *Manifold Charting* [15], globally coordinate local representations that distributed over a data set.

LSDR methods are similar to piecewise PCA and topographic maps in that locally linear models are used to represent globally nonlinear spatial structure. However, LSDR methods perform global coordination based on the underlying structure of a data set, unlike topographic maps that coordinate based on an *a priori* structure and piecewise models that perform no explicit global coordination.

LSDR techniques work by distributing a set of local models across a data set, either at each data point or by cluster assignments, and coordinating these models through eigendecomposion or optimization. In contrast to techniques for *global spectral dimension reduction (GSDR)*, LSDR methods have explicit local models. These models provide an *intermediate representation* of the data, which allows for the definition of the forward and inverse mappings between the input and embedding spaces. GSDR methods lack this representation and, hence, lack the ability to produce defined mappings, using direct distances (or dissimilarity measurements) between all data pairs. While LSDR methods provide representation explicitly, commonly utilized local models are inherently spatial and leave room for incorporating temporal dependencies and the ability to correspond.

3.2.5 Multidimensional Scaling and Global Spectral Dimension Reduction

Multidimensional scaling (MDS) [30, 13] is an approach to dimension reduction that uncovers hidden structure in data by preserving pairwise distances or dissimilarities. More specifically, given an $N \times N$ matrix D of pairwise distances between N points that could be generated from point coordinates in R^P, one can produce coordinates for the

points in an R^R embedding, $R << P$, such that pairwise distances are proportionally preserved.

MDS can be performed with a variety of distance metrics and computation techniques. We will focus on deterministic metric MDS techniques that compute embeddings through eigendecomposition. Steyvers [129] provides a summary of MDS techniques and a survey of variations on the basic MDS approach. Metric MDS uses continuously valued distances between points and nonmetric MDS uses rankings of dissimilarity with respect to a given point. Metric MDS provides more accurate embeddings, while nonmetric MDS is independent of a specific distance metric. Deterministic MDS treats each point independently and pairwise distances are considered undistorted. In contrast, probabilistic MDS [89] assumes pairwise distances are subject to distortion and abstracts distributions of points into probability distributions. Deterministic MDS is a more straightforward technique that does not require additional distribution information, but lacks the flexibility of probabilistic MDS to include information about priors. MDS is typically computed through iterative optimization from an initialized configuration towards a configuration reflective of an input D matrix. Instead, MDS can be performed noniteratively through eigendecomposition on the D matrix. Being noniterative, MDS through eigendecomposition is not susceptible to problems with local extrema and explicit initialization, but typically provides a close approximation to an optimal embedding. If local extrema are avoided, MDS through optimization will typically provide more optimal embedding with respect to D.

Methods for GSDR use MDS at their core to transform pairwise relationships into new (potentially lower dimensional) coordinate spaces reflective of the underlying structure

in a data set. In order to enable such a transformation, pairwise distances that are indicative of the underlying structure must be created from the input space. Given such pairwise distances, a MDS procedure can realize an embedding space indicative of the underlying structure. GSDR methods can utilize MDS via optimization, however, we primarily consider GSDR methods that perform MDS through eigendecomposition, such as *Kernel PCA* [125] and *Isomap* [131].

3.2.5.1 Kernel PCA

As Williams explains [139], metric deterministic MDS is related to *Kernel PCA (KPCA)* [125], which is a kernel-based nonlinear extension of PCA proposed by Schölkopf et al. KPCA is similar to MDS in that embedding is performed through eigendecomposition on a similarity matrix computed from input point coordinates. However, the intuition for understanding the KPCA mechanism is quite different. This intuition is to transform data into a higher-dimensional feature space such that linear PCA in the feature space corresponds to nonlinear PCA in the input space. This rationale is conceptually useful because it can be described as well-understood linear PCA with a nonlinear preprocessing mapping. The caveat to KPCA is that the nonlinear mapping to feature space does not exist explicitly and, thus, feature space coordinates are unknown.

By using the *kernel trick*, KPCA can be performed with explicit feature space coordinates by using pairwise similarities between points. The kernel trick basically allows for an algorithm to implicitly construct a nonlinear mapping function by using kernel-based inter-point dot products, as commonly used for support vector machines [103]. KPCA modifies the linear PCA mechanism using the kernel trick to eliminate the need for ex-

plicit feature space coordinates. Instead, feature space dot products between pairs of points are constructed based on a chosen kernel function centered at each point. The dot product between two points in feature space is computed as the scalar value of a point x with respect to a kernel centered at point y in input space. Based on the choice of kernel, a nonlinear mapping intrinsic to the data are implicitly created.

At an intuitive level, the kernel trick is making a similar assumption as MDS. The core of the kernel trick assumes that dot products between points in input space are proportional to their dot products in feature space. In the context of MDS, the kernel trick builds similarity values as kernel dot products that are independent of a specific coordinate system or configuration. Consequently, the problem of choosing an appropriate kernel is another perspective of choosing an appropriate distance metric for MDS.

3.2.5.2 Isomap

One problem with KPCA is that distances for distal points are typically based on a global Euclidean distance metric. While Euclidean-based measurements work well for proximal data pairs, structural distances between distal points may not be based on a Euclidean measurement due to properties of the underlying structure. Isomap, proposed by Tenenbaum et al. [131], addresses this problem by globally coordinating proximal pairwise distances using all-pairs shortest paths distances. Isomap works using a three-step procedure, shown below.

Isomap assumes underlying structure is manifested as a bordered manifold. This underlying manifold can be uncovered by Isomap, given the input data set is dense enough to *i)* cover the entire manifold and *ii)* form single connected component. A single con-

nected component covering this manifold approximates all-pairs geodesic distances on the underlying manifold. By applying MDS to a matrix of geodesic distances, nonlinearities in the data due to the manifold are removed to produce a coordinate space intrinsic to the underlying manifold.

As with KPCA, a desirable property of Isomap is that no explicit model is used to measure pairwise distances. Isomap and KPCA use distance metrics based on an underlying manifold, however, different underlying structures can be uncovered by "simply" using a different distance metric. In contrast to KPCA, Isomap relies only on measuring distances between proximal points and uses shortest-paths coordination for distances between distal points. The utilization of coordination for distal points typically provides better uncovering of structure, given sufficient data density. In addition, pairwise coordination provides a means to perform distal correspondence, as we describe in the remainder of this chapter.

The three main steps in Isomap:

1. compute local neighborhoods based on proximal spatial neighbors;
2. globally coordinate local neighborhoods into a full distance matrix D by computing all-pairs shortest paths;
3. embed D using MDS.

3.3 Spatio-Temporal Isomap

We developed an extension of Isomap for data with both spatial and temporal dependencies, called *Spatio-temporal Isomap (ST-Isomap)*. ST-Isomap retains the general framework of spatial Isomap for constructing a distance matrix that is embedded through

eigendecomposition. Recapping, Isomap computes distance matrices by *i)* constructing neighborhoods of points local to each data point based on spatial distance and *ii)* computing shortest-path distances between all data pairs starting from distance-weighted edges from local neighborhoods. Within this framework, ST-Isomap includes temporal dependencies between sequentially adjacent points by *i)* proximal disambiguation through temporal windowing (including spatial distances between temporally-extended windows of data points) and *ii)* distal correspondence through *common temporal neighbors* to reduce distances between similar points with respect to the underlying spatio-temporal structure. The second of these adjustments highlights the essential difference between spatial and spatio-temporal Isomap, in that certain points may be spatially distal but spatio-temporally proximal. More specifically, these points are corresponding points in the same spatio-temporal process.

To illustrate further, consider again our example of two arm reaching motions, one reaching high and the other reaching low. Postures occurring half-way through each of these motions are spatially distal and yet are equivalent positions along the underlying reaching spatio-temporal process. The aspect that separates these postures is the space of variation within the reaching process.

We describe two methods of spatio-temporal Isomap for sequentially continuous and segmented data. *Sequentially continuous* ST-Isomap works directly on the input data without abstraction into higher-level features. However, this version of ST-Isomap is not always computationally feasible due to storing and performing eigendecomposition on an $N \times N$ matrix, where N is the number of input data points. In contrast, *sequentially segmented* ST-Isomap first abstracts the data into segment intervals to provide a more

compact representation. This method is less computationally intense than its continuous counterpart, but its accuracy is highly dependent on the ability to faithfully segment input data. The basic procedure of ST-Isomap is outlined in the following.

The four main steps in ST-Isomap:

1. compute local neighborhoods: based on proximal spatial neighbors and adjacent temporal neighbors;
2. identify *common temporal neighbors (CTN)*: to identify data pairs with hard spatio-temporal correspondences:
   (a) reduce their distances by some scalar c_{ctn},
   (b) reduce distances between *adjacent temporal neighbors (ATN)* by some scalar c_{atn};
3. globally coordinate local distances into a full distance matrix D by computing all-pairs shortest paths;
4. embed D using MDS.

3.3.1 The Extendability of Isomap

Our decision to use Isomap as the basis for uncovering structure in spatio-temporal data stems from its flexibility for various distance metrics. Isomap is a combination of global coordination of proximal distances and multidimensional scaling to produce embeddings that uncover the underlying topology in a data set. Spatial Isomap constructs local neighborhoods and their edge weights based solely on Euclidean distance. Euclidean distances can be used because the underlying structure of input data are assumed to be a continuous nonlinear manifold. This manifold assumption allows Euclidean distances to be structurally valid for spatially proximal data points, but requires distal points to be coordinated globally through computing shortest-paths distances. Through shortest-paths global coordination, the distances formed between distal points are representative

of geodesic distances intrinsic to the subspace of the manifold. The MDS embedding of this matrix simply provides a new coordinate system that preserves these intrinsic distances.

The manifold assumption, however, may not hold for, or be completely representative, of the data with different types of underlying structure. For spatio-temporal data, the point was illustrated by our arm motion example at the beginning of this section. Fortunately, the manifold assumption is specific to using Euclidean distances for determining and weighting local neighborhoods. Data structured by different underlying representations can be found in the Isomap framework with appropriate mechanisms for determining and weighting local neighborhoods.

The values in a distance matrix used by Isomap are flexible to modification because there is no explicit representation for pairwise relationships. In contrast, methods like LLE have explicit linear models placed at each data point. Pairwise relationships are based on the weights of neighbors toward the fitting of a linear model at a data point. Explicit local models would be difficult to adjust for structures violating the manifold assumptions. Local models provide an explicit representation of the mapping between input and embedding spaces, while for Isomap mapping between spaces has to occur through another mechanism.

3.3.2 Issues in Applying and Extending Isomap

Isomap is a simple, accurate, and flexible approach to dimension reduction, but it has several shortcomings that must be addressed in its application. These shortcomings are related to the size and density of the input data, forming connected components, and

topologically closed underlying structures. We discuss these aspects of Isomap and how they are addressed for spatio-temporal Isomap.

Regardless of the distance metric used, the basic Isomap framework is computationally sensitive to the size of the input data, but not its dimensionality. Methods like PCA are sensitive to input data dimensionality in building and performing eigendecomposition on an $R \times R$ covariance matrix (for the basic technique). In contrast, the MDS step in Isomap requires computation and eigendecomposition of an $N \times N$ distance matrix. As R grows large for PCA and N grows large for Isomap, embedding becomes intractable due to memory and computational limitations. As a note, Isomap does not use the input data once the distance matrix is computed and, thus, is not dependent on the dimensionality of the input data. In short, Isomap is computationally appropriate for a relatively small number of points with high dimensionality, whereas methods like PCA are suited for large numbers of lower dimensional points.

In order to reduce the sensitivity of Isomap to input size, de Silva and Tenenbaum [34] introduced *landmarks* into Isomap. The reasoning for this modification, called *Landmark MDS*, is to reduce the size of the distance matrix to $N \times M$, where M is the number of landmarks. The distance matrix in Landmark MDS is computed between an $M < N$ subset of points from the input data and all of the other N input points. MDS is then performed on this matrix to provide a *singular-value decomposition*-like alternative for Isomap. By using landmarks, the diminished accuracy of the embedding is exchanged for a lighter computational burden.

In addition to being sensitive to input size, Isomap expects the distance matrix to be representative of a single connected component. More specifically, once local neigh-

borhoods are formed, all pairs in the input should have a connecting path through local neighborhoods. If all data pairs are connected, a single connected component will be found through shortest paths computation. If not, then the input data and resulting distance matrix will be split into at least two subsets of connected components. Multiple connected components causes two problems for spatial Isomap. First, distance matrices with multiple connected components are singular. Consequently, a single embedding including all of these components cannot be produced through eigendecomposition. Second, in order to avoid disconnected components, Isomap assumes the input data contain a dense sampling of the underlying manifold. This assumption is problematic because it requires a dense sampling of an unknown structure and can drastically increase the input size.

ST-Isomap avoids the problems of multiple connected components by not applying the manifold assumption and by including temporal dependencies. For sequentially segmented spatio-temporal Isomap, the purpose of the embedding is to uncover clusters. Disconnected components are not necessarily a problem for clustering (but are somewhat of a nuisance) because the disconnection between components is a coarse partitioning of the points. Each connected component can be embedded separately to find clusters in each of the partitions, providing an overall clustering of the data. However, if all of the input data are temporally related (i.e., occurring in a single sequence), a single connected component is guaranteed for ST-Isomap. This single connected component occurs because ST-Isomap includes temporally-related points (i.e., points adjacent in the sequence, or *adjacent temporal neighbors (ATNs)*) in the same local neighborhood. Thus, all data pairs will have a connecting path through temporal neighbors in local neighborhoods.

Another considerable shortcoming of Isomap is its limitation to bordered spatial manifolds. Consider input data sampled from a circle in 2D. Even though this data are in 2D, they are intrinsically structured by a closed 1D manifold. Spatial Isomap would not be able to uncover the underlying structure of this data because the loop of the circle would have to be broken to produce a 1D embedding.

For closed manifolds without self-intersections, ST-Isomap will not be able to perform any better than spatial Isomap. For self-intersecting manifolds, however, spatio-temporal Isomap has the potential to disambiguate points proximal to intersections of these manifolds. Spatio-temporal Isomap uses the spatio-temporal signature of the input by using a temporal window around each data point and its sequential neighbors. If this window is large enough, the spatio-temporal signature between non-spatio-temporally corresponding points with proximal spatial distance will be distinguishable. As with spatial neighborhoods, the size of the temporal window should be large enough to reflect the local spatio-temporal properties of a data point, but small enough to not be obfuscated by the global properties of the motion.

3.3.3 Incorporating Temporal Dependencies

The basic idea in spatio-temporal Isomap is to assume temporally adjacent points are locally related and incorporate these relationships into the distance matrix. Tenenbaum et al. [131] allude to the incorporation of temporal dependencies in their original Isomap paper, discussing the topic only as a side issue. The most straightforward means for temporal incorporation would be to work in phase-space (i.e., input data concatenated

with velocity information) or through temporal windowing. Such approaches will help with proximal disambiguation, but not distal correspondence.

Incorporating temporal dependencies by only modifying the representation of the input data, however, may lead to problems in interpreting the resulting embedding due to *soft spatio-temporal correspondences*. More specifically, there is no means for points that are spatio-temporally corresponding but spatially distal to be placed into proximity in the embedding. Because spatio-temporal structure is our objective, it is beneficial for large spatial variations to be removed from spatio-temporal corresponding points. *Hard spatio-temporal correspondences* embed corresponding points such that they are relatively proximal to each other than rest of the input data. In contrast, soft correspondences retain the spatial variation of the augmented input data. By removing the spatial variation of correspondences, hard correspondences allow for the separation of spatio-temporal structure and input space variation.

Considering the example of low and high arm reaching motion from earlier in this chapter, soft correspondences could produce an embedding such that the distance between two postures occurring half-way and 2/3-way through the same reach is smaller than two halfway postures in different reaches. In contrast, hard correspondences will embed the two halfway postures into proximity, closer than a 2/3-way posture. Because corresponding points are in relative proximity, the embedding provides a relative structural sequencing or timing description that compliments the spatial variations of the input space.

3.3.3.1 Common Temporal Neighbors

For our approach to ST-Isomap, we identify points with hard spatio-temporal correspondences as *common temporal neighbors (CTNs)*. More specifically, a data point t_x has a hard spatio-temporal correspondence with another point t_y if $t_y \in CTN(t_x)$. The specifics of how a datapair are corresponded as CTNs can vary. These differences will be described later in this section for *segmented CTN* and *K-nearest nontrivial neighbors* for sequentially segmented and sequentially continuous ST-Isomap. CTNs are assumed to be symmetric, $t_y \in CTN(t_x) \Leftrightarrow t_y \in CTN(t_x)$, and transitive, $t_y \in CTN(t_z)$ and $t_z \in CTN(t_x) \Leftrightarrow t_y \in CTN(t_x)$. *CTN transitivity* allows hard correspondences between data pairs to be propagated, forming a distinguishable connected component, or a *CTN component*.

By reducing the distances between CTNs once found, CTN transitivity can be transparently propagated by a shortest-paths procedure within Isomap to be realized in the resulting embedding. After shortest-paths, members of a given CTN component will have significantly smaller distances between any other intracomponent members than any extracomponent data point. Consequently, in the embedding produced through MDS, all members of a CTN component will be relatively proximal to each other and all external points will be relatively distal. Thus, the CTN component will be separable, or *clusterably proximal*, in the embedding, allowing for decisive interpretability of the uncovered spatio-temproral structure.

In addition to using CTNs, spatio-temporal Isomap incorporates a few other features to leverage the temporal relationship among the input data. Particularly useful for finding higher-levels of spatio-temporal structure, distances between adjacent temporal neighbors

can be reduced by a chosen scalar c_{ATN}. *ATN distance reduction* effectively serves to collapse sequentially adjacent points towards proximity. The utility of ATN distance reduction can be further enhanced when applied only to ATNs sharing CTNs to collapse only points indicative of hard spatio-temporal correspondences. Also, spatio-temporal structure can be further accentuated by increasing distances between local neighbors that are not spatio-temporal correspondences. One method of *non-CTN accentuation* is to offset all pairwise distances by some spacing distance before CTN determination. A more precise method for non-CTN accentuation is to increase distances between non-CTN pairs. These pairwise distances may also be set to some large constant value to remove the spatial relationship between non-CTNs.

3.3.4 Sequentially Continuous Spatio-temporal Isomap

We present two techniques for applying ST-Isomap to data. The more straightforward of these techniques applies ST-Isomap directly to the data and assumes temporal coherence. This technique is called *sequentially continuous spatio-temporal Isomap* and consists of the following steps (with items specific to this technique in bold):

1. compute local neighborhoods: **nearest points based on Euclidean distances between temporal windows about each point**;
2. identify common temporal neighbors: **as nontrivial matches within a local neighborhood**;
   (a) reduce their distances by some scalar c_{ctn}
   (b) reduce distances between ATNs by some scalar c_{atn}
3. globally coordinate local distances into a full distance matrix D by computing all-pairs shortest paths;
4. embed D using MDS.

The first step in this procedure builds distances that are spatio-temporal through temporal windowing. A temporal window is a local observation of the spatio-temporal process through a given data point and its ATNs (within some interval). Euclidean distance provides a similarity measure between pairs of these local spatio-temporal observations. Local neighborhoods are constructed based on points with the highest similarity measurements. As stated previously, neighborhood construction in this manner can proximally disambiguate, but not distally correspond.

To determine hard correspondences for this technique, we eliminate *trivial matches* from local neighborhoods. In other words, we determine CTNs as the *K-nearest nontrivial neighbors (KNTN)*. Our notion of KNTN was inspired by Chiu et al. [26], who define the concept of trivial matches in univariate time-series data for data mining purposes. Diverging slightly from their definition, we consider a point t_y to be a *nontrivial match* within the local neighborhood of a point t_x if $x = y$, $t_y \in ATN(t_x)$, or $D(t_x, t_y) < D(t_x, t_z)$ for all z within a temporal vicinity (i.e., $\forall_z |x - z| < \epsilon_{tv}$). The KNTN of t_x are its K most similar nontrivial matches. Given K, a data point $t_y \in KNTN(t_x, K) \Rightarrow t_y \in CTN(t_x)$ for sequentially continuous ST-Isomap.

Once CTNs are found as KNTNs and their distances are reduced, spatio-temporal Isomap proceeds to perform shortest-paths and MDS just as in spatial Isomap.

3.3.5 Sequentially Segmented Spatio-temporal Isomap

The second technique, *Sequentially segmented spatio-temporal Isomap*, addresses the input size problem of its sequentially continuous counterpart by trading embedding precision for lighter computation. A problem in using sequentially continuous Spatio-temporal Isomap

is its sensitivity to the size of the input data. As stated previously, this technique requires the storage and eigendecomposition of an $N \times N$ matrix, which can be an infeasible memory and/or computation load as N gets significantly large. For the sequentially segmented technique, a segment preprocessing step is introduced into spatio-temporal Isomap to permit its usage on larger data sets:

1. **segment preprocessing: partition input data into intervals, replacing input data with N_s higher dimensional segments**;
2. compute local neighborhoods: **K-nearest neighbors using Euclidean distance**;
3. identify common temporal neighbors: **detect similar neighborhood hops as segmented common temporal neighbors**;
   (a) reduce their distances by some scalar c_{ctn}
   (b) reduce distances between ATNs by some scalar c_{atn}
4. globally coordinate local distances into a full distance matrix D by computing all-pairs shortest paths;
5. embed D using MDS.

The initial step in this procedure is the abstraction of the *sample-atomic* input data into *segment-atomic* intervals. Because the Isomap framework is relatively insensitive to high-dimensional data, ST-Isomap is better equipped to handle the input data as a smaller number of higher dimensional segments rather than a large number of lower dimensional samples. In abstracting the input data as segments, however, the quality of the produced embeddings for structure is directly dependent on the segmentation and preprocessing of the input samples. In addition, segmentation presents a particularly challenging problem as there is no definitive ground-truth domain-independent models or mechanisms to guide the abstraction of the input samples. If such models or mechanisms existed, we could

simply analyze those models and/or mechanism to determine the underlying structuring of the data.

This "chicken-and-egg" situation for segmentation between segment generation and known underlying models leads us to segment the data heuristically. In our approach to segmentation, we scan the input data samples for events that indicate the boundaries of meaningful segments. Our framework for spatio-temporal Isomap is not specific to any particular segmentation mechanism and boundary event definition. In order to produce a structurally appropriate embedding, however, the segments produced from the input samples must be *consistent* (i.e., similar input intervals produce similar segments) and *atomic* (i.e., the user considers each segment to contain a conceptually and/or meaningfully indivisible performance/subsequence of the input data). The segments do not necessarily need to be mutually exclusive (i.e., not overlapping in time), but non-overlapping segments are recommended. We discuss segmentation methods specific to kinematic motion in Section 4.2.

Once segments are found, additional preprocessing is applied to normalize the segments into a common representation. Because each segment is given to be atomic, we consider each segment to represent a point in a $R \times l_i$ dimensional space, where R is the dimensionality of the input data and l_i is the number of samples in a segment. However, the segments produced from the input data are likely to be variable in length. This variation in dimensionality prohibits the usage of dimension reduction techniques to this segment data. In order to eliminate dimension variability, the segments are normalized to a fixed length l by fitting a cubic spline to each segment and sampling the spline for l uniformly spaced samples.

After preprocessing the input data, hard spatio-temporal correspondences are found between segments as *segmented common temporal neighbors (SCTN)*. SCTN are found as local neighbors that temporally transition to the same neighborhood. The idea driving SCTN is that two segments that share a common structuring A will always be followed by segments with a common structure B. By corresponding SCTN locally, components of SCTNs with distal correspondences can be found from shortests paths. Because members of a component for A are found implicitly using the members of B, the consequence of this grouping is a spatio-temporal structure like $A \rightarrow B \rightarrow C$, indicating spatial variations within each structure and temporal transitions across structures.

For estimating SCTNs, local neighborhoods are found as in spatial Isomap, potentially through K-nearest neighbors using Euclidean distance. From these neighborhoods, we define a data point $t_y \in SCTN(t_x)$ if $t_y \in nbhd(t_x)$ and $t_{y+1} \in nbhdx + 1$. The intuition for SCTN is that a pair of points are spatio-temporally similar if they are spatially similar and the points they transition to are also spatially similar. Spatial similarity for this definition is determined proximally by local neighborhoods. In a loose analogy, sequentially segmented ST-Isomap provides a "kernelized HMM"-type structure where clusters found as common structures serves as latent variables that are transitioned between.

3.4 Summary

We have developed spatio-temporal Isomap as a dimension reduction method for extracting structure from spatio-temporal data. Unlike other techniques for dimension reduction, spatio-temporal Isomap attempts to correspond structurally similar data points that are

spatially distal, as well as disambiguating spatially proximal data points. Two variations of spatio-temporal Isomap were described for continuous and segmented input data.

Chapter 4

Performance-Derived Behavior Vocabularies

In this chapter, we use sequentially segmented spatio-temporal Isomap, described in the previous chapter, to derive a vocabulary of perceptual-motor action primitives and behavior modules from human motion data. The derived behavior vocabulary contains the basic primitive skills and the compound meta-level behaviors that constitute a repertoire of capabilities for a humanoid agent. These capabilities form a "vocabulary" of behaviors that can serve as the foundation for autonomous control of the agent.

We present *Performance-Derived Behavior Vocabularies (PDBV)* as an approach to automatically derive modular capabilities, for an autonomous humanoid agent, from kinematic motion data of humans. PDBV takes as input a single continuous time-series of kinematic configurations collected from humanoids, preferably from real-world human performances. As output, PDBV produces clusters of motion that modularize the input motion into primitive and meta-level behaviors. Each primitive behavior represents a basic capability as a family of motions with a common theme and can be realized as a nonlinear dynamical system in the joint angle space of the agent.

PDBV assumes the input motion data are structured by an underlying spatio-temporal process. This underlying structure is an essential step in estimating the behaviors underlying an input motion. The use of sequentially segmented spatio-temporal Isomap to uncover this structure is the driving theme in the four main steps of PDBV.

The four steps in PDBV:

1. Preprocessing: producing time normalized motion segments from the input motion data
2. Exemplar grouping: grouping motion segments into behaviors based on common spatio-temporal signatures
3. Behavior generalization: forming nonlinear dynamical systems for behaviors from grouped exemplars
4. Meta-level behavior grouping: grouping exemplars of lower-level behavior based on higher-level spatio-temporal signatures

Behavior vocabularies derived by PDBV serve as a substrate of skill-level capabilities for an autonomous humanoid agent. By expressing behaviors as dynamical systems, behaviors have the ability to predict future kinematic configurations from a current configuration. As described in Chapter 6, providing a humanoid agent with predictive behaviors endows the agent with the ability to produce control desireds from behavior predictions and to classify previously unobserved motion by matching behavior predictions to observed outcomes.

PDBV works with free-space motion data regardless of the method of collection (e.g., motion capture, robot actuation, computer animation, etc.). The quality of motion from any source is clearly critical, with respect to factors such as noise, sampling frequency, and expressiveness. However, we recommend using motion captured from humans as

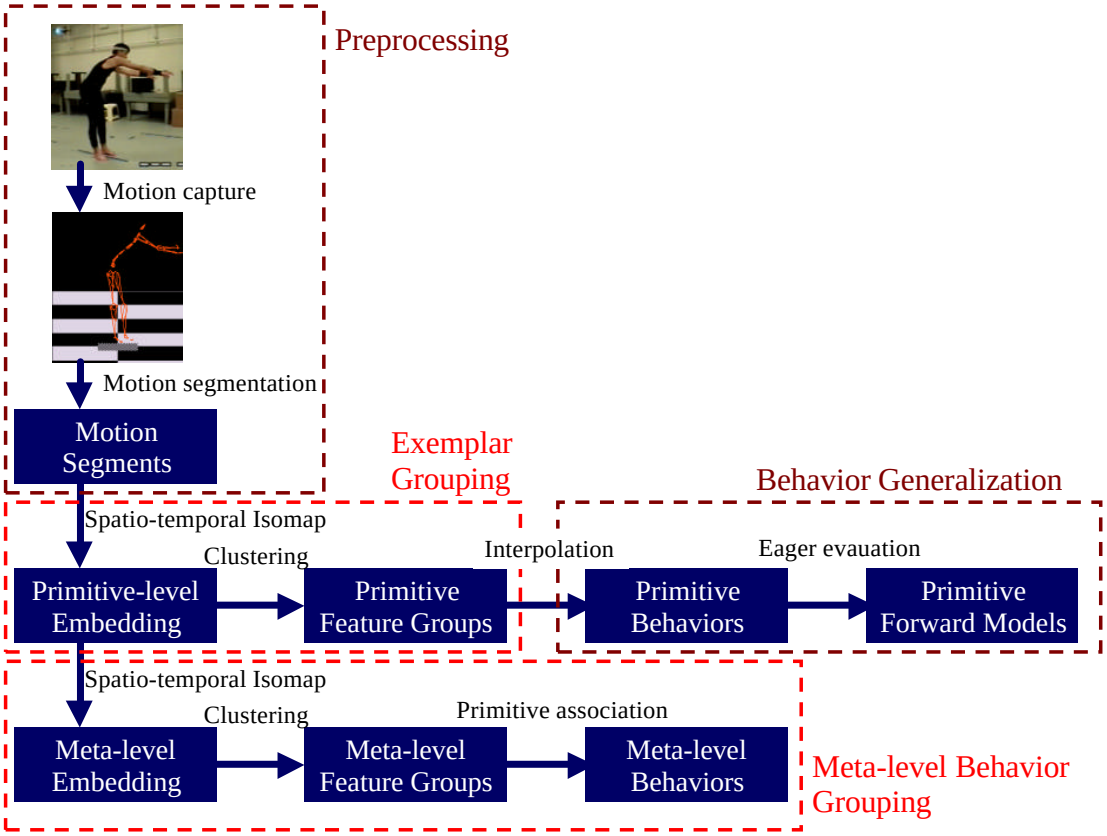


Figure 4.1: *Performance-Derived Behavior Vocabularies consist of four main steps: preprocessing, exemplar grouping, behavior generalization, and meta-level behavior grouping. Preprocessing produces a data set of motion segments from real-world human performance. Exemplar grouping uses spatio-temporal Isomap to cluster motion variations of the same underlying behavior. Exemplars of a behavior are generalized through interpolation and eager evaluation. Compositions of primitive behaviors are found as meta-level behaviors by iteratively using spatio-temporal Isomap.*

input for PDBV. We assume motion produced by humans is structured (consciously or unconsciously) by an underlying set of behaviors. It is most likely that behaviors captured from human performance will be specific to demonstrated activities and not general across all possible activities. However, by estimating underlying behaviors, we can endow an autonomous humanoid agent with human-like skill capabilities without extensive manual intuition, biasing, or implementation.

4.1 What is a Behavior Vocabulary?

Motivated by Verbs and Adverbs motion vocabularies [116], our definition of a behavior vocabulary includes two types of behaviors, *primitive* and *meta-level* behaviors. The set of primitive behaviors is the substrate of basic skills for the humanoid agent.

Each primitive is a parameterized family of motion trajectories in the joint angle space of the humanoid agent. All motions within a primitive have a common spatio-temporal structure, with each motion providing a variation on the underlying structure of the primitive. A primitive is defined by a set of *exemplar* motion trajectories. An interpolation mechanism generalizes these exemplars to encompass the *span of variations* for a primitive. Interpolation also provides the mapping mechanism between the input space (joint angle trajectories) and the parameterization space (adverb space in V-A terminology) of the exemplars. More specifically, non-exemplar points in the lower-dimensional parameterization space are interpolated to new non-exemplar motion trajectories in the input space. Meta-level behaviors represent behaviors with structure at higher levels than a single primitive.

Meta-level behaviors capture these higher-level structures as compositions of the basic primitives. We currently limit primitive compositions to sequential compositions for simplicity, but superposition of primitives is not necessarily excluded. Meta-level behaviors consist of a subset of the primitives and the transition probabilities between them. Because meta-level behaviors are sequential, a single member primitive is active (producing motion) at a given time. Given the currently active primitive, a meta-level behavior provides the probability of transitioning to another primitive.

Our behavior vocabularies are not the only means for defining the basic capabilities for an autonomous humanoid agent, as described in Chapter 2. However, our definition of a behavior vocabulary has several advantageous properties that arise from being exemplar-based. First, behaviors in such vocabularies can be easily modified by adding or removing exemplars, editing existing exemplars, or repositioning exemplars in parameter space. Second, behavior vocabularies can be created manually, as in Verbs and Adverbs, or derived automatically from human motion, as described in this chapter. Furthermore, the representation of a behavior vocabulary is amenable to manual or automatic refinement.

Our perspective towards creating behavior vocabularies is to combine the benefits of automated processing and human intuition. As stated previously, our viewpoint is that "target" behaviors are "compiled" automatically using motion as "source" as an initialization for manual refinement. For the remainder of this chapter, we define the following structures for behavior vocabularies and coordinate spaces for representing motion segment data:

Definitions of structures used for the automated derivation of behavior vocabularies:

- A *behavior instance* or *exemplar* is a motion trajectory in the joint angle space of the humanoid agent that is a specific variation on an underlying behavior.

- A *primitive feature group* is a group of exemplars defining a primitive behavior.

- A *primitive behavior* is a family of trajectories in joint angle space defined by a configuration of a primitive feature group in a parameterization space.

- A *primitive forward model* is a realization of a primitive behavior as a nonlinear dynamical system. More specifically, as flowfield in joint angle space.

- A *meta-level feature group* is a union of exemplars from one or more primitive feature groups indicative a higher-level composite behavior.

- In general, a *meta-level behavior* is a mechanism for composing primitive behaviors through sequencing and/or superposition representative of a higher-level behavior. For this dissertation, meta-level behaviors are restricted to sequencing and realized as transition probabilities between member primitives.

Specification of spaces used to represent data in PDBV

- *Joint angle space* is the space of possible kinematic posture configurations for the humanoid agent. Each axis of the joint angle space is a degree-of-freedom (DOF) for the agent.
- *Input space* or *segment input space* is the space of possible trajectories in joint angle space with l frames.
- An *embedding space* is a nonlinear subspace of the input space produced through dimension reduction, ST-Isomap in particular.
- A primary embedding space or *primitive-level embedding space* or 1^{st}*-level embedding space* is produced directly from the input space. Without further specification, an embedding space is assumed to be a primitive embedding space.
- An *exemplar space* or *adverb space* is specific to a cluster in an embedding space, defining the mapping of the cluster between the embedding and input spaces.
- A *sampling space* is a subspace of an exemplar space from which samples are produced for evaluating the cluster mapping of the exemplar space.
- Secondary embedding spaces or *meta-level embedding spaces* are iteratively produced from lower-level embedding spaces. More specifically, an m^{th}*-level embedding space* is produced from the $(m-1)^{th}$*-level embedding space*.

4.2 Motion Performance Preprocessing

The first step in PDBV is the preprocessing of an input motion to produce a data set of motion segments with constant dimensionality. Motion preprocessing consists of segmenting the input motion followed by time normalization. Motion segmentation can be performed manually or automatically. We present *z-function segmentation* [44] and *Kinematic Centroid Segmentation* [72] as heuristic methods for automatically segmenting free-space motion (i.e., without external object or environment interactions).

The result from any segmentation method is a set of N_s segments of various lengths l_i. The dimensionality of the i^{th} segment is $d_i = l_i \times N_{dof}$, where l_i is the length of the segment i and N_{dof} is the number of regarded performer DOF. Dimension reduction techniques considered for this work require data points of equal dimensionality. Thus, we normalize these segments to a constant length l. Time normalization is performed by constructing a cubic spline for each segment and interpolating for l uniformly spaced samples.

4.2.1 Manual Segmentation

In manual segmentation, a human user segments the input motion manually through visual inspection. This method usually provides the best segmentation due to human common sense and judgment. However, this human intervention introduces human inconsistency into the segmentation and requires significant time and effort. Thus, manual segmentation was used only for comparing the performance of the ST-Isomap in estimating underlying structure.

4.2.2 Z-function Segmentation

Z-function Segmentation is an automatic segmentation mechanism for detecting *strokes* [120] in an input motion. Intuitively, z-function segmentation serves as a "stop detector", identifying frames indicating transitions from periods of motion to periods of inactivity and vice versa. As described by Fod et al. [44], the z-function is the sum of squares of the velocities of the kinematic degrees of freedom (DOF),

$$z = \sum \dot{\theta}_i^2 \tag{4.1}$$

After computing the z-function for a motion, a threshold is applied to approximate when the motion of the performer has stopped. This thresholding is beneficial for finding segments of discrete "point-to-point" motion. For example, Peters and Campbell [107] have successfully applied z-function segmentation to motion data collected from the NASA Robonaut actuated through teleoperation. For motion that does not stop or slow down significantly, however, the z-function is not capable of detecting any segments.

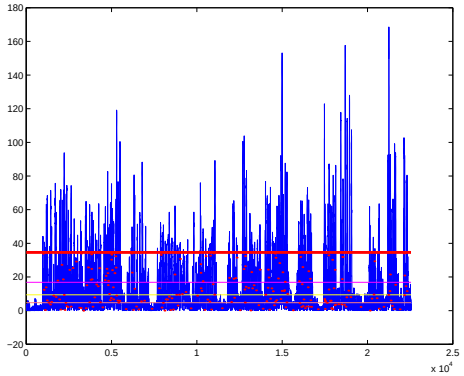


Figure 4.2: *Z-function segmentation of a motion stream. The value of the z-function is plotted over time. Horizontal lines are various thresholds considered based on proportions of the maximum, mean, and median value of the function. The thick line, representing 2 times the mean of the function, was used as the threshold. Dots indicate segment boundaries based on this threshold.*

4.2.3 Kinematic Centroid Segmentation

Jenkins and Matarić [72] propose Kinematic Centroid Segmentation (KCS) as an automatic segmentation method suited to detecting significant movements of a limb. Based on assumptions similar to those of Cutting and Profitt [32], KCS performs "swing detection"

by treating a kinematic substructure of a performer as a pendulum and placing segment boundaries at the beginning and end of pendulum swings. A kinematic substructure is a set of DOF that are coordinated for a common purpose. Two substructures can be working in coordination or independently, but DOF of a single substructure are always in coordination. For instance, the upper body could be separated into three kinematics substructures: left arm, right arm, and torso/head.

Because the DOF in each kinematic substructure are in coordination, we abstract the motion of the substructure into meaningful *marker features*. For segmentation, we use marker features for a *base marker*, the Cartesian location of a base joint, and a *centroid feature*, the centroid of the Cartesian joint locations in the substructure. For instance, an arm substructure could have a base marker as the location of the shoulder and a centroid feature as the average position of the shoulder, elbow, wrist, and hand.

KCS segments one kinematic substructure at a time in a greedy fashion. In segmenting for a single substructure, we first subtract the position of the base from the centroid for each frame, providing a rough configuration of the substructure in the coordinates of the substructure. Starting from the first frame, we compute the distance between centroid locations at the current frame and all subsequent frames. Using the centroid distance function, we start at the current frame and traverse forward in time until a minimum threshold distance is reached. At this point, we continue traversing forward until a local maximum is reached. At this local maximum, we place a segment boundary, set the current frame at the boundary, and repeat the procedure. Once segmentation is performed for a single substructure, segmentation for the next substructure is performed

on each individual segment. Additional substructure iterations perform segmentation individually on the existing set of segments.

The rationale for segmenting in this manner is that the revolute joints and joint limits of a human restrict the reachable space of the substructure. Due to these constraints on the substructure, the distance it can achieve from its initial position is bounded. Thus, a substructure that moves away from its initial position must eventually move back toward its initial position. The end of a "swing" is detected when the substructure reaches a locally extreme distance from its initial position. As we will discuss in Chapter 5, this segmentation method is not perfect, but is sufficient for use with ST-Isomap towards uncovering structure in our motion data.

Summary of Procedure for Kinematic Centroid Segmentation:

1. Set current segment to the first frame
2. Compute distance between centroid at current segment and the centroid at every subsequent frame
3. Find first local maximum in centroid distance function
   (a) Traverse frames until the distance exceeds a threshold
   (b) Traverse frames with a moving window until the current frame is the maximum value in the window
4. Place new segment at the found local maximum, go to step 2

The kinematic centroid distance function is specified as:

$$m_d(t) = D_m(m_{centroid}(t) - m_{base}(t), m_{centroid}(t_a) - m_{base}(t_a)) \tag{4.2}$$

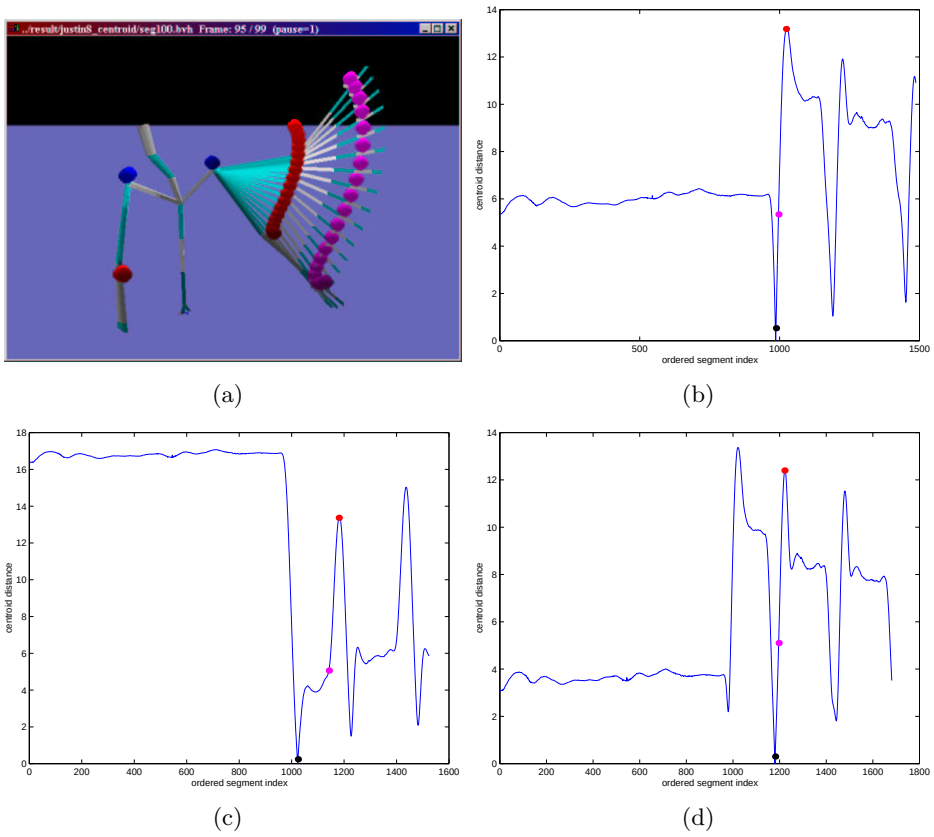


Figure 4.3: *Kinematic centroid segmentation for one segment of a motion stream. (a) A visualization of the motion being segmented with the spheres indicating the trajectories of the shoulders (blue), kinematic centroids (red), and end-effectors (magenta) of the right and left arms. (b-d) Plots of the offset distance for the kinematic centroid of the right arm for three motion segments. The points on these plots indicate the beginning of the current segment, passing of the "large motion" threshold, and placement of the next segment boundary, respectively.*

where $m_d(t)$ is the value of the centroid distance function at time t, t_a is the time index of the current segment boundary, D_m is the metric for computing distances between centroids, $m_{centroid}$ and m_{base} are the Cartesian positions of the centroid and base for the kinematic substructure at a given time.

We can express the placement of the next segment boundary using the current segment boundary by finding the next local maximum in m_d:

$$s_{i+1} = t_b \mid t_b > t_a; m_d(t_b) - m_d(t_a) \geq \tau; m_d(t_b) \geq m_d(t_s), \forall t_b - \epsilon \leq t_s \leq t_b + \epsilon \qquad (4.3)$$

4.3 Grouping Primitive Behavior Exemplars

For deriving primitive behaviors, we use ST-Isomap to transform a data set of motion segments into clusterable feature groups that are representative of modules of a common theme. We consider each motion segment as a point in a $N_{DOF} \times l$ dimensional *segment input space*, where N_{DOF} is the number of DOF and l is the number of frames in a motion segment. For instance, a motion segment of 20 DOF containing 50 frames is a point in a 1000-dimensional input space.

By applying sequentially segmented spatio-temporal Isomap, points in a high-dimensional segment input space should be transformed into a low-dimensional embedding space reflecting the underlying spatio-temporal structure of the segments. From the discussion of spatio-temporal Isomap in Section 3.3, we expect points with hard spatio-temporal correspondences to be placed in clusterable proximity within an embedding. More specifically,

we expect motion segments that exemplify an underlying behavior B to be in clusterable proximity, as well as temporally adjacent exemplars of behaviors A and C, such that the embedding reflect the $A \rightarrow B \rightarrow C$ spatio-temporal structure of the input motion. Such embeddings are produced by setting c_{CTN} to some significantly large value, adjusted through manual tuning, and c_{ATN} to 1.

Several techniques exist for clustering data, including K-means [10], mixture models [10], or hierarchical agglomeration [68]. Many of such techniques require the cluster cardinality to be specified *a priori* (i.e., the number of clusters to be known). However, the number of clusters present in set of motion data or reflected in the embedding is likely to be unknown. Other clustering techniques have also been developed to estimate the number of clusters in a data set.

Because spatio-temporal Isomap embeds points belonging to the same cluster into proximity, these clustering techniques are not necessary because intra-cluster points are separable in the embedding. Instead, "sweep-and-prune" clustering is used to find clusters separable by axis-aligned bounding boxes. Originally proposed by Cohen et al. [28] for detecting approximate collisions between geometries, sweep-and-prune clustering iteratively projects the data onto each axis of the embedding. The data points at each axis are sorted and partitioned based on a threshold separation distance ϵ_{SAP}. Using an assumed separation distance instead of an assumed cluster cardinality, the number of clusters can be estimated automatically. The procedure for sweep-and-prune cluster is as follows:

Procedure for One-Dimensional Sweep-and-Prune:

1. begin with one list of all points and $p = 1$
2. for all lists
   (a) sort the list by the p^{th} dimension
   (b) for all elements of the list
      i. place a marker between the j^{th} and $(j+1)^{th}$ elements if their difference exceeds a threshold
3. construct a new set of lists based on the markers

Once clustering is performed, the points of a single cluster are identified as a primitive feature group. The purpose of the dimension reduction and clustering is to extract features from the segment data indicative of behaviors. The features extracted by this process are groups of motion segments, or primitive feature groups. Because of their spatio-temporal correspondence, motion segments in each group are assumed to be a specific instantiation or exemplar of an underlying primitive behavior. The spatial signature of exemplars in a group may vary significantly in the segment input space. Together, however, the exemplars of a group are indicative of an infinite span of variations that are structurally common to a single behavior.

In addition to the feature extraction, a variation of the manifold assumption can be introduced into the resulting embedding. Temporally adjacent points may be separated into different clusters in the embedding, however these data pairs retain a connecting temporal relationship. Structurally, temporal relationships can be seen as a temporal connection between two primitive behaviors. By placing these connections between data pairs, the embedding takes on the form of a 1-manifold indicative of relative timing between primitives. Taking a kernelized HMM perspective, described in Section 3.3,

temporal relationships are structure-indicative transitions from one primitive to another. From this view point, primitive feature groups are hard latent variables discretizing the motion segments into states. A HMM without observation probabilities can be built from transition probabilities produced from normalized transition counts and initial state probabilities as fractions of cluster populations over the greater population. A spatio-temporal manifold in the embedded space from the HMM perspective can be viewed as a set of nodes connected into a 1-manifold by transition-weighted edges. The temporal relationships between primitive feature groups is further leveraged in Section 4.5 for uncovering meta-level feature groups.

4.4 Generalizing Primitive Feature Groups

Primitive behaviors in PDBV are similar to and motivated by verb behaviors proposed by Rose et al. for Verbs and Adverbs motion vocabularies [116]. Similar to primitive behaviors, verb behaviors are defined by a set of exemplar motion trajectories and are generalized through interpolation. As paraphrased from [85], interpolation is defined as finding a function F that maps from one space G to another space H given an equal number of corresponding points in the two spaces. For both primitive and verb behaviors, points in a lower dimensional *exemplar space* or *adverb space* are constructed with 1-1 correspondences to motion segments in the input space. The specific interpolation mechanism mapping between the input and exemplar is methodologically irrelevant.

Structurally similar motions that are not explicitly represented can be constructed in the input space by selecting and interpolating a non-exemplar point in the exemplar

space. Thus, an infinite number of motion variations can be produced for a given behavior. The span of these variations, or *support volume*, however, is dependent on the particular mechanism used for interpolation. For example, Shepards interpolation [127], used in our implementation, is restricted to interpolating samples within the convex hull of the points in exemplar space.

Unlike primitive behaviors, verb behaviors enjoy a manually defined and human intuitive parameterization. For a verb behavior, exemplars are manually positioned in exemplar space such that the parameterization of the behavior is intuitive to humans. Consequently, a human user will have an intuition about motion that will result from the selection of points in various regions of a verb behavior's exemplar space. This intuitive parameterization serves as a means for indexing into the behavior and avoiding complicated or exhaustive searches for desired motion. As Rose et al. have shown [116], desired motion can be returned from verb behaviors through *lazy evaluation* at run-time. Lazy evaluation, in the context of behaviors, means that mapping from the exemplar to the input space is not required to be explicitly known, but rather evaluated when needed.

Due to their automated derivation, primitive behaviors are not guaranteed to have exemplar space parameterizations that correspond to meaningful human semantics. The basic exemplar space, assumed for this dissertation, is to use the locations of exemplars from a primitive feature group in the spatio-temporal Isomap embedding, although exemplars can be repositioned manually or automatically by some other technique. Our speculation is that exemplar placement in primitive feature groups may have some spatial meaning because their relative distances have been preserved, only scaled by a constant factor to form CTN components. Even with this speculation, however, primitive behav-

iors will not necessarily have an intuitive parameterization. To uncover useful parameterizations, we employ *eager evaluation* to approximate an explicit mapping between the exemplar space of a primitive behavior and resulting motion trajectories in joint angle space. For eager evaluation, we assume that motion trajectories produced by a primitive behavior actually lie within a low-dimensional manifold in joint angle space, or *primitive motion manifold*. Eager evaluation without this assumption is likely to be infeasible. This infeasibility, in its worst case, stems from the number of samples required to represent such a mapping increasing exponentially with manifold dimensionality.

In our approach to eager evaluation, the exemplar space of a primitive behavior is densely sampled and interpolated to produce a dense sampling of its primitive motion manifold. For sampling an exemplar space, we experimented with sampling within bounding axis-aligned shapes (e.g. spheres and boxes) and kernels about each exemplar. However, these methods were susceptible to sampling in regions outside the support volume, as for axis-aligned shapes, or not placing samples in areas of interest, as for sampling kernels. Our best results for sampling were within a hyperellipsoid fit to exemplars using PCA. The orthonormal PC axes of the ellipsoid form a *sampling space* within an exemplar space. Using the sampling space, a primitive behavior is eagerly evaluated by *i)* generating N_{SAM} random samples within the sampling space, *ii)* reconstructing samples in the exemplar space, and *iii)* interpolating the samples into the input space. The mapping produced through eager evaluation is not completely explicit, but a span of variations should have sufficient representation by the samples to provide meaningful indexing. As a note, eager evaluation mappings are specific to a given configuration of

exemplars. In modifying an exemplar space, a current mapping would be invalidated and eager evaluation would be required to reestablish the mapping.

An explicit mapping of a primitive behavior between exemplar and input spaces allows for the behavior to be quickly "looked up" or indexed at run-time for an appropriate or desired motion. Because the span of variations for a primitive has been densely sampled, a primitive is likely to have already interpolated a motion variation that approximates a motion with desired properties. Using primitives and eager evaluation mappings derived by PDBV, Erol et al. [41] augment samples from primitive behavior mappings to be indexed with ending Cartesian coordinates of the hand. With this indexing structure, reaching behaviors for a humanoid robot can be automatically derived and quickly indexed for motion that will result in the robot reaching to a given location.

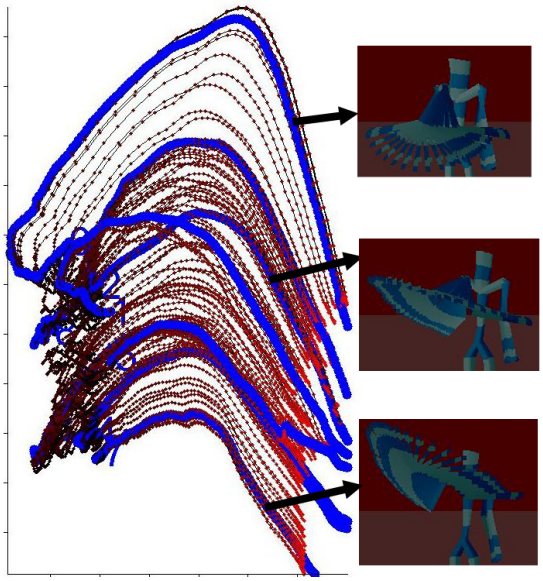


Figure 4.4: *A primitive flowfield for a horizonal arm waving behavior. The flowfield moves forward from black to red, with exemplars in blue. Motion for selected exemplars of this primitive are shown.*

While "lookup" is a useful feature for primitive behaviors, the explicit mapping of the exemplar space allows for the nonlinear dynamics of the behavior to be uncovered in joint angle space. By densely evaluating a primitive behavior, we have densely sampled the span of motion trajectories for the behavior in input space and a primitive motion manifold in joint angle space. The primitive motion manifold is a set of points in joint angle space, with some points connected by the sequential ordering of their respective trajectories. Given their sequential ordering, each point on the manifold provides a *gradient* or direction of displacement from its location. This gradient is not only an expression of the temporal behavior of a trajectory at a certain location but also the primitive motion manifold and its underlying primitive behavior. With trajectory gradients at each point, the primitive motion manifold forms a *flowfield* in joint angle space describing the nonlinear dynamics of the primitive behavior. The ability to describe its dynamics allows a primitive behavior to predict its future location in joint angle space.

We utilize a primitive flowfield for prediction in the context of dynamical systems. Using Jordan's description [73], a dynamical system is described by a next-state equation and an output equation:

$$x[n+1] = f(x[n], u[n])y[n] = g(x[n]) \tag{4.4}$$

where $x[n]$ is the kinematic posture or state at time n, $u[n]$ is the control input given at time n, f is a function producing the updated kinematic state at the next instance of time $x[n+1]$. Because observations occur in joint angle space, the output equation,

mapping the current state $x[n]$ to a current observation $y[n]$, uses an identity function for g.

A primitive flowfield provides a means to evaluate the function f for a given primitive. For dynamical evaluation, the closest neighbors in the flowfield to $x[n]$ are determined. Weighted by distance from the current state, the gradients of flowfield neighbors are summed and normalized to evaluate the linear direction for the gradient of $x[n]$. $x[n+1]$ is determined by stepping from $x[n]$ along its gradient by a length given by $u[n]$. In practice, the step speed will not change drastically between steps and is computed as the magnitude of the difference between the two previous kinematic states.

4.5 Deriving Meta-level Behaviors

Given the primitive behaviors found by PDBV in the preceding sections, we form meta-level behaviors. These are collections of primitive behaviors that are indicative of higher-level behaviors. Previously mentioned in Section 4.3, points and primitive feature groups found in the embedding space have a spatial and temporal relationship that forms an $A \rightarrow B \rightarrow C$ structure of modules. These temporal relationships allow the spatio-temporal structure of the motion to be viewed as a 1-manifold of behavior nodes with edges weighted by transition probabilities. In deriving meta-level behaviors, our intention is to collapse primitive behaviors with strong transitional connections $A \rightarrow B \rightarrow C$ into a single higher-level behavior ABC.

Similar to primitive behaviors, meta-level behaviors are derived through extracting features as *meta-level feature groups* using sequentially segmented spatio-temporal

Isomap. Meta-level feature groups are found within a m^{th}-*level embedding space*, where $m > 1$. We focus on feature extraction for the second-level embedding; however, the same approach can be used for higher level embeddings.

The second-level embedding is produced by applying spatio-temporal Isomap to the data in the first-level embedding. One slight difference for the second-level embedding is in determining local neighborhoods such that first-level feature groups are preserved. Once a feature group is clustered at any level, this feature is considered an indivisible unit for higher-level features. Thus, higher-level embeddings must *i)* preserve the integrity of lower-level features by retaining their clusterable proximity and *ii)* merge lower-level features with strong spatio-temporal relationships indicated by transitioning. Although irrelevant to the larger goals of PDBV, we expect lower-level features to have a finer degree of clusterable proximity than higher-level features. More specifically, a smaller clustering thresholding distance will find lower-level features in a higher-level embedding, whereas a larger clustering threshold will find higher-level features in the same embedding.

Lower-level feature groups are preserved in higher-level embeddings by constructing local neighborhoods to include only data points from a single feature group. These neighborhoods are constructed using an epsilon radius about each point based on the radius of the bounding sphere of their corresponding feature group. Because all data pairs in a neighborhood are CTNs, each neighborhood will preserve feature group integrity by forming a CTN component. The distances between points in this component will be reduced by c_{CTN}, forming a single unit that remains clusterably proximal.

With the integrity of lower-level feature groups addressed through local neighborhoods, lower-levels feature groups with strong temporal relationships can be collapsed

into higher-level feature groups. CTNs of one neighborhood will have ATNs in another neighborhood that are CTNs. By setting c_{ATN} to an appropriate value, data points in these neighborhoods can be placed in clusterable proximity. Consequently, the lower-level feature groups represented by these two neighborhoods will be placed into clusterable proximity in the embedding. Higher-level feature groups are then uncovered by sweep-and-prune clustering.

Feature group integrity and merging provides a means for meta-level embeddings to eventually converge. Feature group integrity ensures that sets of data points can split once associated through clustering. Thus, higher-level embeddings cannot devolve the features found at lower-levels. Feature group merging ensures that lower-level features are merged towards a more concentrated representation of the underlying spatio-temporal structure and behaviors. Eventually, connections between feature groups will be equally strong, no longer amenable to feature group merging at a higher-level, and unable to be split. At this level, the meta-level embeddings have converged.

A meta-level behavior is produced from a meta-level feature group by determining transition and initial state probabilities of its component primitives. For implementation purposes of this dissertation, a meta-level behavior is a composition mechanism of its component primitives indicative of higher-level sequential structure found in the input motion. The first step in constructing such a behavior is to determine its component primitives. A meta-level behavior can determine its component primitives by examining the primitives associated with members of its meta-level feature group. Once component primitives are found, the meta-level behavior recomputes initial state and transition probabilities with respect to its component primitives.

4.6 Summary

In this chapter, we presented Performance-Derived Behavior Vocabularies as our methodology for automatically deriving a capability repertoire for a humanoid agent. PDBV takes as input a single extended time-series of human motion and produces clustered exemplars and nonlinear dynamical systems for each uncovered behavior. The procedure for PDBV consists of a heuristic segmentation followed by sequentially segmented ST-Isomap. As described in Chapter 6, behavior vocabularies derived by PDBV can be used for functions such as motion synthesis, classification, and imitation.

Chapter 5

Evaluation

We evaluate our methodology to deriving behavior vocabularies through an implementation of PDBV. The PDBV system is empirically evaluated through the application of three multi-activity input motions. These motions contain a variety of activities, including dancing, reaching, and punching. Results from deriving behavior vocabularies from these input motions in various contexts are presented. We discuss the appropriateness of these results from manual observation, given *a priori* knowledge about the input motions. An analysis of the PDBV methodology is presented, discussing the issues, advantages, and shortcomings of our approach.

5.1 Implementation Description

The PDBV system used for this work is centered around a main PDBV function implemented in Matlab. The main PDBV function serves to *i)* read in an input motion, *ii)* perform motion preprocessing, *iii)* embed the data in spaces up to a given number m iterations, *iv)* extract feature groups in each embedding, and *v)* sample each primitive behavior for an explicit mapping of its motion manifold in joint angle space.

Input motion for the main PDBV function is read from two files assumed to have originated from a single Biovision BVH motion capture file. As described in [82], kinematic motion stored in the Biovision BVH format contains two sections. The first section, HIERARCHY, is a recursive specification of the kinematic hierarchy of the subject. The second section, MOTION, contains the data describing the motion of the kinematic subject. This information includes the number of frames, sampling frequency, and matrix of DOF values at each frame. For use with the PDBV system, a single bvh file is split into .hdr and .mot files for the HIERARCHY and MOTION sections, respectively. The PDBV system reads in the .mot file as input. The .hdr file is not used for input, rather it is used to create BVH files from motion trajectories produced by PDBV through concatenation.

Input motion, read as motion matrices from .mot files, is preprocessed through segmentation and time normalization. The main PDBV function looks for a .seg file specifying the intervals of the segments in the input motion. If it is not present, the .seg file can be created manually using a text editor or with automatic procedure. For automatic segmentation, the main PDBV function calls another Matlab function to perform segmentation and produce a .seg file. Two automatic segmentation functions were implemented for z-function segmentation and Kinematic Centroid Segmentation, described in Section 4.2. The main PDBV function partitions the input motion into N_s separate segment matrices based on the intervals read from the .seg file. Splines are constructed for each DOF in each segment matrix using Matlab's *spline* function. l (typically $l = 100$) uniform samples are extracted from each of these splines. The l samples from each DOF of a single segment are concatenated into a $l \times N_{DOF}$ vector. All segments are concatenated into an $N_s \times (l \times N_{DOF})$ matrix.

Given m from a user, the PDBV main function applies m iterations of sequentially segmented spatio-temporal Isomap to the segment matrix. The result from these iterations is m sets of N_s data points. Each set is a $N_s \times j_{DIM}$ matrix specifying the configuration of the motion data at the j^{th}-level embedding, where j_{DIM} is the selected dimensionality for the j^{th}-level embedding. Although not applicable in general, we have found that overestimating the dimensionality of the embedding spaces, 15 dimensions for primitive level and 5 dimensions for meta-levels tend to provide enough information for separable clustering. In addition to these parameters, we typically set c_{CTN} to 10 (primitive) and 30 (meta-level), c_{ATN} to 1 (primitive) and 500 (meta-level), primitive local neighbors to be the 5 nearest neighbors. The parameters are passed to our modified version of the standard Isomap Matlab function, provided by Tenenbaum et al. [131], to perform the embeddings.

The main PDBV function calls another Matlab function we have implemented to perform sweep-and-prune clustering on the data for embeddings at each level. The threshold cluster separation distance is selected as a small fraction (typically .01) of the diagonal of the bounding box of the embedded data. A separate structure is used to store the association of each data point to its feature group in each embedding space. Using these feature group associations, transition and initial state probabilities are computed for all feature groups and stored in a separate structure.

The final step taken by the main PDBV routine is to sample the exemplar space of each primitive behavior for motion trajectories in joint angle space. Sample spaces are constructed for each exemplar space as the three major PC of the exemplars found by Matlab's *svd* function. N_{sam} random samples are generated using Matlab's *rand*

function within a unit sphere about the origin. After their reconstruction in exemplar space, these samples are interpolated into the input space using our implementation of Shepards interpolation [127]. The N_{sam} motion trajectories produced from sampling are stored in a database file for each primitive behavior.

In addition to the Matlab routines, a animation viewer was implemented in Microsoft Visual C++ to view motion trajectories produced by PDBV vocabularies. Routines related to motion synthesis, classification, and imitation are described in Chapter 6.

5.2 Empirical Evaluation

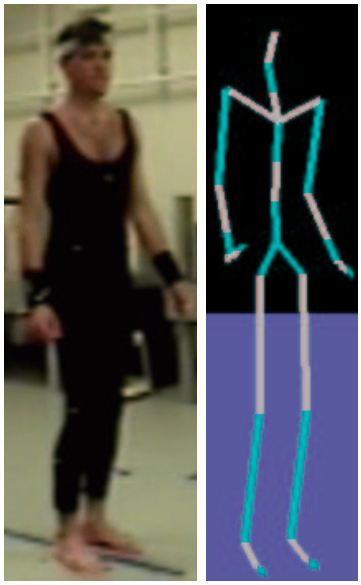
We empirically evaluate our PDBV implementation in several contexts with respect to three sets of input motion. PDBV is applied to each input motion in the following contexts: behavior vocabulary derivation, comparison with PCA and spatial Isomap, individual activity isolation, synthesized motion feedback, segmentation variation, and humanoid agent control.

5.2.1 Input Motion Descriptions

Each input motion[1], individually refered to as Input Motion I, was collected from a human subject performing a series of scripted activities centered around upper-body movement. The input motions were collected using a Vicon optical motion capture system. Input Motions 1 and 2 were scripted to contain multiple activities, including punching, dancing, and arm waving. Input Motion 3 was scripted to contain a single two-arm reaching

[1]Motion data used in this dissertation was graciously acquired and provided by Jessica Hodgins and her motion capture group at Carnegie Mellon University.

activity between a *zero-posture* and various Cartesian locations in the subject's reachable space. Input Motions 1, 2, and 3 consist of 22,549, 9,145, and 9,394 frames, respectively. The kinematics of the input motion consisted of 42 kinematic DOF, sometimes 48 DOF are used to include the global position and orientation of the subject with respect to the world. The original performer motion contained 69 DOF, but less relevent DOF were removed to avoid DOF weighting issues and minimize kinematic mismatches with a humanoid robot. The streams are available from *http://www-robotics.usc.edu/∼cjenkins/motionmodules/*.



(a) (b)

Figure 5.1: *(a) A snapshot of the human performer instrumented with reflective markers during the execution of a demonstration motion stream. (b) A visualization of the human performer's kinematics at the time of the snapshot using the post-processed motion stream.*

The input motions were segmented using three different methods, described in detail in Section 4.2. Each segment was normalized to 100 frames. The number of segments produced by each method is shown in Table 5.2. Using Input Motion 1 as our primary

example for discussion, we specify the scripting of behaviors (Table 5.1) performed in this input motion. The PDBV system has no direct effect on the segments produced, but instead uncovers motion groupings with respect to the underlying process.

Stream 1 Behavior Name	Frame Interval	Number of Segments
Circles	1-3350	23
Spider	3350-4390	10
Horizontal Waving	4390-5600	18
Semaphores	5600-7170	19
Vertical Waving	7170-9580	23
Horizontal Waving	9580-11150	16
The Egyptian	11150-12380	16
The Scuba	12380-12970	7
The Twist	12970-13690	16
The Cabbage Patch	13690-14770	11
The Robocop	14770-15340	11
The Monkey	15340-16750	25
Jab Punch	16750-18160	15
Uppercut	18160-19100	12
Roundhouse	19100-20730	12
Knockout Punch	20730-22549	13

Table 5.1: Script of performed activities for Input Motion 1. This scripts lists manually assigned descriptions of activities, interval of performance, and number of segments from manual segmentation.

5.2.2 Behavior Vocabulary Derivation Results

5.2.2.1 Grouping Exemplars into Features

In this section, we present results from deriving behavior vocabularies for each input motion. Through manual observation, the appropriateness of each behavior vocabulary is manually estimated with respect to scripting of activities for its corresponding input motion. Unless stated otherwise, KCS is used for input motion segmentation.

Results at the various stages in the PDBV procedure for Input Motion 1 are shown in Figure 5.2. Datapairs (of motion segments) within the same CTN component are visualized in the pre-embedding distance matrix as dark blue entries. Square submatrix blocks of CTN components can be visualized along the diagonal of the distance matrix. For activities with temporal relationships between two or more primitives, these blocks visualized as checkerboard-like patterns indicating the structural transitioning between CTN components. For example, the activity with alternating primitives visualize as an block checkerboard, as Figures 5.6 and 5.14. This checkerboard pattern results from the activity performing alternating transitions between "waving outward" and "waving inward" primitives.

As shown in Figure 4.4, spatial Isomap yields submatrix blocks similar to spatio-temporal Isomap. However, disconnected components form in spatial Isomap distance matrices due to disconnections between CTN components of different activities and spatial distances between CTN components of the same activity. These disconnected components prevent spatial Isomap from providing a single embedding global to all of the data points.

Shown in the distance matrices of Figure 5.2, distinct clusters are present in the primitive-level embedding based on the CTN components formed in the spatio-temporal Isomap distance matrix. Even though the primitive-level embedding is 15-dimensional, these clusters are visually distinguishable in the first two or three dimensions of the embedding. However, some of the clusters require more than three dimensions for separability to be visually apparent. From the partitioning by sweep-and-prune clustering, 78 primitive feature groups are found. Each feature group in the primitive-level embedding is color coded and shown with a bounding sphere in 3 dimensions. Lines are drawn

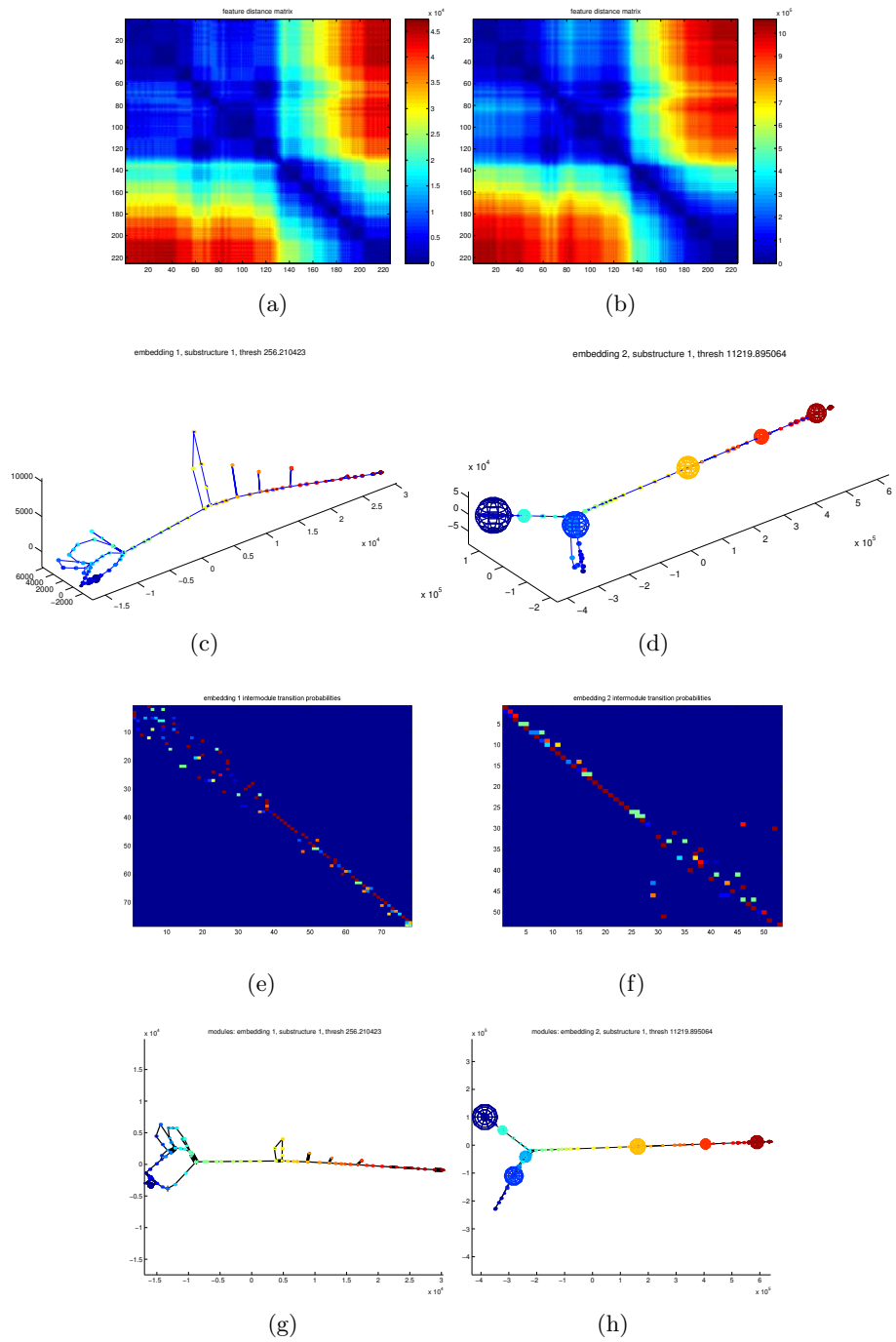


Figure 5.2: *Results of first (left column) and second (right column) level exemplar grouping for Input Motion 1. (a,b) Distance matrices produced by ST-Isomap. (c,d) Embeddings produced (blue line) and feature groups (color coded spheres) found by ST-Isomap. (e,f) Transition probabilities between pairs feature groups. (g,h) Feature groups connected by black lines (width indicating the number of transitions between a pair of feature groups).*

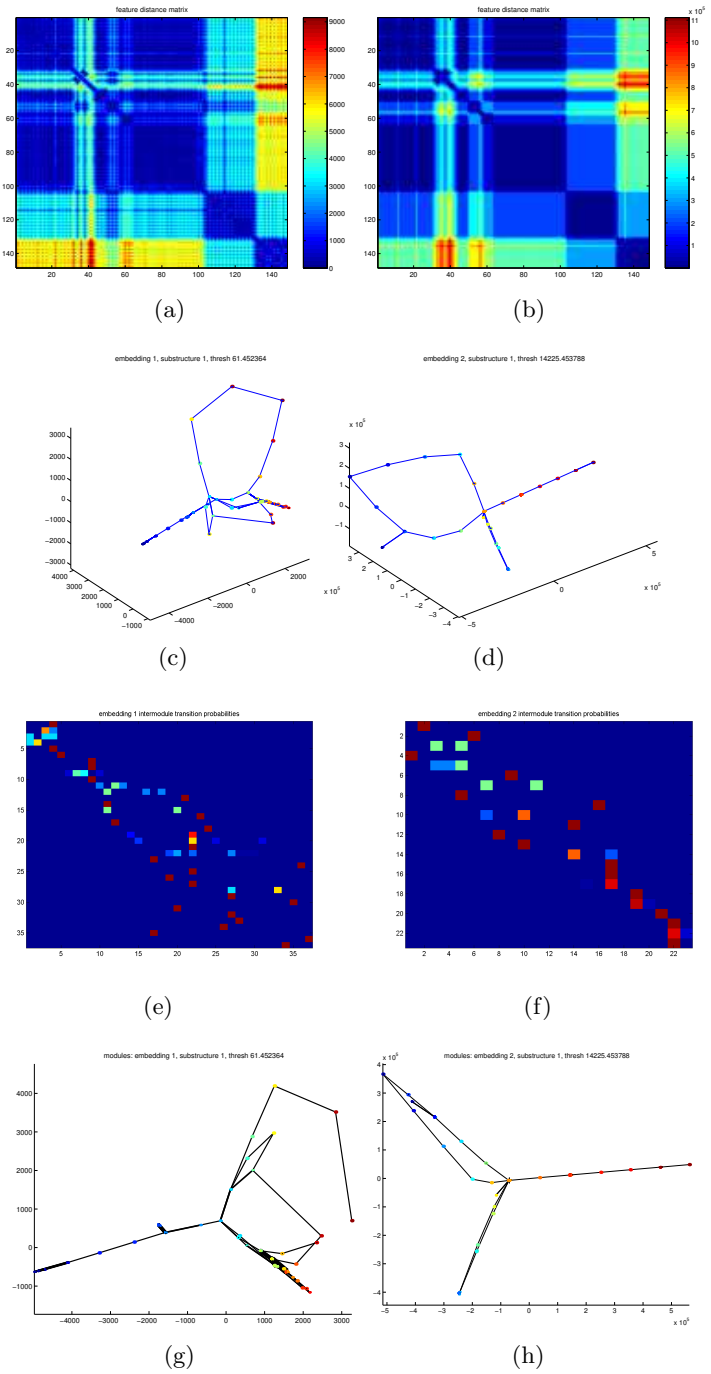


Figure 5.3: *Results of first (left column) and second (right column) level exemplar grouping for Input Motion 2. (a,b) Distance matrices produced by ST-Isomap. (c,d) Embeddings produced (blue line) and feature groups (color coded spheres) found by ST-Isomap. (e,f) Transition probabilities between pairs feature groups. (g,h) Feature groups connected by black lines (width indicating the number of transitions between a pair of feature groups).*

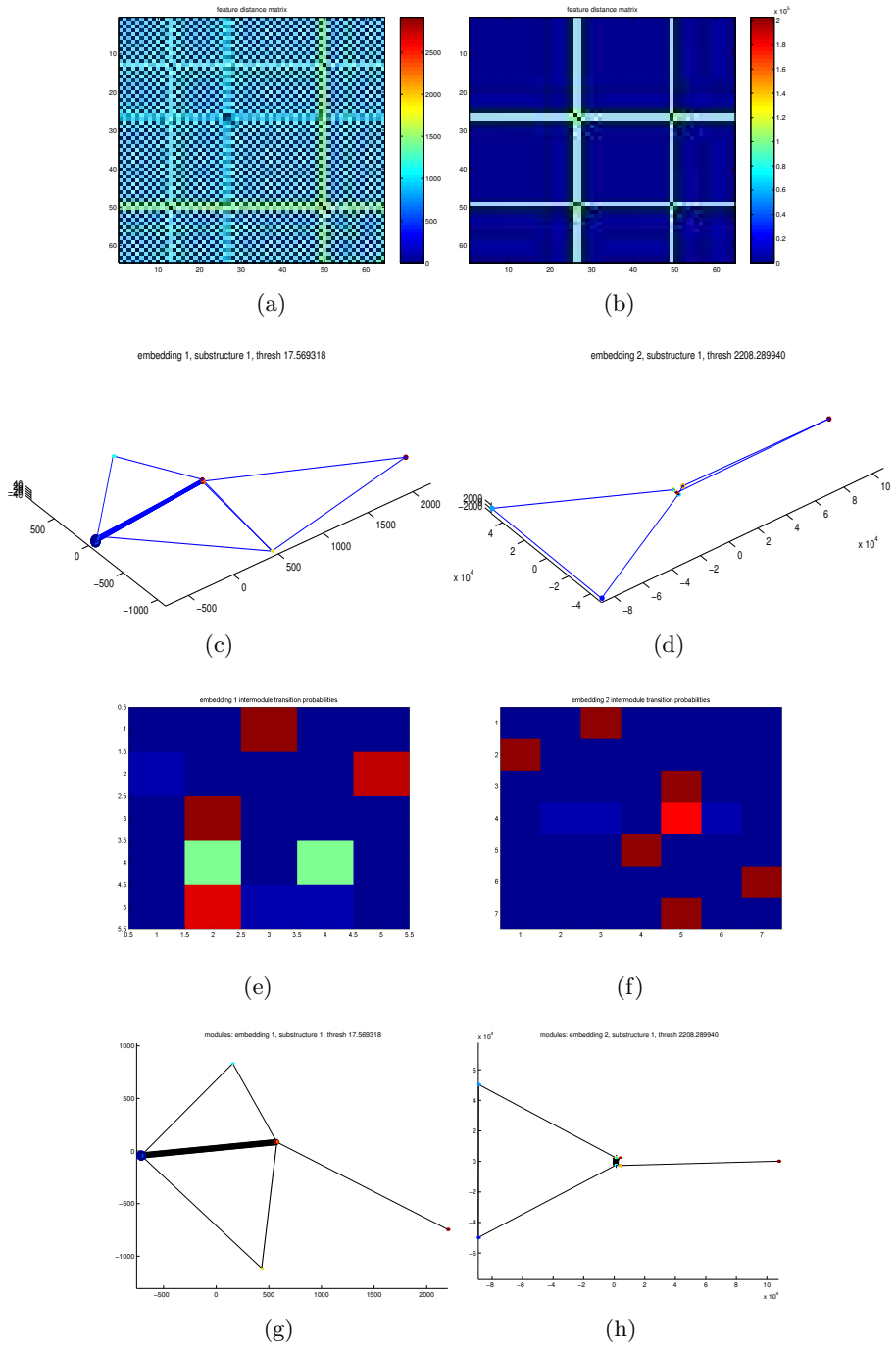


Figure 5.4: *Results of first (left column) and second (right column) level exemplar grouping for Input Motion 3. (a,b) Distance matrices produced by ST-Isomap. (c,d) Embeddings produced (blue line) and feature groups (color coded spheres) found by ST-Isomap. (e,f) Transition probabilities between pairs feature groups. (g,h) Feature groups connected by black lines (width indicating the number of transitions between a pair of feature groups).*

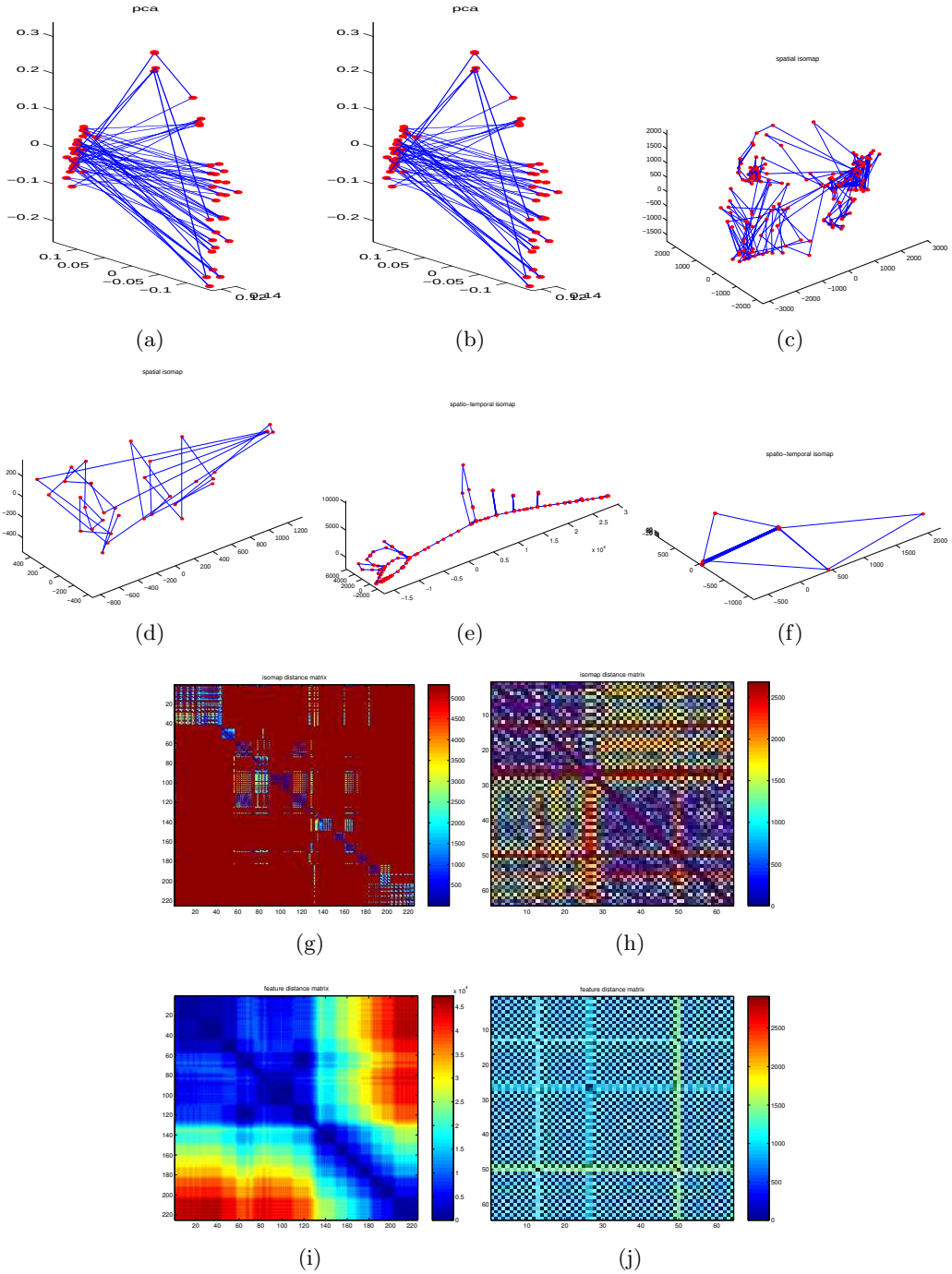


Figure 5.5: *3D embeddings of Input Motion 1 and 3 using (a,b) PCA, (c,d) spatial Isomap, and (e,f) spatio-temporal Isomap. Distances matrices for (g,h) the disconnected components of spatial Isomap and (i,j) the single connected component of spatio-temporal Isomap.*

between pairs of primitives that transition between each other, with thicker lines indicating a greater number of transitions between a pair. We interpret subsets of primitives connected by strong transitional relationships to be indicative of *structurally significant* behaviors. Primitives with weak inter-primitive transitions are considered to be *spurious*, but could be structurally significant behaviors that are underrepresented in the input motion.

We observed that the feature groups found at the primitive-level for all of the motion groups typically have a common theme. For example, Figure 5.6 highlights a primitive from an arm waving activity in Input Motion 1 that appropriately clusters 12 variations that are consistent with an underlying waving behavior. However, a number of primitive feature groups were observed to contain transitional or merging artifacts, highlighted in Figure 5.7. As described in Discussion section of this chapter, transitional artifacts are due to sparse numbers of transitions between performed activities and merging artifacts are due to inaccurate local neighborhood construction.

Similar to the primitive-level, meta-level feature groups form separable clusters in a second-level embedding. For Input Motion 1, these clusters are readily visualized in the second-level distance matrices and second-level embedding shown in Figure 5.2. Unlike the primitive-level, second-level feature groups were not subject to artifacts of improperly merged primitive-level feature groups; however, they inherit the inappropriate groups occurring at the primitive level.

In contrast to Input Motion 1, Input Motion 3 has an intuitively simple structure consisting of a single activity for reaching. The segmentation of Input Motion 3 produces segments that successively alternate between underlying behaviors for "reach to location"

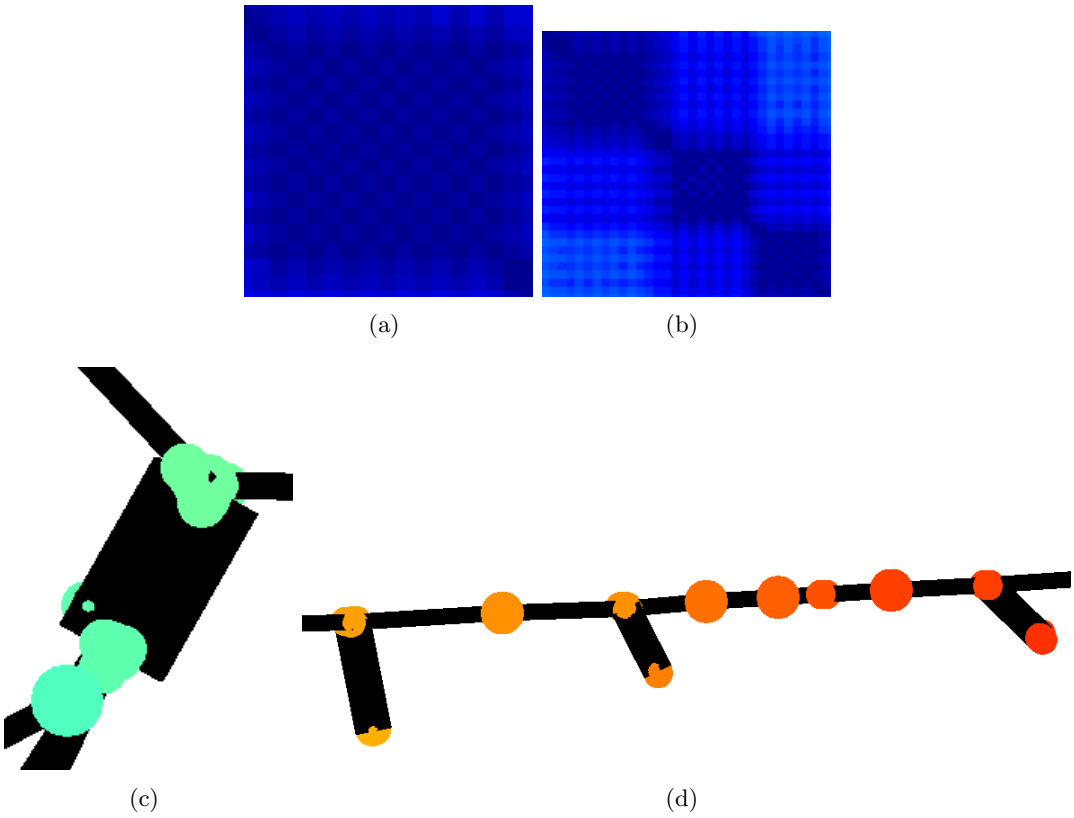


Figure 5.6: *Examples of meta-level behaviors consisting of two alternating primitives within Input Motion 1. (a) A "checkerboard" block of the distance matrix isolating horizontal arm waving and (c) two feature groups identified in the resulting embedding. (b) Checkerboard blocks of the distance matrix corresponding to dancing performances of "The Twist", "The Cabbage Patch", and "The Robocop" and (d) feature groups for these dances in the resulting embedding.*

and "return to zero posture". Structurally, these segments form two clusters with continual transitions alternating between these clusters. These clusters are separable in the input space and can be easily identified visually from the first 3 principal components produced by PCA, as in Figure 5.4. Even though clustering (with or without PCA) is sufficient for this input motion, it is notable that spatio-temporal Isomap provides PDBV with the ability to provide the same clustering result without *a priori* clustering

cardinality, shown in Figure 5.5. Furthermore, feature groups found in the second-level embedding show convergence, described in Section 4.5. Thus, the behavior vocabulary for Input Motion 1 consists of two primitive behaviors and one "root" meta-level behavior.

5.2.2.2 Primitive Eager Evaulation

For PDBV, we assume that a family of motion variations is a low dimensional manifold in the joint angle space of the capture subject. We are able to visualize these primitive motion manifolds by applying PCA and viewing its projection onto the first 3 PCs. Figures 5.7 and 5.12 illustrate primitive motion manifolds derived from Input Motions 1 and 3. These manifolds required no more than 3 PCs for accurate viewing and typically formed as bordered 2-manifolds in joint angle space. From these visualizations, we observed that eager evaluation through sampling provides a representative realization of a primitive's span of variation, even when using basic Shepards interpolation.

5.2.2.3 Meta-level Convergence

A convenient, but not essential, component of PDBV is convergence of feature groups at some meta-level. Convergence of feature groups is intuitively shown for Input Motion 3 in Figure 5.4. For this motion, two alternating primitive feature groups are found. The same two feature groups are also found at the second-level. More generally, all meta-levels will yield the same feature groups. This result indicates that the primitive-level contains the highest level of abstraction in the motion data, assuming an "abstract root" behavior.

For input with more complex underlying structure, such as Input Motion 1, feature group convergence becomes more difficult. PDBV continually collapses the 1-manifolds

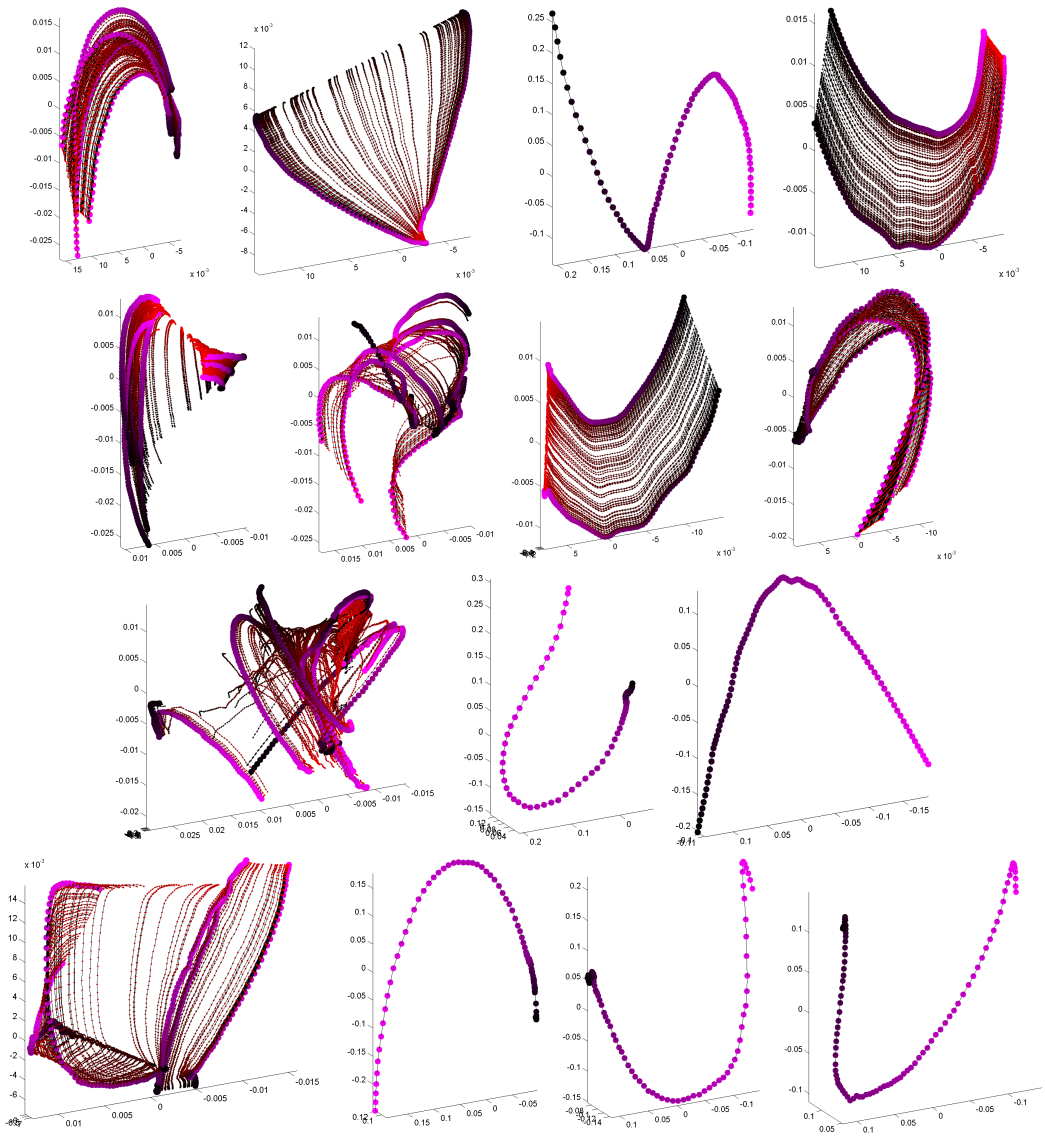


Figure 5.7: *Primitive flowfields 1 to 15 produced by PDBV for Input Motion 1. Each flowfield is shown as trajectories (moving from black to red) of kinematic configurations with respect to its first 3 principal components. Exemplar trajectories are highlighted with larger circular markers moving from black to magenta.*

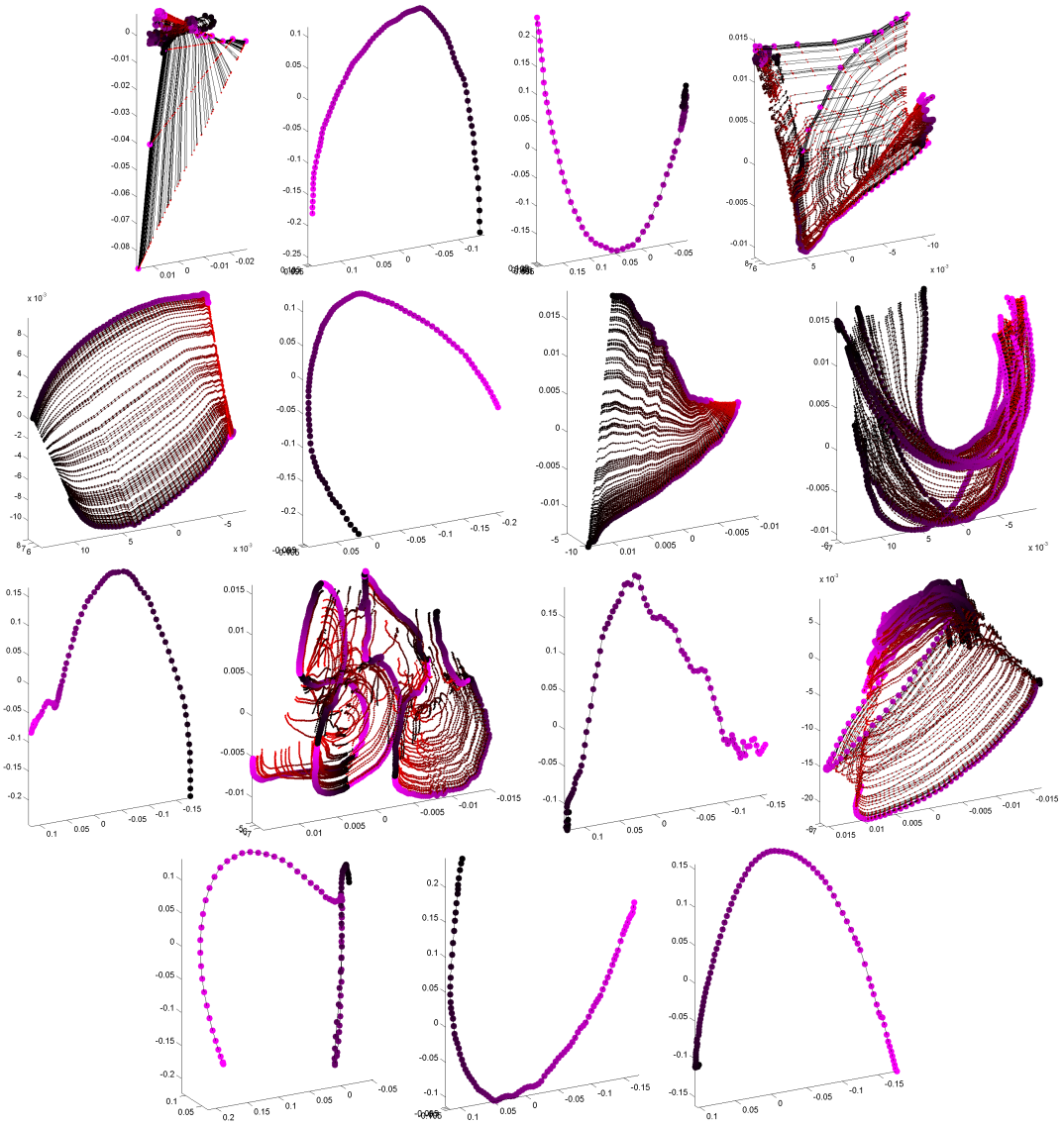


Figure 5.8: *Primitive flowfields 16 to 30 produced by PDBV for Input Motion 1. Each flowfield is shown as trajectories (moving from black to red) of kinematic configurations with respect to its first 3 principal components. Exemplar trajectories are highlighted with larger circular markers moving from black to magenta.*

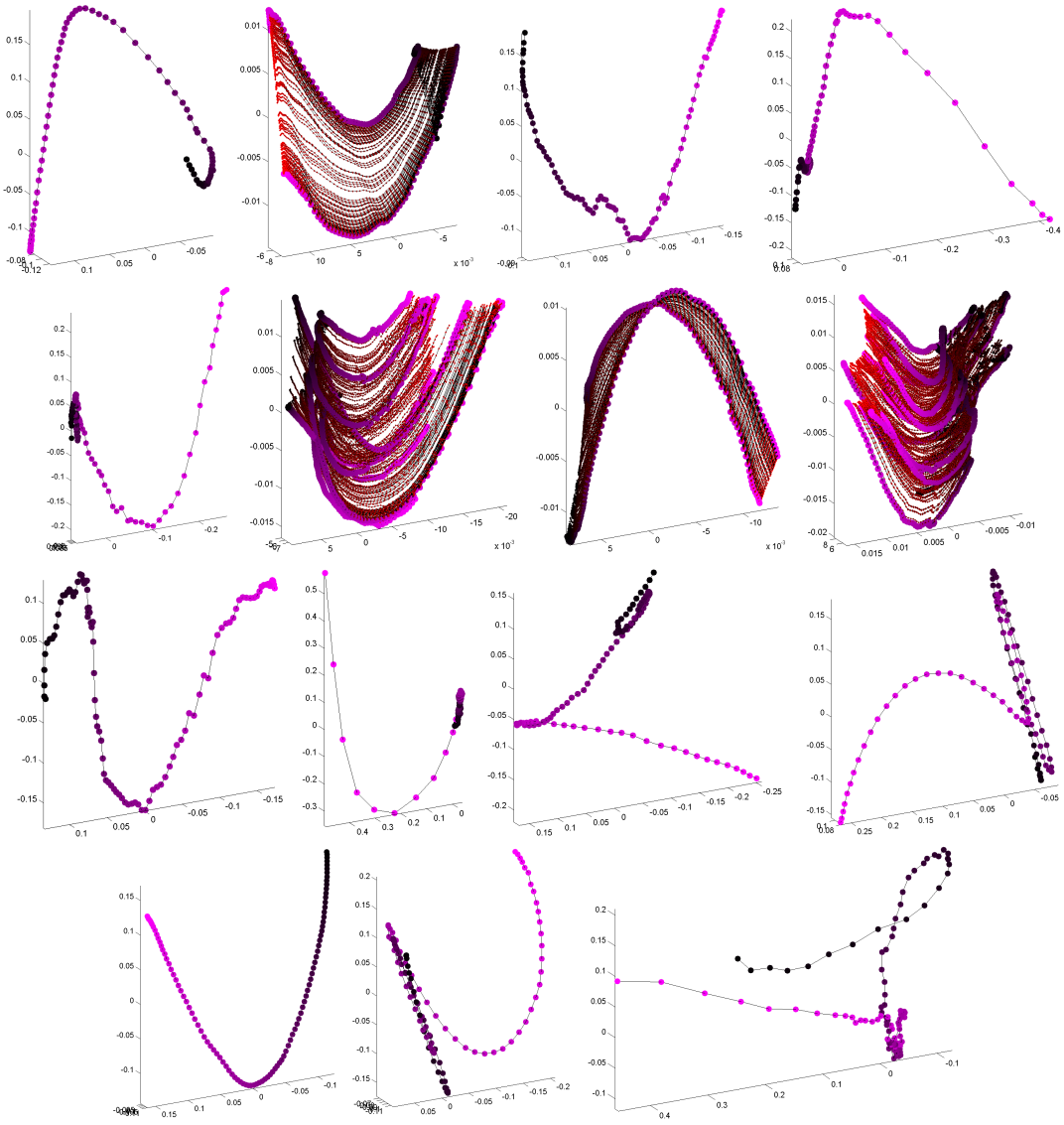


Figure 5.9: *Primitive flowfields 31 to 45 produced by PDBV for Input Motion 1. Each flowfield is shown as trajectories (moving from black to red) of kinematic configurations with respect to its first 3 principal components. Exemplar trajectories are highlighted with larger circular markers moving from black to magenta.*

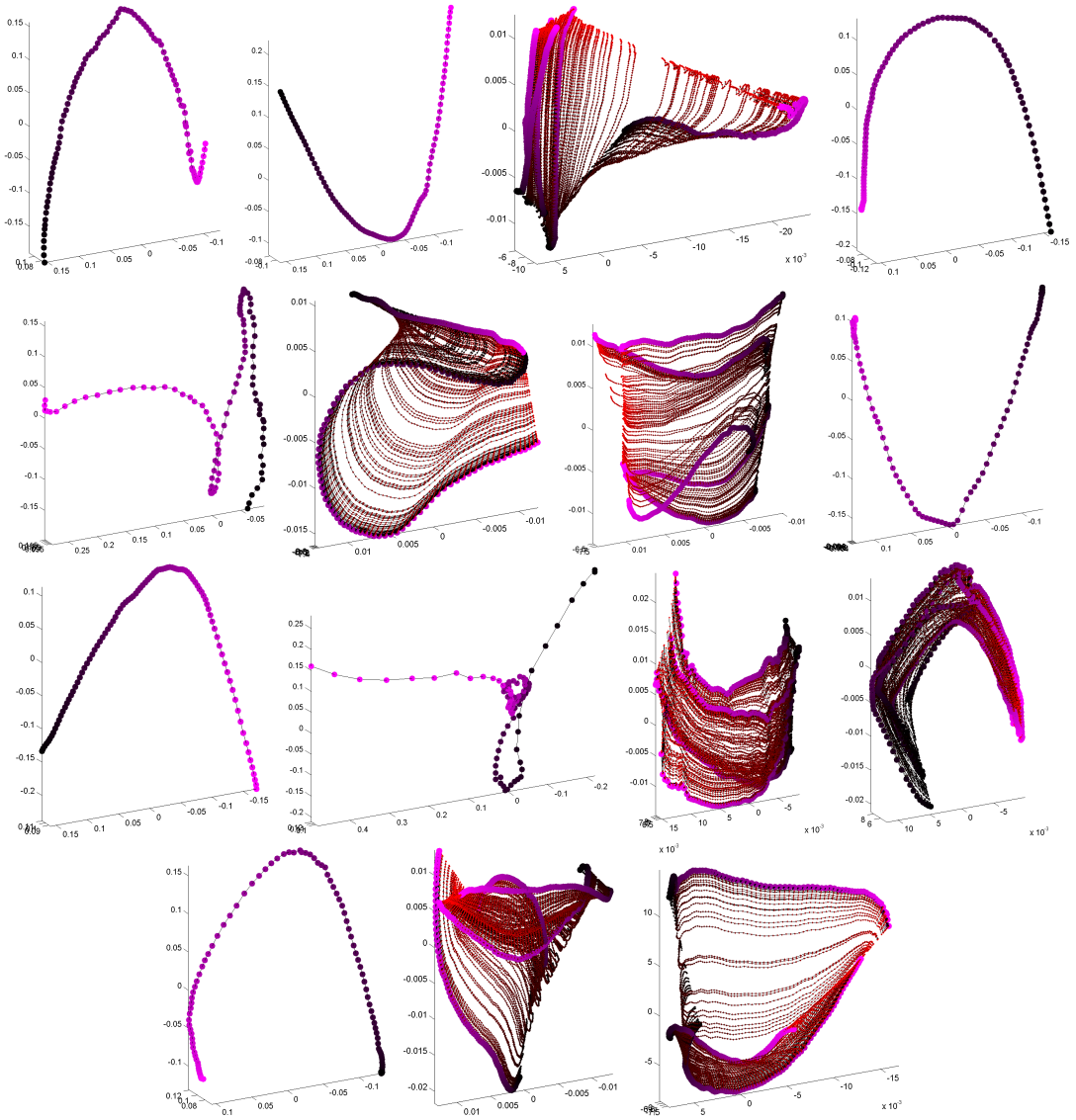


Figure 5.10: *Primitive flowfields 46 to 60 produced by PDBV for Input Motion 1. Each flowfield is shown as trajectories (moving from black to red) of kinematic configurations with respect to its first 3 principal components. Exemplar trajectories are highlighted with larger circular markers moving from black to magenta.*

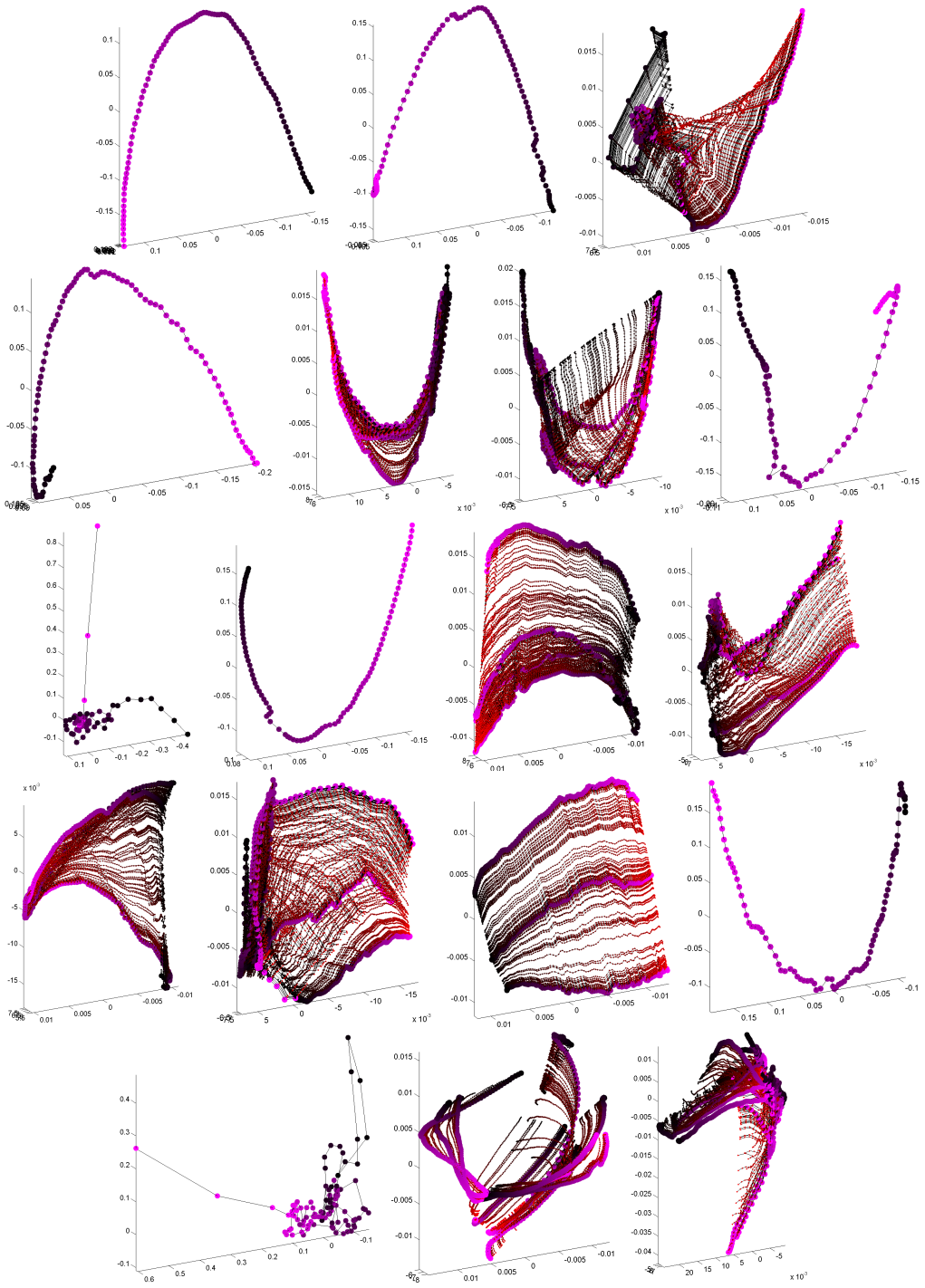


Figure 5.11: *Primitive flowfields 61 to 78 produced by PDBV for Input Motion 1. Each flowfield is shown as trajectories (moving from black to red) of kinematic configurations with respect to its first 3 principal components. Exemplar trajectories are highlighted with larger circular markers moving from black to magenta.*

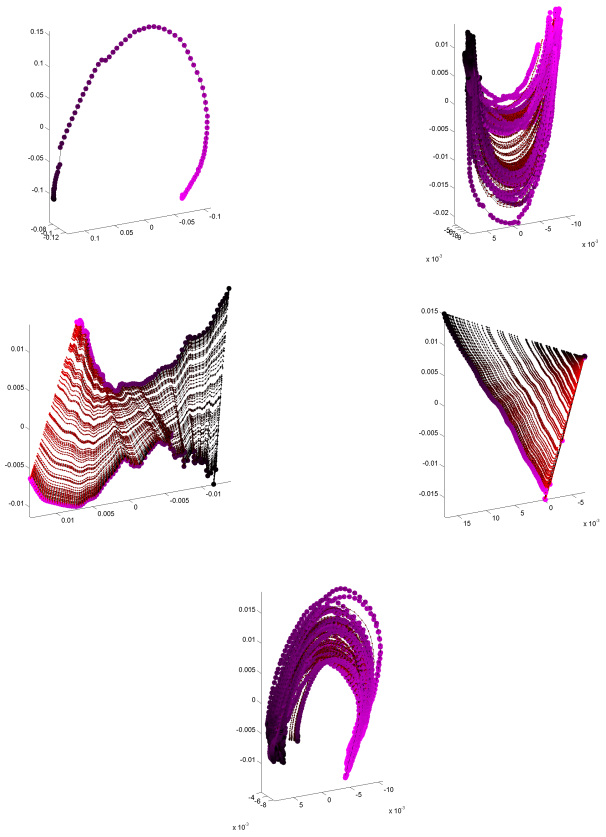


Figure 5.12: *Primitive flowfields produced by PDBV for Input Motion 3. Each flowfield is shown as trajectories (moving from black to red) of kinematic configurations with respect to its first 3 principal components. Exemplar trajectories are highlighted with larger circular markers moving from black to magenta.*

found in each embedding and, qualitatively, the cluster assignments for Input Motion 1 converge for practical purposes, as shown in the last subfigure of Figure 5.13. The problem for these motions, however, is ensuring members of a feature group from the previous embedding remain in same local neighborhoods for the next embedding. An epsilon radius or k-nearest neighbors could be estimated to retain these groups. Such globally applied settings are not likely to maintain feature group integrity due to varying amounts of space spanned by different clusters. In our implementation, directly specifying

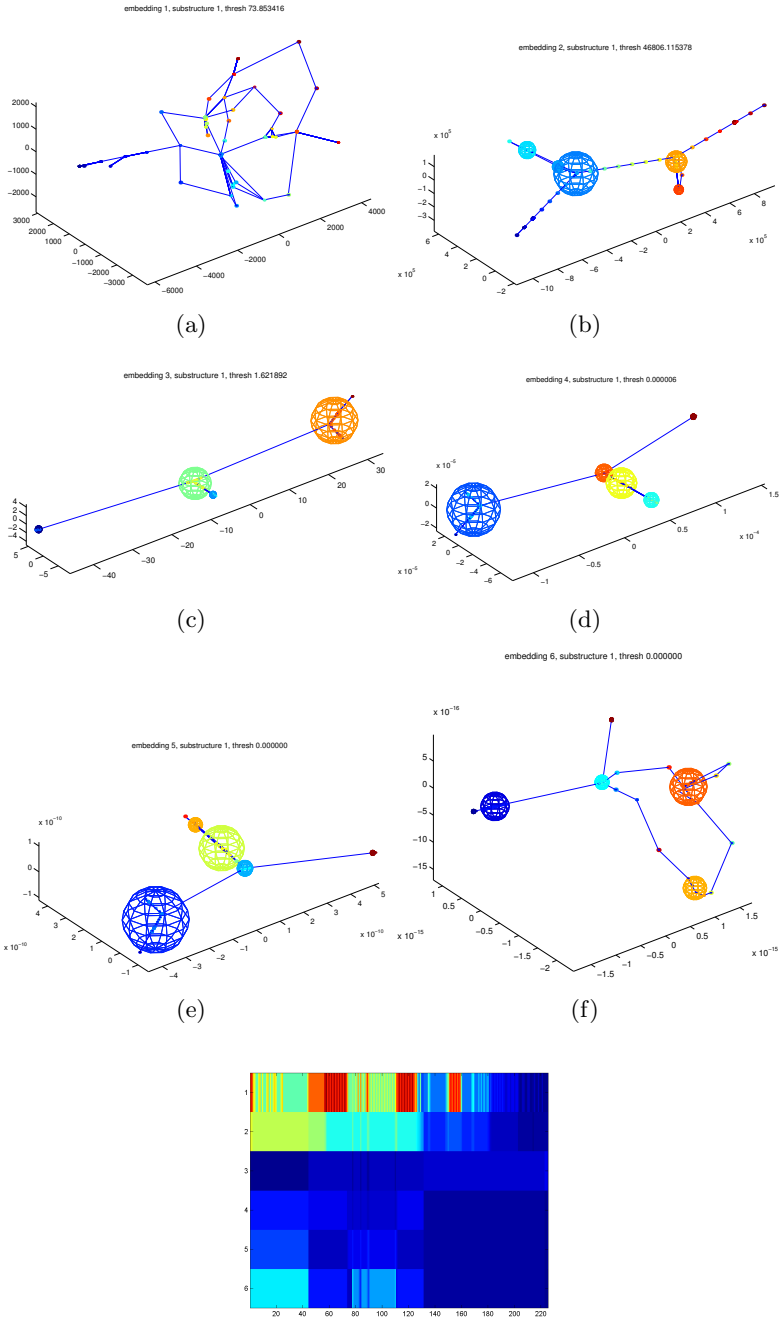


Figure 5.13: *(a-f) Feature groups clusters (as color coded spheres) from the first through sixth level embeddings of Input Motion 1 with global position and orientation information. Blue lines are placed between feature groups that are sequentially adjacent in the input motion. (g) Assignments of the segments from the input motion (on the x-axis) at each embedding level (on the y-axis) indicated by the color of each element.*

neighborhoods based on previous feature groups also proved to not guaranteed feature group integrity. Thus, meta-level convergence cannot be guaranteed by PDBV, at least through our current implementation. This fact is likely to be an artifact resulting from grouping oscillation of spurious motions and parameter tuning of c_{atn} and c_{ctn}.

5.2.3 Individual Activity Isolation

In applying PDBV, we expect multi-activity input motions to form structurally significant feature groups. Furthermore, motion of an individual activity in isolation should form feature groups consistent with behaviors derived for the same activity within the complete input motion. More generally, arbitrary truncation of the input motion should produce feature groups consistent with the input motion. Isolated activity consistency is dependent primarily on *i)* consistent segmentation of the input and isolated motions, *ii)* adequate density of motion segments for each underlying behavior, and *iii)* construction of appropriate local neighborhoods. Issues related to motion density and local neighborhoods are avoided for activities in isolation, but will arise for arbitrary truncation. These issues are discussed in detail within the Discussion section of this chapter.

We applied PDBV to various isolated activities from Input Motion 1 for comparison to feature groups from complete input motion. Intervals isolating individual activities in the input motion were manually specified. Activities for punching, arm waving, and hand circling were isolated. Feature groups resulting from the isolated activities for the "cabbage patch", "the twist", vertical waving, jabbing, and punching. These activities are structured by two alternating primitives, except for the punching motion. The punching motion is structured by five primitives, two for the jab and three for the uppercut punch.

As shown in Figure 5.14, PDBV was able to uncover these underlying primitives and not include spurious motions into the main feature groups.

5.2.4 Humanoid Agent Control

PDBV can be complimented with a task-level control mechanism for motion synthesis, as described in Section 6.1. For humanoid robot control, a behavior vocabulary can be synthesized off-line, as a trajectory of desired kinematic postures, or on-line, by resetting a desired posture based on continual prediction of primitives. Synthesized desireds are input into a chosen low-level control mechanism, such as a PD-Servo [31], for producing joint torques that actuate the motion.

Control of robots is one of the more obvious applications of a behavior vocabulary. We evaluated this potential application using a basic arbitration mechanism for setting agent desireds, described in Section 6.1. This mechanism performs a "random walk"-like traversal over the primitives, using the derived transition probabilities for arbitration. An active primitive produces control updates until nearing completion, at which time the arbitrator randomly transitions the activation to another primitive. Using this arbitrator, motion was synthesized in an incremental fashion for several derived vocabularies. The synthesis procedure came to completion to specified number of frames for horizontal arm waving (3000 frames), jabbing (5000 frames), and "cabbage patch" (20000 frames) vocabularies, shown in Figure 5.15. The synthesis process for these vocabularies were manually stopped, but could continue for a much longer duration. However, the punching vocabulary, for jab and and uppercut punches, the synthesis procedure was prone to

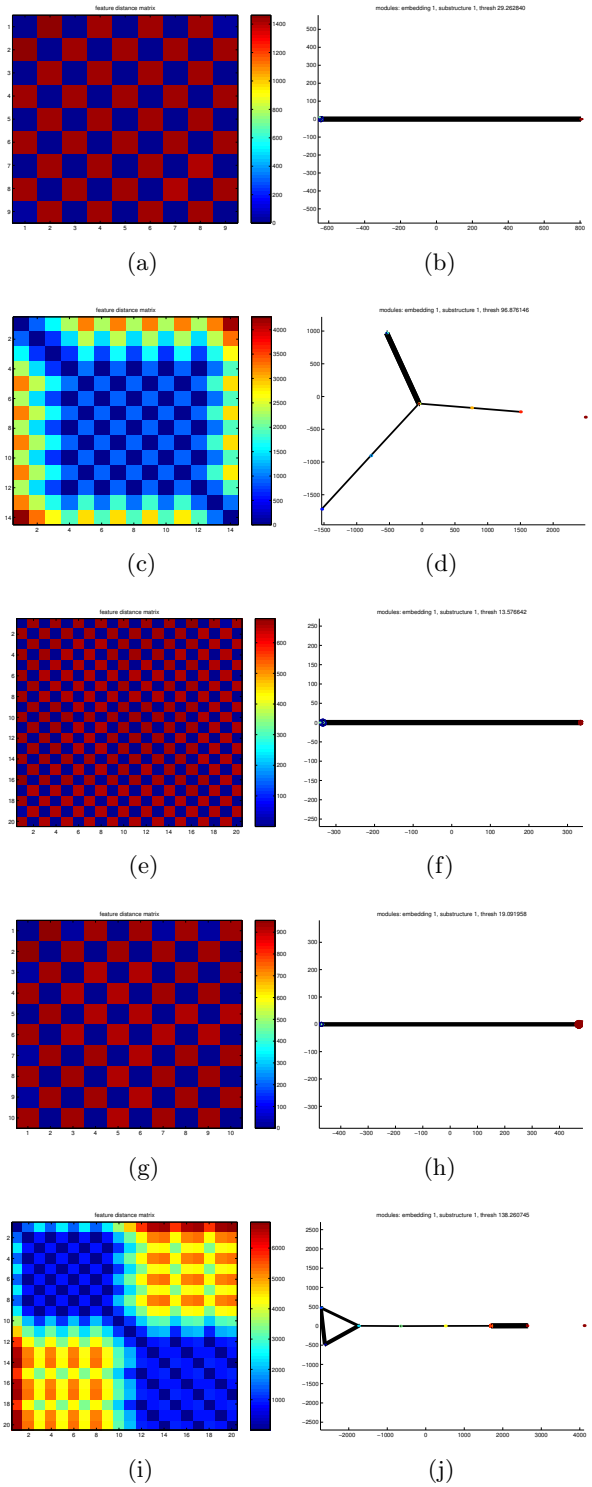


Figure 5.14: *Distance matrices (left column) and feature group clusters (right column displayed color coded spheres) with transitionally weighted connections for activities isolated from Input Motion 1: (a,b) the "cabbage patch" dance, (c,d) the "twist" dance, (e,f) vertical waving of a single arm, (g,h) jab punching, and (i,j) combined jab and uppercut punching.*

premature termination. The issues regarding premature conclusions of motion synthesis are discussed in Section 5.3.5.2.

Selected motions synthesized from our input motions was actuated by a dynamical humanoid simulation, Adonis [91]. Adonis is a 20 DOF humanoid torso containing joints for the waist, neck, shoulders, elbows, and wrists. Applicable DOF from the synthesized motion were used to drive Adonis by setting moving desired postures for low-level PD-servos. Synthesized motion were output as Biovision BVH files [82] and read into Adonis for desired trajectories to actuate. The purpose of actuating Adonis is to demonstrate a humanoid robot can be controlled by derived primitives, not to evaluate performance of purposeful motion. In fact, the kinematic mismatch between Adonis and our input motions produces awkward looking motion when actuated by Adonis.

5.2.5 Synthesized Motion Feedback

Conceptually, motion synthesized in the previous section should be structurally similar to an original input motion. Consequently, a behavior vocabulary derived from synthesized motion should be similar to the originally derived vocabulary. To evaluate the validity of this *feedback property*, motion synthesized from the previous section can be used as input motion into PDBV. As discussed in Section 5.3.5.2, however, motion synthesized with simple, unstructured, and undirected high-level controllers may not hold to this property. Instead, vocabularies from isolated activities, such as the jab punching, were used to synthesize motion to test the feedback property.

In particular, we focus on the jab activity when evaluating the feedback property, as this activity is one of the simplest. Figure 5.17 shows feature groups and cluster

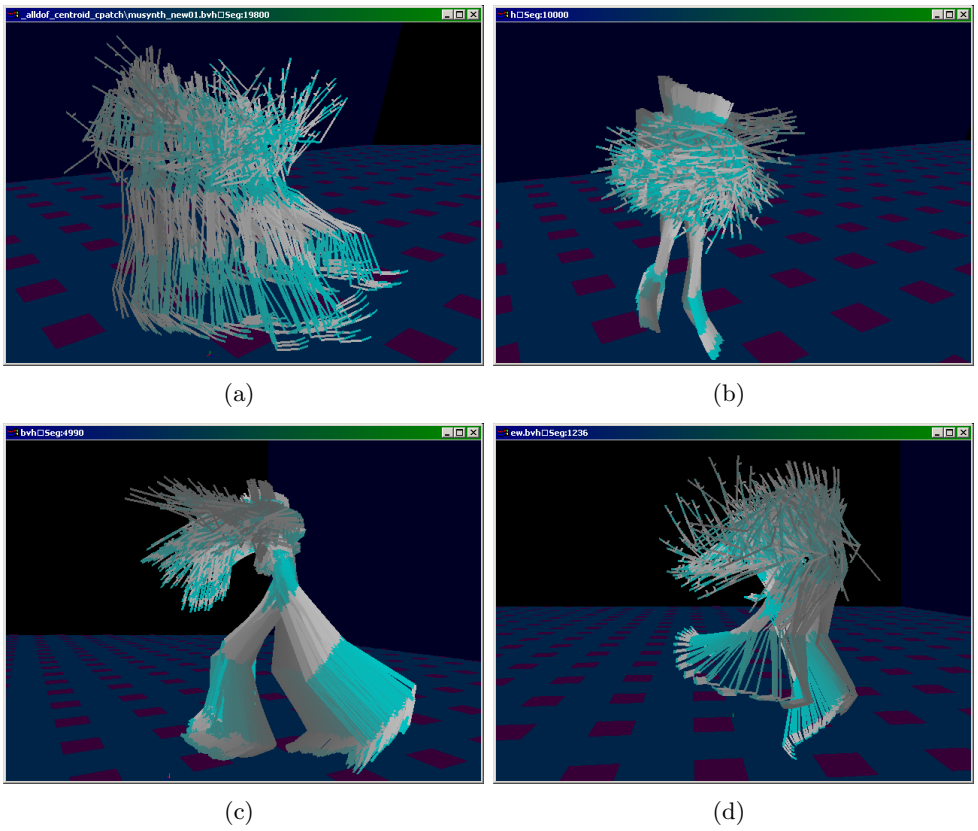


Figure 5.15: *Postures sampled from motion synthesized by derived vocabularies. Global position and orientation information was included for (a) the "cabbage patch", (b) horizontal arm waving, and (c) jab punching and not included for (d) combined jab and uppercut punching.*

assignments from using the synthesized jab motion as input into PDBV. As illustrated in Figure 5.15, motion synthesized from the jab vocabulary is structurally similar to its original input motion. However, this motion begins to diverge towards a "downward punch", partially due to the inclusion of global position and orientation information. PDBV uncovers this motion as two main meta-level behaviors, one for the original jab and one for the downward punch. Included with these behaviors was an additional meta-

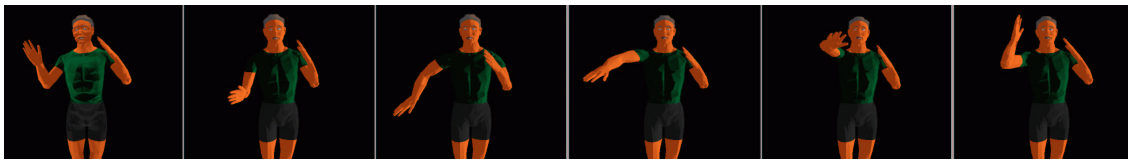


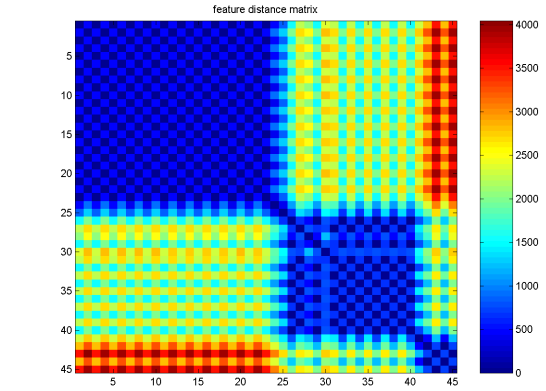
Figure 5.16: *Snapshots from Adonis performing punching motion synthesized from a behavior vocabulary.*

level behavior consisting of "small" high frequency motions that were inappropriately segmented.

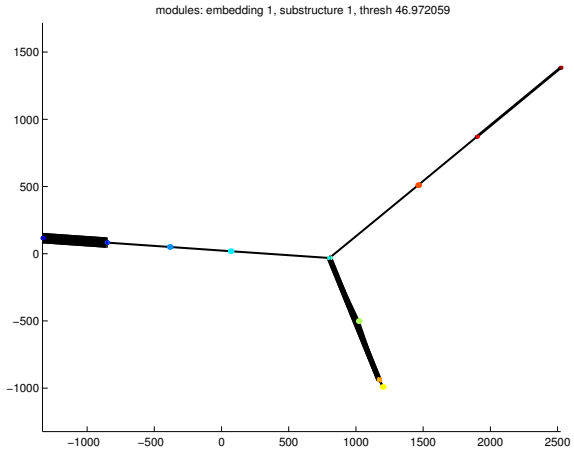
5.2.6 Segmentation Variation

Because sequentially segmented ST-Isomap is used, PDBV is highly dependent on the features extracted by a chosen segmentation mechanism. Behavior vocabularies for Input Motions 1-3 were derived using manual, z-function, and kinematic centroid segmentation. Primitive feature groups found for these vocabularies are shown in Figure 5.18. Although none of these methods were without flaws, each of them yielded structurally indicative feature groups based on the type of segmentation heuristic used.

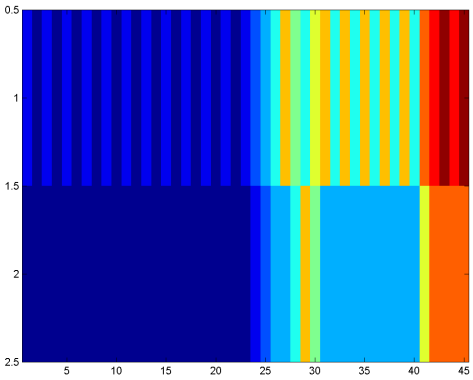
From manual inspection of the segmentation and extracted feature groups, manual and kinematic centroid consistently provided more plausible results than z-function segmentation. This statement is illustrated for Input Motion 3. This input motion was structured to consist of two main primitives, "reach out" and "return", with occasional intervals of being idle in the zero posture. Using manual segmentation, PDBV cleanly finds these two primitives exactly, with idle intervals divided into two primitives for "enter idle" and "return to reaching". The only segment that slightly varies from this structure is the beginning of the input motion. This segment is placed in proximity to the



(a)



(b)



(c)

Figure 5.17: *Results from using synthesized motion from the vocabulary derived for the isolated jab activity, including: (a) the distance matrix produced by ST-Isomap, (b) feature groups and transition weighted connections, and (c) cluster assignments at the first and second levels.*

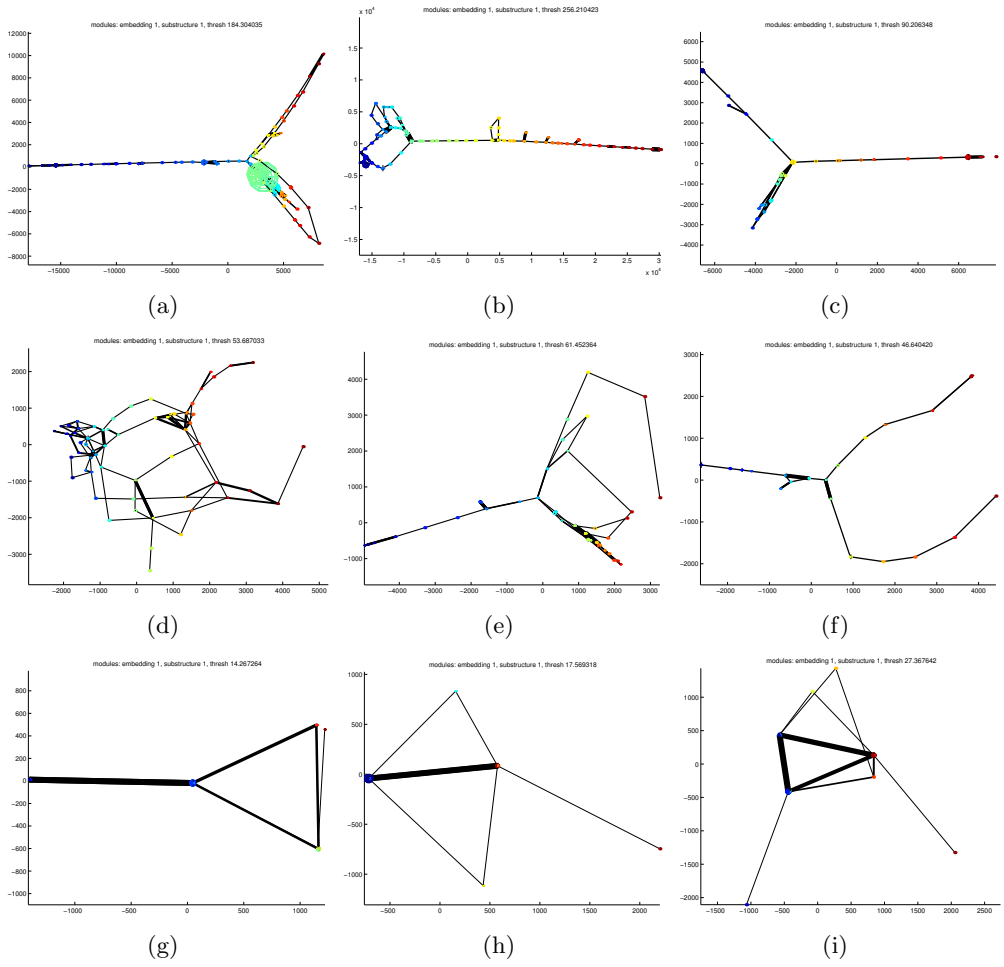


Figure 5.18: *Initial spatio-temporal embeddings clustered into actions for Streams 1, 2, and 3 (from top to bottom) clustered for actions. The rows of subplots are for manual, kinematic centroid, and z-function segmentation (left to right).*

	Num. Segments	Mean	St. Dev.	Min.	Max.
Input Motion 1 (Manual)	250	90.19	113.72	15	970
Input Motion 1 (Centroid)	226	99.30	127.93	17	1017
Input Motion 1 (Z-function)	62	329.85	539.02	6	2778
Input Motion 2 (Manual)	210	44.73	20.95	10	194
Input Motion 2 (Centroid)	148	62.05	23.95	20	153
Input Motion 2 (Z-function)	64	141.55	127.30	25	950
Input Motion 3 (Manual)	73	125.12	41.88	6	200
Input Motion 3 (Centroid)	64	142.45	40.44	7	261
Input Motion 3 (Z-function)	84	104.52	48.02	5	259

Table 5.2: Statistics about the segments produced by each segmentation method for each input motion without global position and orientation. The statistics for each segmentation specify the number of segments produced, mean segment length, standard deviation of the segment lengths, minimum segment length, and maximum segment length.

	Manual	Centroid	Z-Function
Input Motion 1	84	78	32
Input Motion 2	62	37	20
Input Motion 3	7	5	10

Table 5.3: Number of primitives derived for each input motion and each segmentation procedure.

appropriate feature, but has no preceding common temporal neighbor to bring it into greater proximity. PDBV with KCS estimated the structure of Input Motion 3 as two reaching primitives and several spurious clusters. These spurious clusters are actually segments of idle motion that should have been corresponded into proximity. In contrast to the consistent output of manual and kinematic centroid segmentation, z-function did not provide a consistent segmentation. For instance, it yielded 2 reaching primitives over some intervals and 3 reaching primitives over other intervals. The resulting embedding and feature groups are mainly indicative of 3-primitive punching motion that is erratically spliced with spurious, but structurally relevant, 2-primitive reaching and idle segments.

We attribute this problem to the inability to find an appropriate global threshold value for the z-function due to local properties in the motion, as shown in Figure 5.19. The z-function for Input Motion 3 has varying minimum and maximum extrema values over different intervals of the motion. The variance in local maximum values across the motion is due to greater movement speed for reaching higher than lower reaching motions. The variance in minimum extrema is a combination of the subject stopping for too short of a duration between reaches and the low pass filter deteriorating the stops to reduce noise.

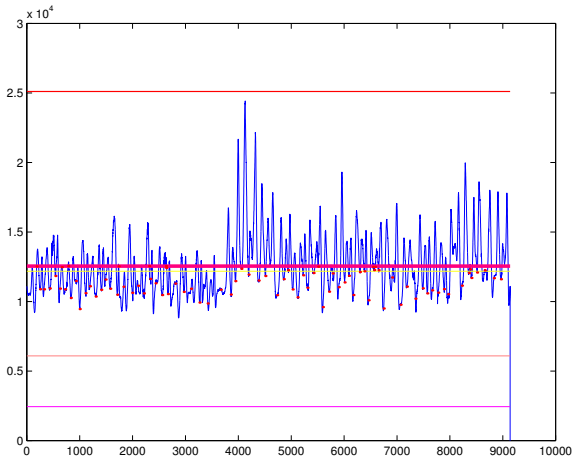


Figure 5.19: *Z-function values over Input Motion 3 shown by the blue line. Segment placements are specified by red dots. The thick magenta line was used as the global threshold.*

5.3 Discussion

In our methodology for deriving behavior vocabularies, the quality of the resulting structures is heavily dependent on the preprocessing mechanism and the appropriateness of the parameters used in spatio-temporal Isomap. The preprocessing of the input motion

stream prepares the motion data for dimension reduction mostly through segmentation. If the segmentation is not sufficient, the subsequent data are useless regardless of the capability of dimension reduction for discovering spatio-temporal structure. Given that the preprocessing is sufficient, the user parameters for spatio-temporal Isomap define the type of structure we are looking for in the data. If the parameters are not reflective of the spatio-temporal structure, the resulting action and behaviors will not provide a useful substrate for perceptual-motor algorithms.

5.3.1 Consistency and Sensibility in Motion Preprocessing

In preprocessing an input motion, our aim is to provide an ordered set of motion segments amenable to discovering underlying structure in the motion. The heuristic used for segmentation should provide a division of the motion that is both *consistent* and *sensible*. Consistency, in this context, means similar intervals of motion yield similar motion segments. By sensible, we mean that each segment is *i)* a significant expression motion that can be intuitively labeled by a human user and *ii)* considered by a human user to consist of an indivisible, atomic performance of some behavior. To illustrate our concept of sensible segmentation, segments that are too short in duration may not express any useful behavior. Together, a set of "short segments" may express an intuitive behavior, but not individually, and would be similar to overfitting using motion textures [87]. Segments that are too long in duration may contain behaviors that are specific to a sequence of sub-behaviors, which would provide of more modular description of the motion. "Long segments" may be difficult to place into clusterable proximity because to their spatial signatures may be distal and common temporal neighbors will be more sparse. Sensible

segments are in the middle ground between short and long segments, large enough to describe meaningful motion but small enough to provide modularization.

The obvious means for ensuring a segmentation with both of these properties is to use human judgment. An automatic method, however, requires much less time and effort and yields a potentially reasonable segmentation. The z-function method provides a reasonable segmentation if the stream contains discrete movements, movements from one configuration to another described by peaked velocity profiles. If we believe these configurations are keyframes underlying the motion, we would expect to see peaks and valleys in the z-function indicating transitions between and achievement of the keyframe configurations. The differences in transition and achievement intervals are distinguished by thresholding. In practice, however, segmentation is very sensitive to the setting of a global threshold and produces several intervals of erratic segments. Low thresholds produce many spurious segments. High thresholds produce several merged segments. Z-function segmentation might work well if an optimal threshold can be found, locally estimated, or the actual segment peaks can be further accentuated. Furthermore, if movement in the DOF does not stop when a keyframe is achieved or there are no specific keyframes to signal accomplishment, the z-function cannot detect segment boundaries. Thus, z-function segmentation provides a somewhat consistent, but not sensible segmentation for motion that is not clearly point-to-point movement.

Kinematic Centroid Segmentation proved to be a much more sensible method for our test data with a small cost in consistency. Motion segmentation in this manner has three problems. The first is that motions in each limb must make the centroid move past a threshold distance from its previous position at the current segment boundary. Thus, the

motion must be large enough to be detected. All of our test motion can be segmented sensibly with this assumption, except for the synthesized jab motion. Second, the segmentation is performed in a greedy fashion, the frames preceding the current segment boundary are considered optimally segmented. This assumption causes problems if segment boundary is placed badly. Such an errant boundary could propagate through the segmentation and yield inconsistencies. In practice, we found that bad placements were typically localized to intervals when a transition between scripted high-level activities occurs. Thus, a behavior may begin with an awkward segment, but the rest of the behavior is consistent. The third problem occurs when two reasonable segments are detected as a single segment. This situation happens when one segment moves the centroid some distance away from the previous placement and the second segment continues the increase in displacement away from the previous placement. An example of this circumstance is the motion of an arm beginning at the side of the performer, brought to a fighting posture, and smoothly extended to a punch. While undesirable, the same ambiguity could result from human judgment. Is the motion a single punch segment or two segments for a "ready action" followed by a punch? Even though this type of motion is infrequent in the data, it is handled in a sensible, but not perfect, manner.

The current implementation of kinematic centroid segmentation handles intervals of idle motion less elegantly than the z-function, but could be modified to include a final pass to detect idle motion. The two pass approach, which segmented the right arm followed by the left arm, worked well. However, it is not clear how well such a method would scale to full body motion, in which additional levels of substructure coordination occur in the

kinematic hierarchy. There also exists the possibility for combining the two automatic methods by using the z-function as one pass in the segmentation process.

In order to apply the segments to our embedding procedure, we must normalize them to an equal number of samples. This normalization removes the timing elements of the segments. While the normalization produces no side effects on the embedding, the feature groups that result from the embedding do not have the same timing as the segments from the motion stream. Motion trajectories interpolated from a primitives are of the same temporal length. If we wanted to somehow retain segment timing, a scaling variable could be added to the segment matrix before embedding. However, we may prefer to leave timing as a step length parameter for a primitive flowfield.

5.3.2 Parameter Tuning for ST-Isomap and Exemplar Grouping

Parameter tuning is a significant issue for ST-Isomap and clustering in grouping motion segments with a common spatio-temporal structuring. ST-Isomap parameters are particularly important because they define the local neighborhood of each point. Local spatial neighborhoods are the foundation for determining common temporal neighbors, and consequently for performing distal correspondences. Distal correspondence is the primary criterion for whether two segments will be in clusterable proximity in the embedding space. Ideally, we would prefer for the local neighborhood of each point to consist only of exemplars with the same underlying behavior. Modularization in this case would be simpler because exemplars of another behavior will never be CTNs with this point. Consequently, *merging artifacts* would not occur in feature groups because inter-behavior data pairs would never be included in the same CTN component. Instead,

local neighborhoods are selected by k nearest neighbors, potentially producing a motley mixture of exemplars from different behaviors in a given neighborhood. The potential for such *nonrepresentative local spatial neighborhoods* is at the root of both merging artifacts, but also plays a role in *splitting artifacts*. Splitting artifacts occur when exemplars of a single underlying behavior are placed into multiple CTN components. Splitting artifacts are structurally due to underrepresentation and sparseness of exemplars for a particular underlying behavior. More specifically, splitting occurs when a data point has no other exemplars of the same behavior in proximity or there is a large gap between subsets of exemplars of the same behavior. As stated in Chapter 3, splitting is typically the result of *exemplar sparsity*, exemplars of an underlying behavior are underrepresented and/or not distributed across a behavior's span of variations. However, local neighborhoods that are too small in volume or scope magnify the problems of splitting.

The construction of local neighborhoods is determined by the criteria for selecting nearest neighbors and the distance metric used for spatial Isomap. In order to improve the quality of the local neighborhood, one or both of these mechanisms must be improved. While such an undertaking may prove difficult in general, domain knowledge may be applied for particular types of motion data to improve the distance metric for the neighborhood selection mechanism. In the absence of domain knowledge, we currently use a estimate of local neighborhood cardinality or volume that is applied globally based on Euclidean distance. Globally application of a local criteria has proved useful for us, but presents problems when different subsets of the data warrant different local parameters. For the distance metric, a data-driven approach to DOF weighting, such as [137], can be used to scale a metric to produce more appropriate distance. Our current feature

distance metric is the straightforward Euclidean distance for all the DOF over all the samples in a segment. Using this metric, two movements following the same trajectory in different directions may be relatively equal to distance the of one of those movements and a slightly offset version of itself. Intuitively, motion following the same direction is weighted more heavily together than motion opposite directions.

5.3.3 Splitting and Merging of Feature Groups

As alluded to in the previous section, splitting and merging artifacts are a problem for appropriately modularizing motion. Ideally, a feature group should group exactly those motions that are exemplars of an underlying behavior. However, this ideal result may not occur as a consequence of nonrepresentative local neighborhoods and exemplar sparseness. These problems occur due to *exemplar merging* and *exemplar splitting*. Exemplar merging is the inclusion of instances of different underlying behavior into the same derived action. Exemplar spitting is separation of instances of the same underlying behavior into different derived behaviors. Local neighborhoods that are too inclusive could potentially allow instances of different behaviors to have common temporal neighbors. Consequently, these instances will be included in the same feature group. This resulting action will not be representative of a single underlying behavior, but rather an unstructured amalgamation of multiple behaviors. This situation prevents a clear modularization of input data. In practice, behavior merging is likely to occur in the presence of *spurious motion segments*, typically resulting from transitioning between higher-level behaviors or an idle posture. These segments can usually be disregarded after manual inspection.

Behavior splitting occurs in two forms: split instances and split behavior contexts. In the case of split instances, the instances of an underlying behavior require multiple CTN components. More simply, there exists a pair of instances that are not connected to each other through the neighborhoods of the other instances in the CTN component. As stated previously, the cause of this splitting is exemplar sparseness. For instance, the performance of a high waving motion and low waving motion will be derived as two separate behaviors if there are not other waving exemplars in middle to connect these instances. Behavior splitting alone is not overly problematic because if not grossly over-split, the modularization is still clear. Splitting in conjunction with merging, however, is a problem because the effects of merging become magnified and the modularization is weakened.

The problem of *split behavior contexts* occurs when instances of the same behavior appear in the temporal context of two different behaviors. For example, exemplars of behavior C may appear in temporal contexts such as $A \rightarrow C \rightarrow D$, $A \rightarrow C \rightarrow E$, or $B \rightarrow C \rightarrow F$. In this situation, PDBV will appropriately separate exemplars into feature groups based on these different temporal contexts. This separation is a direct application of the definition of common temporal neighbors. In order to form one cluster for the exemplars of C, A and B would become a merged behavior, as would D, E, and F.

These context-dependent behaviors of C may be related through local neighborhoods, but will not have common temporal neighbors. Context-dependent feature groups provide more information that may or may not be desired. If the modularization is required to be concise as well as clear, these extra modules will be undesirable. However, these extra

action modules provide information as to which actions are more appropriate in certain contexts.

Behavior splitting and merging is partly an artifact of local neighborhood construction. Better construction of these neighborhoods can partially address these artifacts. Clustering of exemplars in embedding spaces, however, can be just as responsible for producing artifacts. ST-Isomap may be able to appropriately bring exemplars of an underlying behavior into clusterable proximity, but this does not ensure that such clusters are appropriately found. We have proposed the use of sweep-and-prune clustering as an alternative for separable clusters that does not require initialization or *a priori* specification of the number of clusters. Instead, embedded data are partitioned based on a threshold distance for separating points projected onto an axis. Setting this threshold distance, however, can be problematic. We currently set the threshold distance as fraction c_{sap} of the length of the embedded data bounding box diagonal. Using $c_{sap} = 0.01$ for most of our input motions worked well for extracting the greater majority of feature groups. Minor artifacts occurred for input motions containing more structure, such as Input Motion 1. For these input motions, the embedding bounding box was more densely packed with clusters and, thus, was more susceptible to making clustering errors. Similar to problems with z-function segmentation and local neighborhood construction, these errors are partially due to locally applying a global decision setting. Feature group clustering errors at level propagate to higher-level feature groups. This propagation is due to local neighborhoods at one level being defined by the feature groups at the previous level. We partially attribute the inability to guarantee meta-level convergence to clustering errors.

Between high-level activities were an additional artifact observed in the derived feature groups, *spurious transitions*. These transitions were not intrinsic to the derived behavior structure. In some cases, these transitions are actually intrinsic to the behavior and a result of behavior splitting. In other cases, they were simply performance transitions between activities in the input motion. These transitions are considered spurious because they are not necessary to the behavior, but are possibly useful for transitioning between behaviors. The problem of spurious transitions suggests the use of HMMs for sequencing primitives as states. While such an idea is useful, it is outside the scope of the dissertation.

One possibility for avoiding splitting and merging artifacts is to post-process the derived feature groups. The post-process provides a better modularization and removes spurious segments by splitting and/or merging behaviors. The merging post-process could analyze a pair of behaviors and decide whether to merge them based on some similarity criteria. The splitting post-process could analyze a behavior for instances that do not belong based on criteria specific to the module or a more specific distance measure. Our inclination is that procedures for eager evaluation and pre/post-condition determination could aid in post-processing.

The issues regarding splitting and merging are rather subjective. Our vocabulary derivation methodology does not seek to provide a definitive derivation, but rather a reasonable one. Rabiner made a similar statement in the context of HMMs [111]. We should only expect our derivation to be as good as the appropriateness of our parameters, distance metrics, clustering, etc. Subjective judgment can always be incorporated through manual refinement, post-processing, and domain knowledge.

5.3.4 Temporal Neighbors vs. Phase Space

We developed sequentially segmented ST-Isomap with two approaches to incorporating the temporal characteristics of an input motion [71]. One method is to adjust pairwise distances between spatial and temporal neighbors, as described in Chapter 3. Another approach is to work with the data in *phase space* by concatenating each data point with velocity information. In the phase space view, we assume that spatial Isomap can be performed for spatio-temporal structure by concatenating velocity information to each data point.

This information alone will not provide a clear modularization because it is suited for proximal disambiguation and not distal correspondence. Even with a general weighted distance in the form of $x^T W x$, phase space pairwise distances would be unable to perform distal correspondences. Additionally, the mechanisms used for phase space distances are significantly different than in adjusting temporal neighbors. Most significantly, the phase space distance is a continuous measure as opposed to the step function imposed by common temporal neighbors.

For modularization purposes, the step function of common temporal neighbors is preferred to continuous phase space distances. This preference stems from the desire to provide a clear distinction as to whether two points will be in the clusterable proximity or not. However, considering problems in local neighborhood construction, phase space may serve as an appropriate representation of the input data for finding constructing local neighborhoods and estimating CTN components.

5.3.5 Primitive Behavior Generalization

Primitive feature groups, providing the lowest level of modularization, are generalized via eager evaluation into primitive behaviors. Interpolation of a dense sampling with a primitive is performed for generalization. Shepards interpolation was used in our current implementation, providing a relatively simple and intuitive means for estimating unobserved motions of a primitive. However, interpolation in this manner is far from perfect and yields several undesirable artifacts. First, Shepards interpolation is essentially radial-basis interpolation and is not suited for precisely accurate reconstruction mappings. Consequently, sampling of an exemplar space may still result in significant gaps in a primitive manifold. These gaps are small, inaccessible areas of the primitive manifold that are unaccounted for but representative of significant areas of motion. These pockets are especially problematic near the boundary of the support volume of the exemplar space.

One good property of Shepards interpolation is that it can work for dense scattered data sets. If we instead assume our feature group to be a sparse set of examples, we could apply an exemplar based interpolation method, such as in [116] and [128]. These methods has been applied to articulated motion by fitting hyperplanes and radial-basis functions to ensure that the interpolation reconstructs each example accurately.

A potential contributor to the problems of Shepards interpolation is using exemplar spaces produced directly by ST-Isomap. Once feature groups have been formed, manually setting up an "adverb" space as in [116] may not require large amounts of human effort. For an automatic alternative, we could perform another unsupervised learning mechanism

to the exemplar space, to reembed in a possibly more meaningful space, or perform supervised learning to estimate the mapping between joint angle and exemplar spaces. Other alternatives are probably available as well.

Meta-level behaviors in our current implementation are not parameterized, but the primitive components of each meta-level behavior are. The meta-level behavior specifies how primitives can be performed in sequence, similar to the verb graph of [116]. However, our behaviors are meta-level structures that cannot be interpolated. Instead, behaviors index into interpolable and predictive primitives. We could create this interpolation space to truly make the behavior a primitive. Such an interpolation space could be in a similar form to a style machine [17].

5.3.5.1 Support Volume Coverage for Primitive Flowfields

In the presentation of PDBV, we use and construct primitive forward models in a very basic manner, as flowfields. We have focused on such flowfields as a means for combining the flexibility of exemplar-based behaviors with the ability to express the nonlinear dynamics of kinematic behaviors. However, there are several unaddressed issues in using flowfields for primitive forward models. One such issue relates to volume of support for each primitive forward model in joint angle space. We currently consider that each forward model provides valid predictions only with the volume spanned a flowfield. Primitive forward models produce no prediction when outside the span of its flowfield. Additionally, primitive forward models currently incorporate no explicit information about skill level objectives (e.g., "reach to this point"), preconditions, or postconditions. A primitive forward model can provide information about the temporal progression of a location in

joint angle space. For example, a given kinematic configuration indicates 2/3 temporal progress through a primitive when located near elements of the flowfield representative of 2/3 completion. Precondition and postcondition volumes could be built around points at the beginning and end of a flowfield's temporal progression.

These issues for primitive support volumes raise two questions: *i)* how much space should the support volume of a primitive forward model cover and *ii)* how much precondition and postcondition information is required for task-level controllers? We do not have exact answers to these questions, but we will provide some alternatives for support volume coverage. We could restate the first question to ask: should a primitive forward model span the entire joint angle space? The answer to this question depends on the approach to covering the joint angle space. Covering the joint angle space with flowfield nodes is infeasible and subject to the *curse of dimensionality*. However, an exploration procedure that retains a sparse number of exemplars while exploring new areas of the joint angle space may be feasible. Another more straightforward approach is to add an attraction component that draws kinematic configurations towards the support volume. Our inclination is to avoid "attraction into validity" because motion produced during the attraction would not necessarily be representative of the underlying behavior.

5.3.5.2 Motion Synthesis

One application where the issues of support volume coverage become more clear is motion synthesis. Our basic motion synthesis mechanism, described in Chapter 6, was limited in its ability to produce motion due to these issues. One problem we encountered were dead-ends that abruptly terminated the synthesis procedure. A dead-end is encountered

when the current kinematic configuration is not within the support volume of the currently active primitive without completing the temporal progression of this primitive. Deadends illustrate the problems of our current synthesis mechanism use of the primitive support volumes. First, dead-ends would never be encountered if the primitives supported all of the joint angle space. Second, the high-level arbitrator cannot simply switch activation to another primitive without knowledge about that primitive's preconditions. We briefly experimented with allowing the arbitrator to switch activation during the progression of a primitive, which resulted in erratic movement from constant switching between primitives.

5.3.6 Kinematic Substructures

As stated previously, kinematic substructures are limbs that could be acting in coordination or independently. Thus, it may make sense to treat these structures independently, using separate embeddings and feature groups for each substructure to help accentuate their independence. The result would be multiple behavior vocabularies, each specific to a particular substructure with limb level primitives. However, these behavior vocabularies would then require coordination across limbs through some mechanism, similar to Bregler's movemes [18].

5.3.7 When is PCA or Spatial Isomap Appropriate For Motion Data?

We do not propose ST-Isomap as a replacement for other dimension reduction techniques when the input data are spatio-temporal or kinematic motion. Every dimension reduction method has its strengths and weaknesses for analyzing and visualizing motion. We have emphasized the limitation of PCA to linear PCs throughout this dissertation. Unlike spa-

tial and spatio-temporal Isomap, however, PCA is not as sensitive to the size and density of the input data set and does not require parameter tuning. PCA can reveal clusterable structure for several types of data sets. For situations where the linear structure of PCA is too limiting, spatial Isomap provides a flexible model-free means for extracting structure. For both PCA and spatial Isomap, however, the emphasis is on estimating embeddings that provide greater parsimony, not uncovering clusters. Sequentially segmented ST-Isomap is geared for modularizing input data into clusters. This focus is what drives the derivation of behaviors by PDBV. Given this difference, we find that methods like PCA and spatial Isomap are more suited for visualizing than modularizing motion data.

5.4 Summary

We have presented an evaluation of our methodology for deriving behavior vocabularies for humanoid agents. Empirical evaluation was performed through an implementation of our methodology. This implementation was applied to multiple multi-activity input motions. Results from this empirical evaluation were discussed in detail.

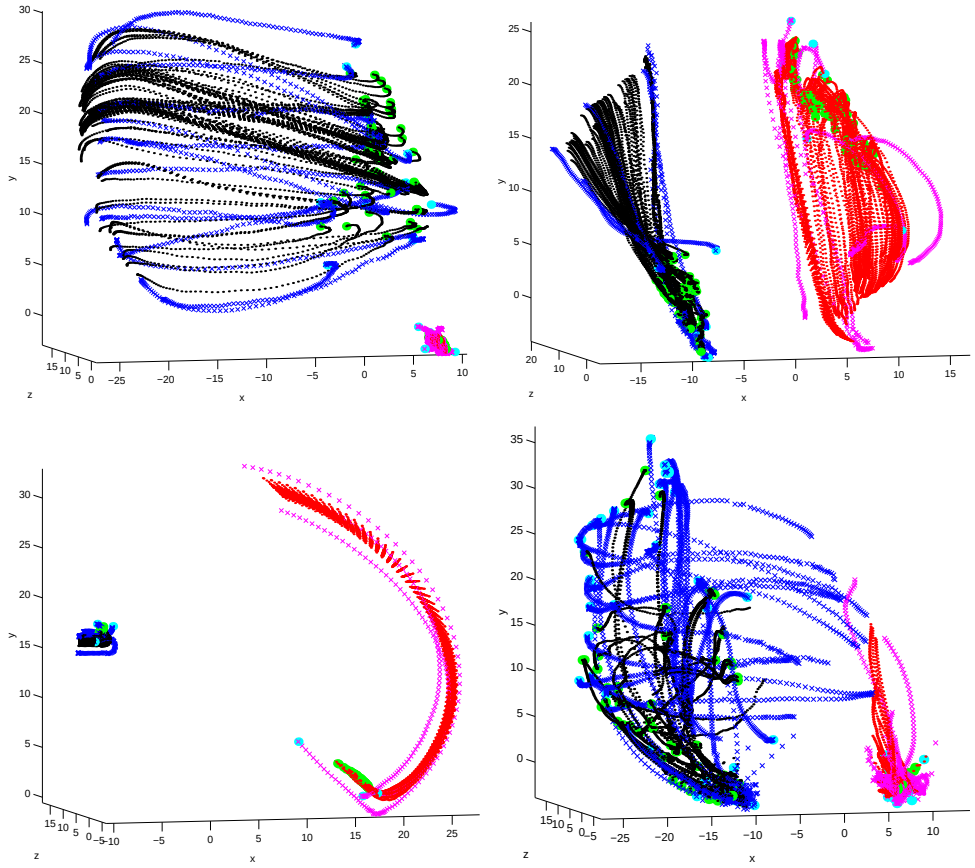


Figure 5.20: *Results from interpolating selected feature groups. Each plot shows trajectories for right and left hands in Cartesian coordinates. Trajectories displayed with crosses are instances from the feature groups, trajectories displayed with dots were generated through interpolation. The start of each trajectory is marked with a large circle. (Top Left) Horizontal arm waving, (Top Right) an feature group from "The Monkey" dance, (Bottom Left) follow through from a punch, (Bottom Right) a merged feature group containing exemplars for vertical and horizontal arm waving.*

Chapter 6

Applying Behavior Vocabularies to Movement Imitation

In this chapter, we describe the use of a derived behavior vocabulary for imitating previously unseen motion. This approach to imitation uses a vocabulary to classify and synthesize a new motion performance.

A behavior vocabulary derived using PDBV provides no explicit means for humanoid agent control. Instead, a behavior vocabulary is a modular and generalized representation of the input motion. Implicit in this representation are the structurally significant components of the input motion, which are useful for explicit applications.

One such application of behavior vocabularies is the synthesis of motion. Rose et al. [116] have demonstrated the use of on-line motion synthesis using Verbs and Adverbs vocabularies for driving a kinematically animated humanoid agent. This approach to motion synthesis produces a single motion trajectories via lazy evaluation with explicit "frame-to-frame" timing. Consequently, resulting motion has limited applicability for agents subject to physical dynamics due to the uncertainty of the dynamics. This limitation is due to two factors: *i)* a desired posture given to an agent with dynamics is not guaranteed to be achievable in a single time-step from its current state and *ii)* continually

indexing a behavior for updated desireds for deviations due to dynamics would require searching, hence defeating the purpose of lazy evaluation.

Ijspeert et al. [63] have proposed a motion synthesis (or trajectory formation) technique that represents a behavior such that an agent's desireds are robust to environmental perturbations. A primitive behavior in this context is represented as an attractor encoding a joint space trajectory as a nonlinear dynamical system. Desired motion is through a weighted combination of the primitives. This representation provides flexibility and robustness because desireds are expressed without an explicit temporal dependency. However, this representation lacks the modularity of Verbs and Adverbs, which allow an agent's capabilities to be divided into individual units. Behaviors produced by PDBV aim to incorporate the *modularity* of Verbs and Adverbs with the *flexibility* of nonlinear dynamical systems. As discussed in Chapter 4, behaviors resulting from PDBV are clustered for modularity and sampled into nonlinear dynamical systems. Thus, each behavior expresses its own nonlinear dynamics with respect to its underlying structure.

Modularity and flexibility are particularly important when considering biological mechanisms of *motor primitives* [11] and *mirror neurons* [115]. Although PDBV behaviors are not models of biology, the conceptual purpose of *functional transparency* remains the same: *The use of the same mechanisms for representing capabilities regardless of the specific function being performed.* Functions in this context include mechanisms for control, perception, or internal planning.

For PDBV primitives, functional transparency is achieved by treating each behavior as a *forward model predictor*, similar to [37]. A primitive from this perspective specifies a predicted future state of an agent based on its current state and the dynamics of

the primitive. Primitive prediction models can be transparently used for motion synthesis, treating predictions as desireds, or for motion classification, using predictions within match operators, without modification. Given these abilities for synthesis and classification, we endow an agent with the ability to imitate. We describe and discuss the application of predictors for synthesis, classification, and imitation in the remainder of this chapter.

6.1 Motion Synthesis from a Vocabulary

Our approach to motion synthesis is top-down sequential indexing into primitive behaviors using a user-defined control mechanism. This approach requires two components: primitive behaviors for producing motion desireds and a high-level control mechanism for arbitrating control among primitives. At a given instant, the high-level controller activates a single primitive, giving this primitive the responsibility for setting desireds. The activated primitive then performs prediction to produce desireds for the agent until deactivated. This arbitration-prediction cycle is continually repeated until a user-defined stopping condition is reached or a "dead-end" is encountered by the high-level controller. Similar to methods for *video texturing* [124], dead ends are encountered when no primitive can be viably activated.

The high-level controller is not necessarily limited to using primitive arbitration. The controller could perform fusion for primitive superposition by activating multiple primitives and summing their gradients for resetting the desireds. However, primitive superposition is outside the scope of this dissertation.

In its most basic form, this approach to motion synthesis is realized as *concatenation* motion synthesis, as we describe in [72]. In this form, the high-level controller is a user selected meta-level behavior, and primitive behaviors are eagerly evaluated, but are not used as dynamical systems. The high-level controller activates primitives based on derived initial state and transition probabilities. An initially active primitive is selected based on initial state probabilities. This primitive selects a variation produced from eager evaluation to initialize the synthesized motion. The current state becomes the ending posture of the synthesized motion. The high-level control mechanism then determines the next active primitive from transition probabilities and transition validity. The next active primitive selects an applicable motion trajectory variation to append from the current state. Transition validity ensures that the initial posture of the trajectory is within a certain distance of the current state, such that no large instantaneous "hops" occur in the synthesized motion.

Synthesis through concatenation suffers from several shortcomings. Similar to verb behaviors, arbitration decisions occur so infrequently that the synthesized motion is inflexible to perturbations. Furthermore, points of concatenation typically appear visually as invalid transitions. Transitions between verb behaviors are managed by an explicit transition smoothing mechanism, which may not be viable for dynamical agents. Eager evaluation produces a dense enough sampling of primitives such that this transition mechanism should not be required. However, concatenation mechanisms are typically faced with tradeoffs for adjusting the transition validity threshold. These tradeoffs exchange visually invalid transitions for increased likelihood for encountering dead-ends.

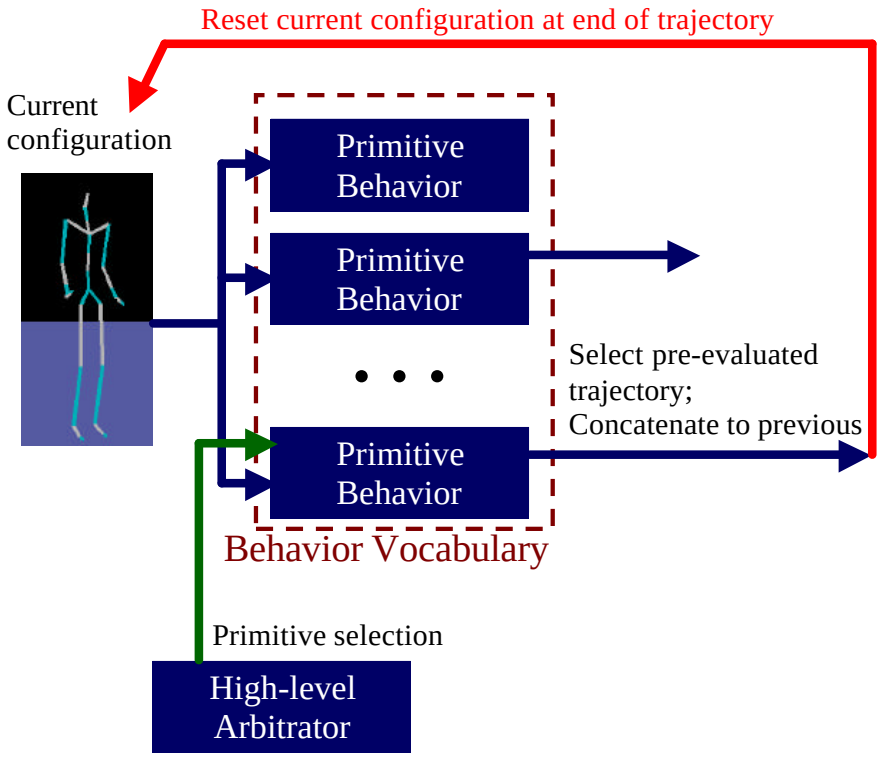


Figure 6.1: *Flowchart for concatenated motion synthesis from behavior vocabulary.*

To avoid the problems of concatenation, we use *forward model motion synthesis*. Here, each eagerly evaluated primitive is treated as a nonlinear dynamical system, shortening the time-scale of the arbitration-prediction cycle. At a given instant, the high-level controller activates a single primitive, giving this primitive the responsibility for setting desireds only for the next time-step, not until deactivation. Joint angle desireds are then reset as the predicted state returned from predictor of the currently active primitive. This arbitration-prediction cycle occurs at every time-step, unless specified by a user. Desireds produced from the prediction do not require achievement by the agents, but are requests that guide the motion along the behavior from its current state. Because desireds are updated in each time-step, invalid transitions do not occur and so smooth motion re-

sults from the synthesis. A newly activated primitive simply updates the desireds and time-steps occur from the current state. Forward model motion synthesis is susceptible to synthesizing motion with high jerk when transitions produce a new gradient that is directionally opposite to the previous gradient and when instability occurs due to large step lengths.

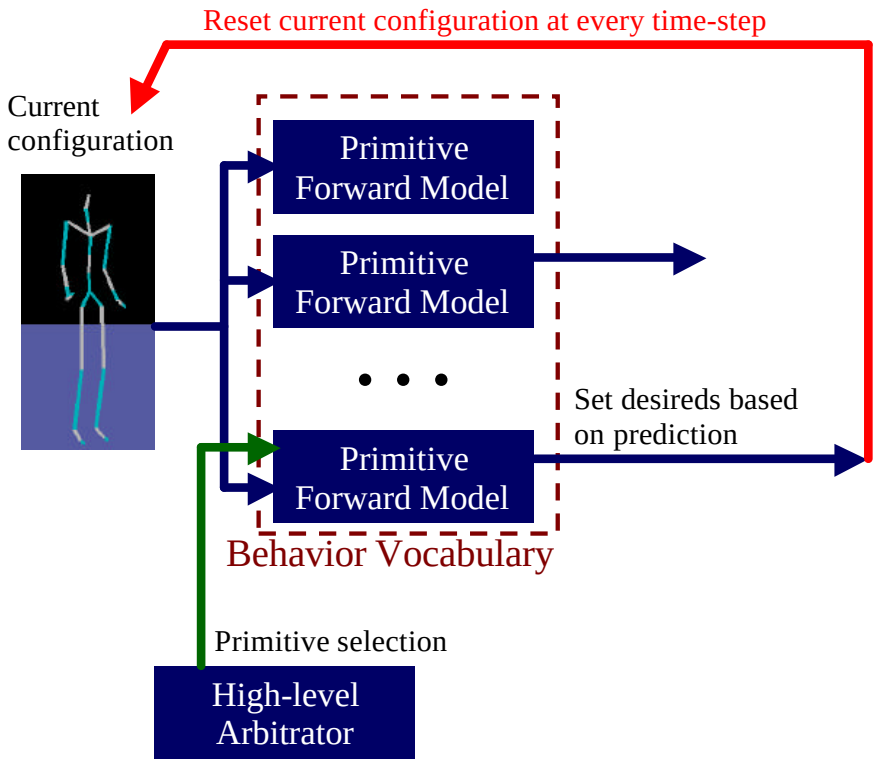


Figure 6.2: *Flowchart for forward model motion synthesis from behavior vocabulary.*

In addition to using forward models, high-level controllers can be constructed that express more complex behavior than meta-level behaviors. A variety of high-level controllers can be created to arbitrate or fuse for complex or situationally appropriate control. One possible avenue for research is to combine PDBV behaviors with the approach of Nicolescu and Matarić [105] for building task-level controllers learned from from examples.

6.2 Classification of Motion into a Vocabulary

Through classification, we encode previously unobserved subject movement into a modular description with respect to a derived behavior vocabulary. Classification could be performed top-down, starting with meta-level behaviors, or bottom-up, starting with primitive predictors. However, we present bottom-up classification procedures because they are conceptually complimentary with our top-down motion synthesis procedure. Furthermore, a high-level description of an unobserved input motion may not be represented by any meta-level behavior in a vocabulary. The methods we present for classification are not the only means for classification. A number of other classification methods, such as the one proposed by Drumwright and Matarić [39], could be used towards encoding unobserved motion into a vocabulary.

We discuss two methods for classification. The first method, *trajectory encoding*, produces a joint space trajectory imitating the input motion. The second method, *controller encoding*, encodes an input motion as a high-level controller that performs imitation by activating primitive behaviors. At their core, classification for both trajectory and controller encoding uses the same fundamental procedure and predictive primitive models. However, trajectory encoding performs imitation in a single pass to produce a frame-by-frame trajectory, whereas controller encoding separates the procedures of classification and execution by encoding demonstrated motion as a more general controller. We discuss both trajectory and controller encoding in this section, although only trajectory encoding will be demonstrated for this dissertation.

6.2.1 Imitation through Trajectory Encoding

A trajectory encoding classifier works by continually comparing predictions from each primitive with observed future states from the input movement. The current state in this context is the posture at the instance of time after the last classification decision was made. Each primitive produces a predicted trajectory from the current state over a user specified duration horizon. Classification decisions are made using a matching operator at intervals defined by the user. The matching operator provides a scalar value indicating the similarity between a primitive's prediction and the observed future state of the input movement. Motion within the decision interval is classified into the primitive with the greatest similarity value produced from the matching operator. These classifications are "hard" (winner-take-all) and do not account for motion that is not a good match to any primitive. The encoded trajectory is formed by concatenating the predicted trajectory of the classified primitive with the previously encoded motion.

A variety of match operators can be used for classification in this manner. The most straightforward is the Euclidean distance between observed and predicted postures in joint angle space. Yet, directly working in joint angle space does not consider perceptually relevant weightings of individual joints. For example, differences about the wrist and shoulder are equally weighted in joint angle space, but have different perceptual or affective weights. Wang and Bodenheimer [137] have recently proposed methods for DOF weighting. The match operator we use is Euclidean distance between Cartesian end-effector features (e.g., hands). Endeffector features abstract away DOF weighting problems but also introduce (managable) ambiguity.

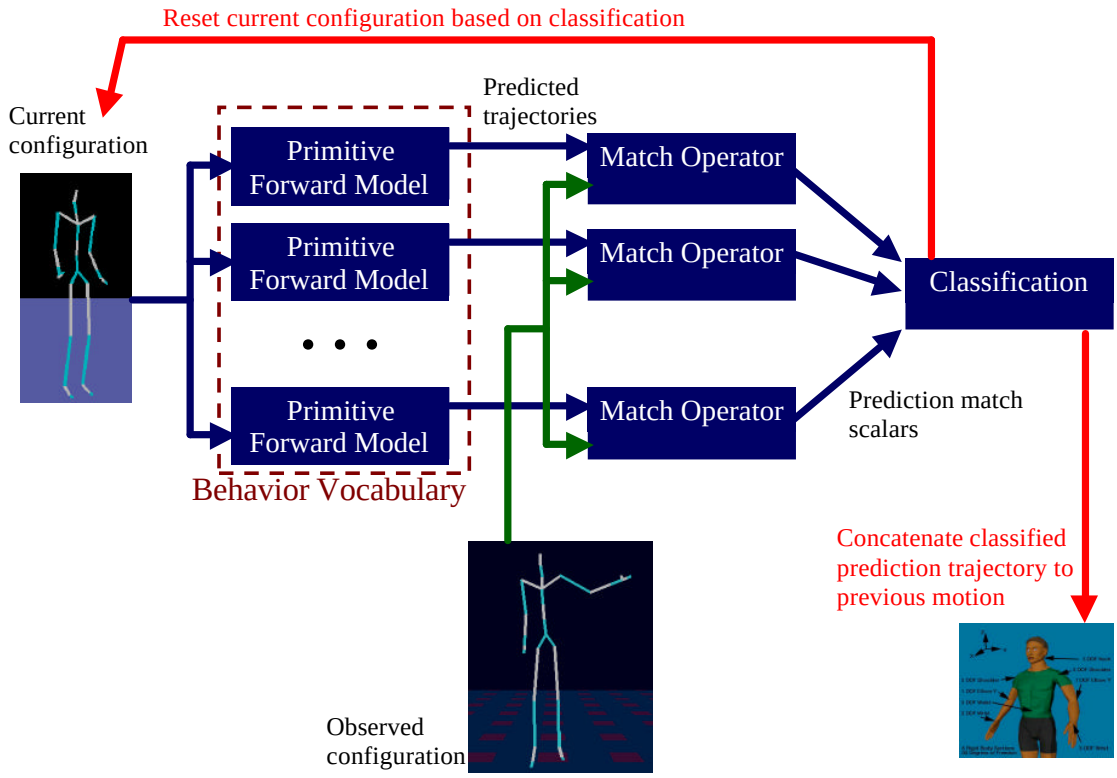


Figure 6.3: *Flowchart for imitation through trajectory encoding using a behavior vocabulary.*

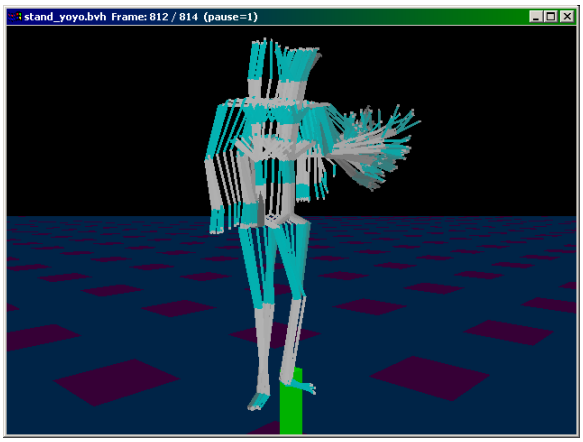


Figure 6.4: An input motion used for the testing of trajectory encoding, the "Standing Yo-Yo" motion taken from the Mega Mocap V2 collection [65].

6.2.2 Imitation through Controller Encoding

For controller encoding, classification serves to encode an input motion into a behavior vocabulary for constructing a high-level controller. Imitation, in this context, is performed by *i)* constructing a high-level controller representative of the input motion and *ii)* synthesizing new motion using this high-level controller to perform an imitation. Integral to this approach to imitation is that the same primitive predictors are used for both classification and synthesis procedures, grounding the process of imitation in the primitives.

Controller encoded motion is represented as a string of identifiers, indicating a sequencing of primitive behaviors. This string representation contains the basic structure for motion encoding. Executing a controller using the encoding string alone is not aimed at accurate reproduction of an input motion. Other movement-specific information, such

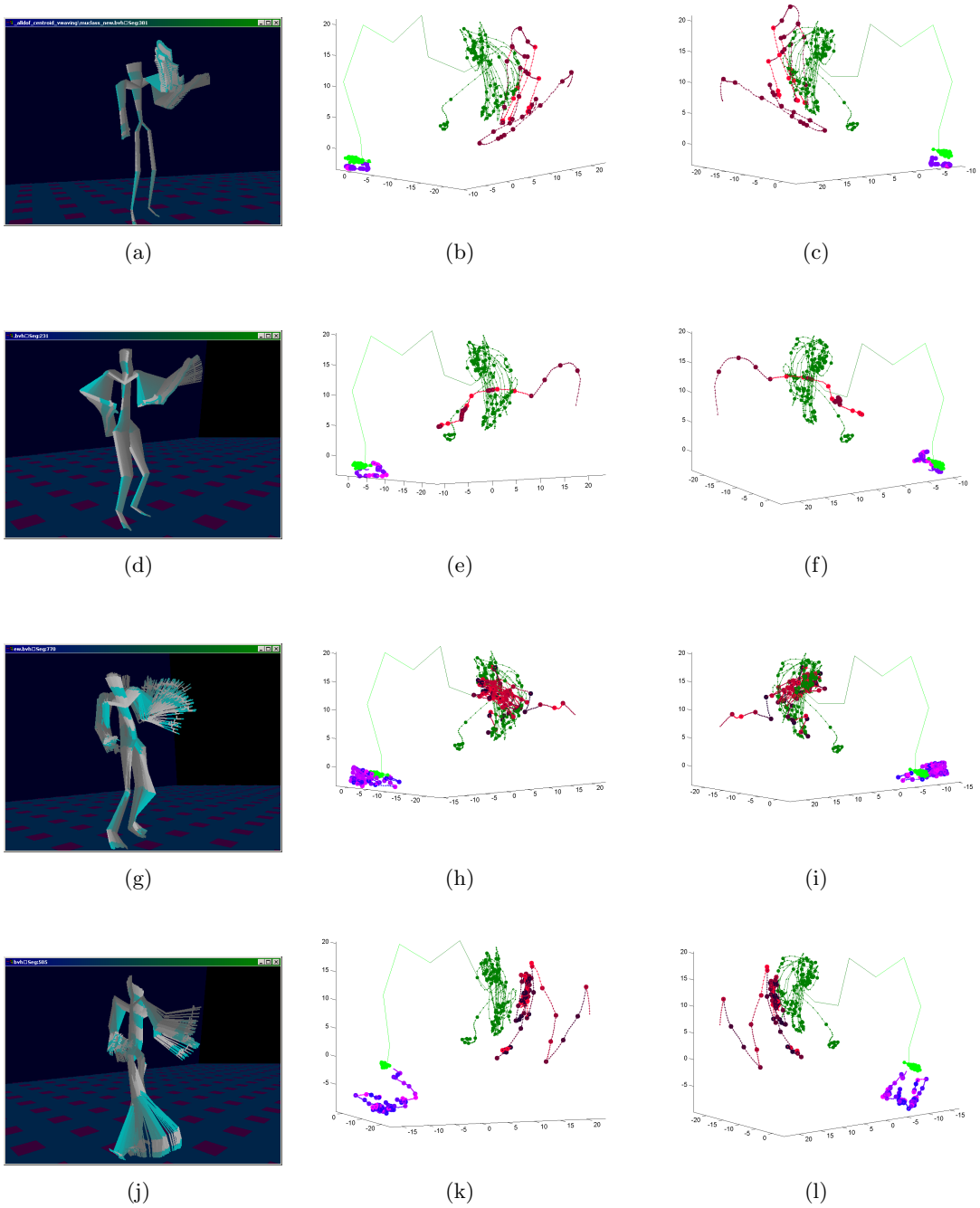


Figure 6.5: Results of trajectory encoding of the "Standing Yo-Yo" motion. This motion was encoded into vocabularies for (b-d) vertical waving, (e-g) the "cabbage patch", (h-j) punching, (k-m) the "twist". The columns of encoding show a kinematic visualization and two views of the end-effectors classification. For the end-effector classification, the observed trajectory is shown in green (light for right hand, dark for left hand) and the encoded trajectory is color coded based on the classified primitive over a decision interval.

as intra- and inter- primitive timing and specific primitive parameters are needed to augment elements of the string encoding to provide more precise reconstructions. In other words, encoding additional information changes the properties of the imitation motion from structurally general to movement specific.

Similar to trajectory encoding, a controller encoding classifier also works by continually comparing predictions from each primitive with observed future states from the input movement. The resulting encoding string is produced by merging equivalent classification decisions that are temporally adjacent. This merging assumes all transitions occur between two different primitives and ignores structural self-transitions between the same primitive.

Classification for controller encoding uses the same decision mechanism as in trajectory encoding. However, controller encoding does not simply concatenate predicted trajectories. Instead, classification results are used to encode the input motion as a string describing its execution. The encoding string is produced by merging equivalent classification decisions that are temporally adjacent. This merging assumes all transitions occur between two different primitives and ignores structural transitions between a single primitive.

For imitation, motion is synthesized from a high-level controller built from the string encoding the input motion. The imitation high-level controller is built to execute primitives specified by the sequence of the encoding string. We currently consider a controller as a limited instance of our imitation approach. Such imitation controllers are constrained to imitate the basic structure of input motion specified by the encoding string. Consequently, the high-level controller is specified to allow activated primitives to completely

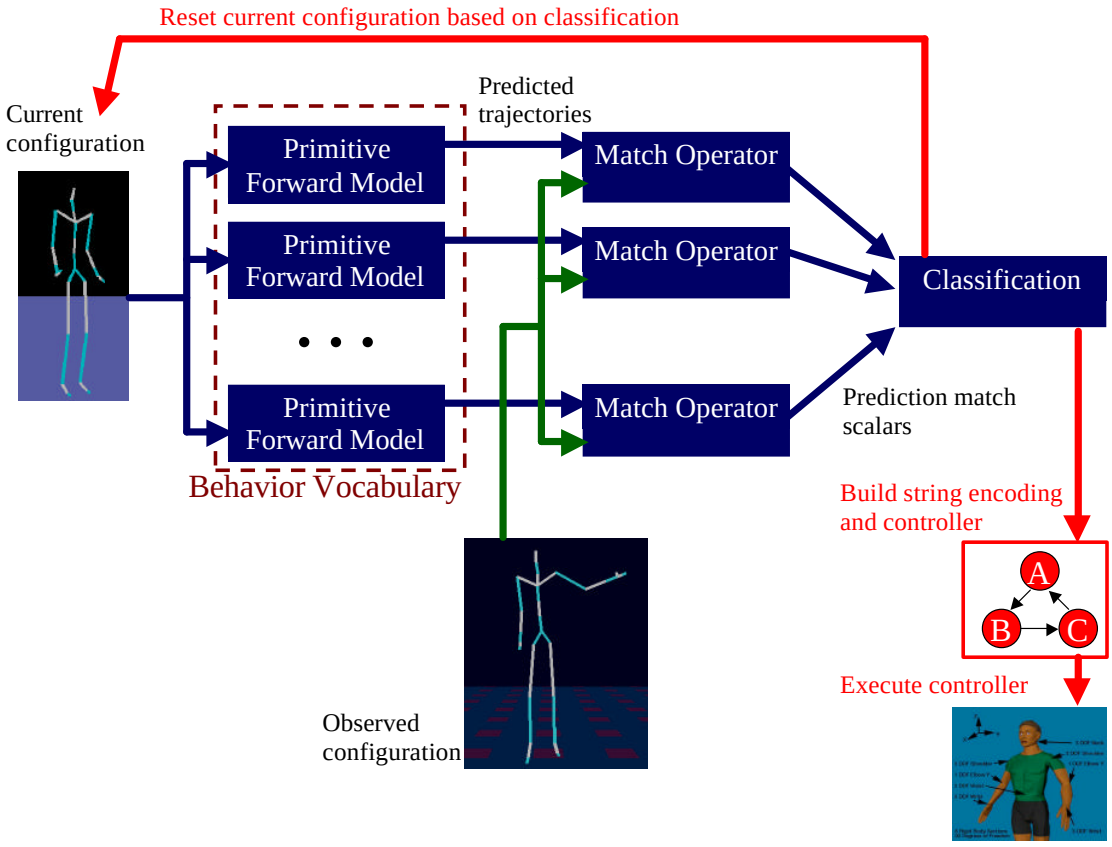


Figure 6.6: *Flowchart for imitation through controller encoding using a behavior vocabulary.*

execute (i.e., reach the end of its flowfield) before deactivation. Given an initial posture, the high-level controller sequentially executes each primitive in the string and stops after executing the final element.

Also not within the scope of this dissertation, high-level controllers can be constructed to include more functionality for more faithful imitation or more specific variations on the structure of the input motion. The structure encoded via classification is the basic template for the input motion. Given appropriate parameters, a specific instance of this template can produce an approximate reconstruction of input motion, whose faithfulness is bounded by the primitives in the vocabulary. However, other parameters can be used to adjust or create motions that are structurally similar variations of the input motion. Modifying motion in this manner is similar to work in *motion editing*, such as [109], for computer animation. The flexibility of a high-level controller to vary from the basic structure of an input motion is dependent on information used to augment the controller. This information includes *timing parameters* for inter-primitive transitioning, *indexing parameters* for accessing and extrapolating primitives, and *applicability information* describing pre- and post-conditions for primitives.

6.3 Summary

In this chapter, we illustrated the potential for applying automatically derived behavior vocabularies as basic capability modules for imitation by a humanoid agent. By using behaviors as predictors, primitive capabilities can provide a transparent substrate for a variety of other agent functions.

Chapter 7

Conclusion

In this dissertation, we have presented Performance-Derived Behavior Vocabularies (PDBV) as a methodology for automatically deriving a repertoire of skill capabilities for autonomous humanoid agents. Our methodology derives these skills from motion data of human performance, leveraging the structure underlying this motion in a data-driven manner. Behaviors underlying human motion data are uncovered through unsupervised learning, partially through our extension of Isomap for spatio-temporal data dependencies. Behaviors derived through unsupervised learning are realized as a modular set of exemplar-based behaviors that encode the nonlinear dynamics in the joint angle space of the agent. Modular behavior vocabularies derived by our methodology serve as a substrate to endow a humanoid agent with autonomy for a variety of functions, such as those used by constructing perceptual-motor algorithms.

Our contributions arising from this dissertation are summarized as follows:

1. automated grouping of free-space motion data into exemplars of underlying behaviors,

2. extension of geodesic-based multidimensional scaling for clustering behavior exemplars with spatio-temporal structure,

3. expression of exemplar-based behaviors as flowfields encapsulating nonlinear dynamics of an underlying skill, and

4. using the dynamics of skill behaviors for motion synthesis, classification, and imitation.

7.1 Avenues for Further Research

The PDBV methodology we have presented provides an automated data-driven means for modularizing motion into behaviors. For this dissertation, however, we have limited the scope of PDBV to free-space motion. This limitation has allowed us to disregard issues of *i)* incorporating sensory information and *ii)* handling potential physical interactions between the agent and its environment.

In order for a humanoid agent to be truly autonomous, it must be able to rely only on *local sensing* provided by its embodiment. In this dissertation, we have assumed that the agent is provided with a complete kinematic configurations from sensing. However, local sensing on a humanoid agent may provide more, less, or different information than described by kinematic configurations. This potential issue can be addressed through *sensory-motor coordination*, where sensory information and motor actuation are linked in some fashion. Sensory-motor questions for PDBV include: can *sensory-motor primitives*, similar to those described by Matarić [93], be automatically derived and what types of sensory data are required to derive these primitives in a data-driven fashion?

In addition to using local sensing, an autonomous humanoid agent is likely to be subject to external interactions with objects and structures in its environment. Primitive forward models derived by PDBV can provide control commands when the agent is subject to external interactions. However, these forward models encode free-space behavior and, thus, are likely to perform poorly in situations that include object interactions (e.g., tool manipulation, ball throwing), inter-agent interactions (e.g., competitive athletics, collaborative dancing), and static and dynamic balancing interactions (e.g., standing, walking, running, jumping). Similar to dealing with the problems of local sensing, dynamic interaction questions for PDBV include: can primitive behaviors be derived automatically and what information must be incorporated into the input data and/or derivation process?

Reference List

[1] R. O. Ambrose, H. Aldridge, R. S. Askew, R. R. Burridge, W. Bluethmann, M. Diftler, C. Lovchik, D. Magruder, and F. Rehnmark. Robonaut: Nasa's space humanoid. *IEEE Intelligent Systems*, 15(4):57–63, July-Aug. 2000.

[2] Okan Arikan and D. A. Forsyth. Interactive motion generation from examples. *ACM Transactions on Graphics (TOG)*, 21(3):483–490, 2002.

[3] Ronald C. Arkin. *Behavior-Based Robotics*. MIT Press, Cambridge, Massachusetts, USA, 1998.

[4] Francis R. Bach and Michael I. Jordan. Kernel independent component analysis. Technical Report UCB//CSD-01-1166, University of California, Berkeley, November 2001.

[5] A.J. Bell and T.J. Sejnowski. An information maximisation approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

[6] Asa Ben-Hur, David Horn, Hava T. Siegelmann, and Vladimir Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125–137, December 2001.

[7] Darrin C. Bentivegna, Ales Ude, Christopher G. Atkeson, and Gordon Cheng. Humanoid robot learning and game playing using pc-based vision. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2449–2454, Lausanne, Switzerland, October 2002.

[8] A. Billard and S. Schaal. A connectionist model for on-line robot learning by imitation. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS 2001)*, Maui, Hawaii, USA, 2001.

[9] Aude Billard and Maja J. Matarić. Learning human arm movements by imitation: Evaluation of a biologically inspired connectionist architecture. *Robotics and Autonomous Systems*, 37(2):145–160, Nov 2001.

[10] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, England, 1995.

[11] E. Bizzi, S. F. Giszter, E. Loeb, F. A. Mussa-Ivaldi, and P. Saltie. Modular organization of motor behavior in the frog's spinal cord. *Trends Neurosci.*, 18:442–446, 1995.

[12] R. K. Bock and W. Krischer. *The Data Analysis Briefbook*. Springer, Berlin; New York, 1998.

[13] I. Borg and P. Groenen. *Modern Multidimensional Scaling, Theory and Applications*. Springer-Verlag, New York, 1997.

[14] Jean-Yves Bouguet. Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/index.html.

[15] M. Brand. Charting a manifold. *Neural Information Processing Systems (NIPS'2002)*, 15, 2002.

[16] M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 994–999. IEEE Computer Society Press, 1997.

[17] Matthew Brand and Aaron Hertzmann. Style machines. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 183–192. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, July 2000. ISBN 1-58113-208-5.

[18] C. Bregler. Learning and recognizing human dynamics in video sequences. In *IEEE Conference on Computer Vision and Pattern Recoginition*, pages 568–574, San Juan, Puerto Rico, 1997.

[19] C. Bregler and S. M. Omohundro. Nonlinear manifold learning for visual speech recognition. In *IEEE International Conference on Computer Vision*, pages 494–499, Cambridge, MA, USA, 1995.

[20] Christoph Bregler and Jitendra Malik. Tracking people with twists and exponential maps. In *IEEE Conference on Computer Vision and Pattern Recoginition*, pages 8–15, Santa Barbara, CA, USA, 1998.

[21] R. A. Brooks. Intelligence without representation. *Artificial Intelligence Journal*, 47:139–159, 1991.

[22] G. Chappell and J. Taylor. The temporal kohonen map. *Neural Networks*, 6(3):441–445, 1993.

[23] G.K.M. Cheung, S. Baker, and T. Kanade. Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture. In *Proceedings of IEEE Computer Vision and Pattern Recognition (CVPR03)*, pages I: 77–84, Madison, Wisconsin, USA, 2003.

[24] Kong Man Cheung, Takeo Kanade, J.-Y. Bouguet, and M. Holler. A real time system for robust 3d voxel reconstruction of human motions. In *Proceedings of the 2000 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '00)*, volume 2, pages 714 – 720, Hilton Head, SC, USA, June 2000.

[25] Diane Chi, Monica Costa, Liwei Zhao, and Norman Badler. The emote model for effort and shape. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 173–182. ACM Press/Addison-Wesley Publishing Co., 2000.

[26] B. Chiu, E. Keogh, and S. Lonardi. Probabilistic discovery of time series motifs. In *To appear in the Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC, USA, August 2003.

[27] Chi-Wei Chu, Odest Chadwicke Jenkins, and Maja J Matarić. Markerless kinematic model and motion capture from volume sequences. In *Proceedings of IEEE Computer Vision and Pattern Recognition (CVPR 2003)*, pages II: 475–482, Madison, Wisconsin, USA, June 2003.

[28] Jonathan D. Cohen, Ming C. Lin, Dinesh Manocha, and Madhav K. Ponamgi. I-COLLIDE: An interactive and exact collision detection system for large-scale environments. In *Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 189–196, 218, 1995.

[29] Michael F. Cohen. Interactive spacetime control for animation. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 293–302. ACM Press, 1992.

[30] T. Cox and M. Cox. *Multidimensional Scaling*. Chapman and Hall, London, 1994.

[31] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley, Reading, MA, USA, 1989.

[32] J.E. Cutting, D.R. Proffitt, and L.T. Kozlowski. A biomechanical invariant for gait perception. *Journal of Experiemental Psychology: Human Perception and Performance*, 4(3):357–372, August 1978.

[33] "James Davis, Aaron Bobick, and Whitman Richards". "categorical representation and recognition of oscillatory motion patterns". In *"Proceedings of Computer Vision and Pattern Recognition"*, pages I:628–635, June 2000.

[34] Vin de Silva and Joshua B. Tenenbaum. Local versus global methods for nonlinear dimensionality reduction. *Advances In Neural Information Processing Systems*, 15, 2003.

[35] D. DeMers and G. Cottrell. Non-linear dimensionality reduction. In *Advances in Neural Information Processing Systems 5*. Morgan Kaufmann, 1993.

[36] J. Demiris and G. Hayes. Active and passive routes to imitation. In *In Proceedings of the AISB symposium on imitation in animals and artifacts.*, 1999.

[37] Yiannis Demiris and Gillian Hayes. Imitation as a dual-route process featuring predictive and learning components: a biologically-plausible computational model. In K. Dautenhahn and C. Nehaniv, editors, *Imitation in Animals and Artifacts*, chapter 13, pages 327–361. MIT Press, 2002.

[38] J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 126–133, Hilton Head, SC, USA, 2000.

[39] Evan Drumwright and Maja J. Matarić. Generating and recognizing free-space movement in humanoid robotics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, Oct 2003.

[40] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[41] D. Erol, J. Park, E. Turkay, K. Kawamura, O. Jenkins, and M. Matarić. Motion generation for humanoid robots with automatically derived behaviors. In *IEEE Systems, Man, and Cybernetics (SMC 2003)*, pages 1816–1822, Washington, DC, USA, October 2003.

[42] Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos. Composable controllers for physics-based character animation. In *Proceedings of ACM SIGGRAPH 2001*, pages 251–260, Los Angeles, CA, USA, August 2001.

[43] Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos. Autonomous reactive control for simulated humanoids. In *IEEE International Conference on Robotics and Automation (ICRA)*, Taipei, Taiwan, 2003.

[44] A. Fod, M. Matarić, and O. Jenkins. Automated derivation of primitives for movement classification. *Autonomous Robots*, 12(1):39–54, January 2002.

[45] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics — Principles and Practice*. The Systems Programming Series. Addison-Wesley, second edition in c edition, 1996.

[46] Alexandre R.J. Francois and Gérard G. Medioni. Adaptive color background modeling for real-time segmentation of video streams. In *Proceedings of the International on Imaging Science, Systems, and Technology*, pages 227–232, Las Vegas, NV, USA, June 1999.

[47] D.M. Gavrila and L.S. Davis. 3d model-based tracking of humans in action: A multi-view approach. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 73–80, San Francisco, CA, USA, 1996.

[48] M. Gleicher. Motion editing with spacetime constraints. In *SIGGRAPH 1997*, pages 139–148, 1999.

[49] R. L. Gorsuch. *Factor Analysis (2nd. ed.)*. Lawrence Erlbaum Associates, Hillsdale, N.J., 1983.

[50] Roderic A. Grupen, Manfred Huber, Jefferson A. Coelho Jr., and Kamal Souccar. A basis for distributed control of manipulation tasks. *IEEE Expert, Special Track on Intelligent Robotic Systems*, 10(2):9–14, 1995.

[51] S. Harnad. The symbol grounding problem. *Physica D*, 42:335–346, 1990.

[52] J.W. Harris and H. Stocker. *Handbook of Mathematics and Computational Science*. Springer-Verlag, New York, 1998.

[53] M. Haruno, D. M. Wolpert, and M. Kawato. Multiple forward-inverse models for human motor learning and control. In *Advances in Neural Information Processing Systems 11*, pages 31–37. MIT Press, 1999.

[54] T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84:502–516, 1989.

[55] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David E. Culler, and Kristofer S. J. Pister. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

[56] Adrian Hilton, Jonathan Starck, and Gordon Collins. From 3d shape capture to animated models. In *1st International Symposium on 3D Data Processing Visualization and Transmission (3DPVT'02)*, pages 246–257, Padova, Italy, Jun 2002.

[57] Geoffrey E. Hinton and Richard S. Zemel. Autoencoders, minimum description length and Helmholtz free energy. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 3–10. Morgan Kaufmann Publishers, Inc., 1994.

[58] A. Howard, M. J Matarić, and G. S. Sukhatme. An incremental deployment algorithm for mobile robot teams. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2849–2854, Lausanne, Switzerland, October 2002.

[59] Manfred Huber and Roderic A. Grupen. A hybrid architecture for learning robot control tasks. *Robotics Today*, 13(4), 2000.

[60] A. Hyvarinen. Survey on independent component analysis. *Neural Computing Surveys*, 2:94–128, 1999.

[61] M. Iacoboni, R. P. Woods, M. Brass, H. Bekkering, J. C. Mazziotta, and G. Rizzolatti. Cortical mechanisms of human imitation. *Science*, 286:2526–2528, 1999.

[62] Wayne Iba. Learning to classify observed motor behavior. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 1991)*, pages 732–738, 1991.

[63] J.A. Ijspeert, J. Nakanishi, and S. Schaal. Learning rhythmic movements by demonstration using nonlinear oscillators. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS 2002)*, pages 958–963, Lausanne, Switzerland, October 2002.

[64] J.A. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings of the International Conference on Robotics and Automation (ICRA2002)*, Washington, D.C., USA, May 2002.

[65] Credo Interactive Inc. Mega mocap v2. http://www.credo-interactive.com/press/megav2.html.

[66] Measurand Inc. Shapetape. http://www.measurand.com.

[67] Polhemus Inc. Fastrak. http://www.polhemus.com.

[68] A.K. Jain and R.C. Dubes. *Algorithms For Clustering Data*. Prentice Hall, Englewood Cliffs, NJ, USA, 1988.

[69] Doug L. James and Kayvon Fatahalian. Precomputing interactive dynamic deformable scenes. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003)*, 22(3):879–887, 2003.

[70] O. C. Jenkins, M. Matarić, and S. Weber. Primitive-based movement classification for humanoid imitation. In *Proceedings of the First IEEE-RAS International Conference on Humanoid Robots (Humanoids 2000)*, Cambridge, MA, USA, 2000.

[71] Odest Chadwicke Jenkins and Maja J Matarić. Deriving action and behavior primitives from human motion data. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-2002)*, volume 3, pages 2551–2556, Lausanne, Switzerland, October 2002.

[72] Odest Chadwicke Jenkins and Maja J Matarić. Automated derivation of behavior vocabularies for autonomous humanoid motion. In *Proceedings of Autonomous Agents and Multiagent Systems (AAMAS 2003)*, pages 225–232, Melbourne, Australia, July 2003.

[73] M. I. Jordan. Computational aspects of motor control and motor learning. In H. Heuer and S. Keele, editors, *Handbook of Perception and Action: Motor Skills*. Academic Press, New York, 1996.

[74] M. Kallmann, A. Aubel, T. Abaci, and D. Thalmann. Planning collision-free reaching motions for interactive object manipulation and grasping. In *To appear in the Proceedings of Eurographics*, 2003.

[75] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[76] Balazs Kégl, Adam Krzyzak, Tamas Linder, and Kenneth Zeger. Learning and design of principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3):281–297, 2000.

[77] Teuvo Kohonen. *Self-Organizing Maps*, volume 30. Springer, Berlin, Heidelberg, New York, 1997.

[78] Lucas Kovar and Michael Gleicher. Flexible automatic motion blending with registration curves. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 214–224, San Diego, California, USA, 2003. Eurographics Association.

[79] Lucas Kovar, Michael Gleicher, and Frdric Pighin. Motion graphs. *ACM Transactions on Graphics (TOG)*, 21(3):473–482, 2002.

[80] J.J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue. Motion planning for humanoid robots under obstacle and dynamic balance constraints. In *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA'2001)*, pages 692– 698, Seoul, Korea, May 2001.

[81] Andrew M. Ladd, Kostas E. Bekris, Algis Rudys, Guillaume Marceau, Lydia E. Kavraki, and Dan S. Wallach. Robotics-based location sensing using wireless Ethernet. In *Proceedings of The Eighth ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Atlanta, GA, USA, September 2002.

[82] Jeff Lander. Working with motion capture file formats. *Game Developer Magazine*, pages 30–37, January 1998.

[83] K. J. Lang, A. H. Waibel, and G. E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3(1):33–43, 1990.

[84] S.M. LaValle and J.J Kuffner. Rapidly-exploring random trees: Progress and prospects. In *Robotics: The Algorithmic Perspective. 4th Int'l Workshop on the Algorithmic Foundations of Robotics*, pages 293–308, Hanover, NH, USA, March 2000. A.K. Peters.

[85] Steve Lawrence, A.C. Tsoi, and A.D. Back. Function approximation with neural networks and local methods: Bias, variance and smoothness. In Peter Bartlett, Anthony Burkitt, and Robert Williamson, editors, *Australian Conference on Neural Networks*, pages 16–21. Australian National University, 1996.

[86] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics (TOG)*, 21(3):491–500, 2002.

[87] Yan Li, Tianshu Wang, and Heung-Yeung Shum. Motion texture: a two-level statistical model for character motion synthesis. *ACM Transactions on Graphics (TOG)*, 21(3):465–472, 2002.

[88] Jason Luck, Dan Small, and Charles Q. Little. Real-time tracking of articulated human models using a 3d shape-from-silhouette method. In *Robot Vision, International Workshop RobVis*, volume 1998, pages 19–26, Feb 2001.

[89] David B. MacKay. Probabilistic multidimensional scaling: An anisotropic model for distance judgments. *Journal of Mathematical Psychology*, 33:187–205, June 1989.

[90] M. J Matarić. Behavior-based control: Examples from navigation, learning, and group behavior. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2-3):323–336, 1997.

[91] M. J. Matarić, V. B. Zordan, and Z. Mason. Movement control methods for complex, dynamically simulated agents: Adonis dances the macarena. In *Autonomous Agents*, pages 317–324, Minneapolis, St. Paul, MI, 1998. ACM Press.

[92] Maja J. Matarić. Behavior-based robotics. In Robert A. Wilson and Frank C. Keil, editors, *MIT Encyclopedia of Cognitive Sciences*, pages 74–77. MIT Press, Apr 1999.

[93] Maja J Matarić. Sensory-motor primitives as a basis for imitation: Linking perception to action and biology to robotics. In Chrystopher Nehaniv and Kerstin Dautenhahn, editors, *Imitation in Animals and Artifacts*, pages 392–422. MIT Press, 2002.

[94] Maja J Matarić, Victor B. Zordan, and Z. Mason. Movement control methods for complex, dynamically simulated agents: Adonis dances the macarena. In *Autonomous Agents and Multi-Agent Systems*, pages 317–324, Minneapolis, Minnesota, USA, 1998. ACM Press.

[95] Maja J Matarić, Victor B. Zordan, and Matthew Williamson. Making complex articulated agents dance: An analysis of control methods drawn from robotics, animation, and biology. *Autonomous Agents and Multi-Agent Systems*, 2(1):23–44, March 1999.

[96] Ivana Mikić, Mohan Trivedi, Edward Hunter, and Pamela Cosman. Articulated body posture estimation from multi-camera voxel data. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 455–460, Kauai, HI, USA, December 2001.

[97] H. Miyamoto and M. Kawato. A tennis serve and upswing learning robot based on bi-directional theory. *Neural Networks*, 11:1331–1344, 1998.

[98] H. Miyamoto, S. Schaal, F. Gandolfo, Y. Koike, R. Osu, E. Nakano, Y. Wada, and M. Kawato. A kendama learning robot based on bi-directional theory. *Neural Networks*, 9:1996, 1281-1302.

[99] T. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268, March 2001.

[100] L. Molina-Tanco and A. Hilton. Realistic synthesis of novel movements from a database of motion capture data. In *Proceedings of IEEE Workshop on Human Motion (HUMO 00)*, pages 137–142, Austin, Texas, USA, December 2000.

[101] Andrew Moore, C. G. Atkeson, and S. A. Schaal. Memory-based learning for control. Technical Report CMU-RI-TR-95-18, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, April 1995.

[102] Meta Motion. Gypsy 3.0. http://www.metamotion.com.

[103] Klaus-Robert Müller, Sebastian Mika, Gunnar Rätsch, Koji Tsuda, and Bernhard Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 42:181–202, 2001.

[104] F. A. Mussa-Ivaldi. Nonlinear force fields: A distributed system of control primitives for representation and learning movements. In *Proc. of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 84–90. IEEE Computer Society Press, 1997.

[105] Monica Nicolescu and Maja J Matarić. A hierarchical architecture for behavior-based robots. In *First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 227–233, Bologna, Italy, July 2002.

[106] Simon G Penny, Jeffrey Smith, and Andre Bernhardt. Traces: Wireless full body tracking in the cave. In *Ninth International Conference on Artificial Reality and Telexistence (ICAT'99)*, December 1999.

[107] Richard Alan Peters and Christina L. Campbell. Robonaut learning through teleoperation: Automatic acquisition of trajectories for articulated motion. Unpublished report, August 2002.

[108] Paolo Pirjanian. Behavior coordination mechanisms - state-of-the-art. Technical report, Institute for Robotics and Intelligent Systems Technical Report IRIS-99-375, 1999.

[109] Zoran Popović and Andrew P. Witkin. Physically based motion transformation. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 11–20, Los Angeles, California, August 1999. ACM SIGGRAPH / Addison Wesley Longman.

[110] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *Proceedings of Mobile Computing and Networking*, pages 32–43, 2000.

[111] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–285, February 1989.

[112] Amit Ramesh and Maja J. Matarić. Learning movement sequences from demonstration. In *Proceedings of the International Conference on Development and Learning*, pages 203–208, MIT, Cambridge, MA, Jun 2002.

[113] J. Rittscher, A. Blake, and S.J. Roberts. Towards the automatic analysis of complex human body motions. *IVC*, 20(12):905–916, October 2002.

[114] Jens Rittscher and Andrew Blake. Classification of human body motion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 634–639, Kerkyra, Corfu, Greece, September 1999.

[115] G. Rizzolatti, L. Gadiga, V. Gallese, and L. Fogassi. Premotor cortex and the recognition of motor actions. Cognitive Brain Research, 3:131–141, 1996.

[116] Charles Rose, Michael F. Cohen, and Bobby Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics & Applications*, 18(5):32–40, September - October 1998. ISSN 0272-1716.

[117] M.T. Rosenstein and A.G. Barto. Supervised learning combined with an actor-critic architecture. Technical Report Technical Report 02-41, Department of Computer Science, University of Massachusetts, Amherst, 2002.

[118] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[119] T. D. Sanger. Human arm movements described by a low-dimensional superposition of principal component. *Journal of NeuroScience*, 20(3):1066–1072, Feb 1 2000.

[120] S. Schaal. Is imitation learning the route to humanoid robots. *Trends in Cognitive Sciences*, 3(6):233–242, 1999.

[121] S. Schaal, A. Ijspeert, and A. Billard. Computational approaches to motor learning by imitation. *Philosophical Transaction of the Royal Society of London: Series B, Biological Sciences*, 358(1431):537–547, 2003.

[122] S. Schaal and D. Sternad. Programmable pattern generators. In *Proceedings of the Third International Conference on Computational Intelligence in Neuroscience*, pages 48–51, 1998.

[123] Stefan Schaal. Dynamic movement primitives - a framework for motor control in humans and humanoid robots. In *Proceedings of the International Symposium on Adaptive Motion of Animals and Machines*, 2003.

[124] Arno Schödl, Richard Szeliski, David H. Salesin, and Irfan Essa. Video textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 489–498. ACM Press/Addison-Wesley Publishing Co., 2000.

[125] Bernhard Schölkopf, Alex J. Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[126] Steven M. Seitz and Charles R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1067–1073, San Juan, Puerto Rico, 1997.

[127] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the ACM national conference*, pages 517–524. ACM Press, 1968.

[128] P. Sloan, C. F. Rose, and M. F. Cohen. Shape and animation by example. Technical Report MSR-TR-2000-79, Microsoft Research, 2000.

[129] M. Steyvers. Multidimensional scaling. In *Encyclopedia of Cognitive Science*. Nature Publishing Group, London, UK.

[130] Vicon Motion Systems. http://www.vicon.com.

[131] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

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

[133] Michael E. Tipping and Christopher M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11:443–482, 1999.

[134] Munetoshi Unuma, Ken Anjyo, and Ryozo Takeuchi. Fourier principles for emotion-based human figure animation. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 91–96. ACM Press, 1995.

[135] M. Varsta, J. Heikkonen, J. Lampinen, and J. del R. Milln. Neural processing letters. *Temporal Kohonen Map and the Recurrent Self-Organizing Map: Analytical and Experimental Comparison*, 13:237–251, 2001.

[136] S. Vijayakumar and S. Schaal. An o(n) algorithm for incremental real time learning in high dimensional space. In *Proc. of the Seventeenth International Conference on Machine Learning*, pages 1079–1086, Stanford, California, 2000.

[137] J. Wang and B. Bodenheimer. An evaluation of a cost metric for selecting transitions between motion segments. In *Proceedings of 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 232–238, San Diego, California, USA, July 2003.

[138] Kamin Whitehouse. The design of calamari: an ad-hoc localization system for sensor networks. Master's thesis, University of California at Berkeley, 2002.

[139] Christopher K. I. Williams. On a connection between kernel pca and metric multidimensional scaling. *Machine Learning*, 46(1-3):11–19, 2002.

[140] M. Williamson. Postural primitives: Interactive behavior for a humanoid robot arm. In *Fourth International Conference on Simulation of Adaptive Behavior*, pages 124–131, Cape Cod, MA, USA, 1996. MIT Press.

[141] M. Williamson. *Robot Arm Control Exploiting Natural Dynamics*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1999.

[142] Andrew Witkin and Michael Kass. Spacetime constraints. In *Computer Graphics (Proc. SIGGRAPH '88)*, volume 22, pages 159–168, 1988.

[143] D. M. Wolpert and M. Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11(7-8):1317–1329, 1998.

[144] Christopher Richard Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

[145] L. Goncalves Y. Song and P. Perona. Unsupervised learning of human motion. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(7):814–827, 2003.

[146] T. Zhao, T. Wang, and H. Shum. Learning highly structured motion model for 3d human tracking. In *Proceedings of Asian Conference on Computer Vision (ACCV'02)*, Melbourne, Australia, 2002.

[147] Yong Zhou and Arthur W. Toga. Efficient skeletonization of volumetric objects. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):196–209, July-September 1999.

[148] V. B. Zordan and J. K. Hodgins. Tracking and modifying upper-body human motion data with dynamic simulation. In *Computer Animation and Simulation '99, Eurographics Animation Workshop*, pages 13–22. Springer-Verlag, Sept. 1999.

Appendix A

Collecting Natural Human Performance

A significant problem encountered during the course of this dissertation is the substantial lack of available and usable motion data. Motion data serve as input into PDBV and, thus, is a critical component for endowing a humanoid agent with basic capabilities. However, such data have not been readily available for general use. In addition, motion from human performance is typically limited to scripted behaviors and structured situations and is not typical of naturally occurring behaviors. In this chapter, we present two approaches we developed aimed at collecting motion from naturally occurring human performance. These approaches strive for uninstrumented subjects that are model-free, making no assumptions about the structure of the subject.

The author is the primary developer of these approaches to motion capture; Chi-Wei Chu was the primary implementer and secondary developer.

A.1 Kinematic Model and Motion Capture

The ability to collect human motion data are invaluable for applications such as computer animation, activity recognition, human-computer interfaces, and humanoid robot control and teleoperation. This fact is evidenced by the increasing amount of research geared towards developing and utilizing *motion capture* technologies. Typical motion capture mechanisms require that the subject be instrumented with several beacons or markers. The motion of the subject is then reconciled from the sensed positions and/or orientations of the markers. However, such systems can:

1. be prohibitively expensive;
2. require subjects to be instrumented with cumbersome markers;
3. greatly restrict the volume of capture;
4. have difficulty assigning consistent labels to occluding markers;
5. have difficulty converting marker data into kinematic motion.

An emerging area of research suited to address these problems involves uninstrumented capture of motion, or *markerless motion capture*. For markerless motion capture, subject data are acquired through some passive sensing mechanism and then reconciled

into kinematic motion. Several *model-based* markerless capture approaches [38, 96, 20, 47, 24, 88, 144, 56] have been proposed that assume an *a priori* kinematic or body model. However, it would be preferable to eliminate this model dependence to capture both the subject's motion and kinematic model and, thus, perform *model and motion capture*.

We developed a solution for model-free vision-based markerless motion capture of subjects with tree-structured kinematics from multiple calibrated cameras. Using the functional structure of a motion capture system described by Moeslund and Granum [99], we summarize our approach for markerless motion capture. Moeslund and Granum describe a motion capture system as consisting of four components: initialization, tracking, pose estimation, and recognition. For initialization, a set of cameras is calibrated, using a method such as Bouguet's [14]. Because we assume no *a priori* kinematic model, no model initialization is necessary. We assume for the tracking component a system capable of capturing an individual subject's movement over time as a volume sequence, such as [106, 126]. The pose estimation component we developed is more than pure pose estimation because it performs model and motion capture. In this component, we perform automatic model and posture estimation for each frame in the volume sequence. The models and postures produced from each frame are aligned in a second pass to determine a common kinematic model across the volume sequence. The common kinematic model is then applied to each frame in the volume sequence to perform pose estimation with respect to a consistent model. Our current methodology for capture does not include a recognition component. However, we envision our capture system providing vast amounts of motion data for other uses. For instance, Jenkins and Matarić [72] require long streams of motion data as demonstrations for automatically deriving vocabularies of behaviors and controllers for humanoid robot control.

Central to our model and motion capture approach is the ability to estimate a kinematic model and its posture from a subject's volume in a single frame. Towards this end, we developed a model-free method, called *nonlinear spherical shells (NSS)*, for extracting *skeleton point features* that are linked into a tree-structured *skeleton curve* for a particular frame within a motion. A skeleton curve is an approximation of the "underlying axes" of a subject, similar to principal curves [54], the axis of a generalized cylinder, or the wire spine of a posable puppet. NSS works by accentuating the underlying axes of a volume through Isomap nonlinear dimension reduction [131] and traversing the resulting "Da Vinci"-like posture. Isomap essentially eliminates the nonlinearities caused by joint rotations. Using skeleton curve provided via NSS, we automatically estimate the tree-structured kinematics and posture of the volume.

Several advantages arise in using our approach for markerless motion capture. First, our method is fast and accurate enough to be tractably applied to all frames in a motion. Our method can be used alone or as an initialization step for model-based capture approaches. Second, our dependence on modeling human bodies is eliminated. Automated model derivation is especially useful when the subject's kinematics differ from standard human kinematics due to missing limbs or objects the subject is manipulating. Third, the posture of the human subject is automatically determined without complicated label assignments.

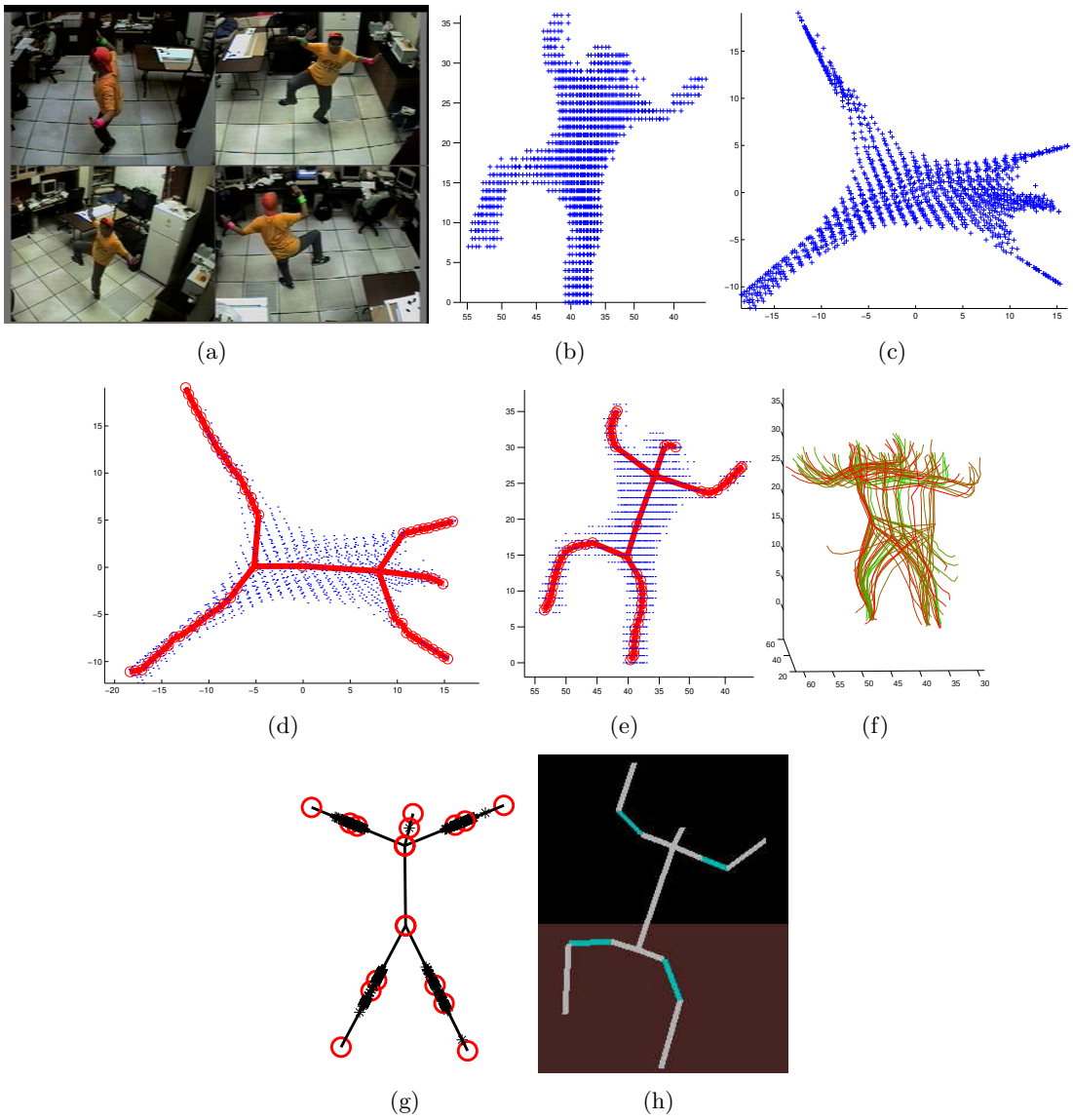


Figure A.1: An illustrated outline of our approach. (a) A subject viewed in multiple cameras over time is used to build (b) a Euclidean space point volume sequence. Postures in each frame are estimated by: transforming the subject volume (c) to an intrinsic space pose-invariant volume, finding its (d) principle curves, project the principal curves to a (e) skeleton curve, and breaking the skeleton curve into a kinematic model. (f) Kinematic models for all frames are (g) aligned to find the joints for a normalized kinematic model. The normalized kinematic model is applied to all frames in the volume sequence to estimate its (h) motion, shown from an animation viewing program.

A.1.1 Volume Sequence Capture

The volume sequence data used for this work came from two sources. One source of captured volume data are from real-world subjects (humans) viewed by multiple cameras. The other source was from synthetically generated volume data using an articulated 3D geometry with arbitrary kinematics.

For real-world volume capture, we used an existing volume capture technique for multiple calibrated cameras. While not the focus of our work, this implementation does provide an adequate means for collecting volume sequences. The implementation is derived from the work of Penny et. al. [106] for real-time volume capture; however, several other approaches are readily available (e.g., [126, 24]). The capture approach is a basic brute-force method that checks each element of a voxel grid for inclusion in the point volume. In our capture setup, we place multiple cameras around three sides of a hypothetical rectangular volume, such that each camera can view roughly all of the volume. This rectangular volume is a voxel grid that divides the space in which moving objects can be captured.

The intrinsic and extrinsic calibration parameters for the cameras are extracted using a camera calibration toolbox designed by [14]. The parameters from calibration allow us to precompute a look-up table for mapping a voxel to pixel locations in each camera. For each frame in the motion, silhouettes of foreground objects in the capture space are segmented within the image of each camera and used to carve the voxel grid. A background subtraction method proposed in [46] was used. It can then be determined if each voxel in the grid is part of a foreground object by counting and thresholding the number of camera images in which it is part of a silhouette. One set of volume data are collected for each frame (i.e., set of synchronized camera images) and stored for off-line processing.

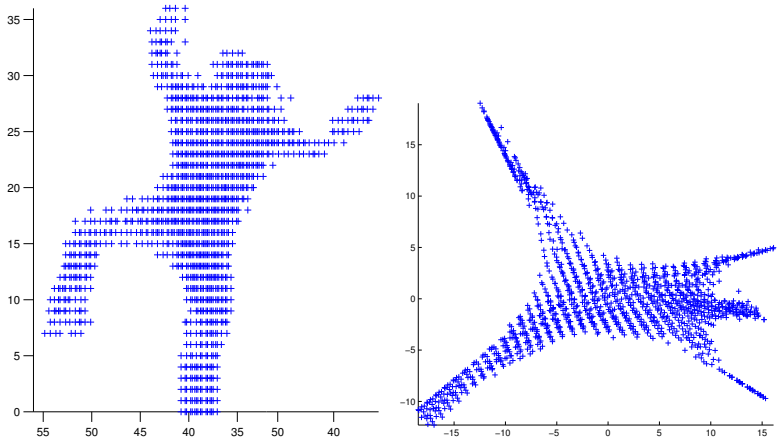
For synthetic data, we artificially created motion sequences from a synthetic articulated object with arbitrary tree-structured kinematics. We used this data to test our approach for objects readily available or controllable in the real world. In creating this data, we manually specified the kinematic model, rigid body geometries (cylinders), and joint angle trajectories. The motion of the object was converted into a volume sequence by scan converting each frame according to a voxel grid.

A.1.2 Nonlinear Spherical Shells

Nonlinear spherical shells (NSS) is our model-free approach for extracting a skeleton curve feature from a Euclidean-space volume of points. For NSS, we assume that nonlinearity of rigid-body kinematic motion is introduced by rotations about the joint axes. By removing these joint nonlinearities, we can trivially extract skeleton curves.

Recent work on manifold learning techniques, summarized in Chapter 2, has produced methods capable of uncovering nonlinear structure from spatial data that can also be applied for this problem. Isomap, in particular, has been demonstrated to extract meaningful nonlinear representations for high dimensional data such as images of handwritten digits, natural hand movements, and a pose-varying human head.

The procedure for (NSS) works in three main steps:



(a) (b)

Figure A.2: (a) A captured human volume in Euclidean space and (b) its pose-invariant intrinsic space representation.

1. removal of *pose-dependent* nonlinearities from the volume by transforming the volume into an intrinsic space using Isomap;

2. dividing and clustering the *pose-independent* volume such that principal curves are found in intrinsic space;

3. project points defining the intrinsic space principal curve into the original Euclidean space to produce a skeleton curve for the volume.

We applied Isomap in the first step of the NSS procedure to remove pose nonlinearities from a set of points compromising the captured human in Euclidean space. We used the implementation provided by the authors of Isomap (available at http://isomap.stanford.edu/) and applied it directly to the volume data. Isomap requires the user to specify only the number of dimensions for the intrinsic space and how to construct local neighborhoods for each data point. Because dimension reduction is not our aim, the intrinsic space is set to have 3 dimensions. Each point determines other points within its local neighborhood using k-nearest neighbors or an epsilon sphere with a chosen radius.

Our application of Isomap transforms the volume points into a pose-independent arrangement in the intrinsic space. The pose-independent arrangement is similar to a *"Da Vinci"* pose in 3 dimensions (Figure A.2). Isomap can produce the Da Vinci point arrangement for any point volume with distinguishable limbs.

The next step in the NSS procedure is processing intrinsic space volume for principal curves. The definition of principal curves can be found in [54] or [76] as "self-consistent" smooth curves that pass through the "middle" of a d-dimensional data cloud, or nonlinear principal components. While smoothness is not our primary concern, we are interested in placing a curve through the "middle" of our Euclidean space volume. Depending on the

posture of the human, this task can be difficult in Euclidean space. However, the pose-invariant volume provided by Isomap makes the extraction of principal curves simple, due to properties of the intrinsic space volume. Isomap provides an intrinsic space volume that is mean-centered at the origin and has limb points that extend away from the origin.

Points on the principle curves in intrinsic space be found by the following subprocedure (Figure A.4):

1. partitioning the intrinsic space volume points into *concentric spherical shells*;
2. clustering the points in each partition;
3. averaging the points of each cluster to produce a principal curve point;
4. linking principal curve points with overlapping clusters in adjacent spherical shells.

Clustering used for each partition was developed from the one-dimensional "sweep-and-prune" technique, described by Cohen et al. [28], for finding clusters bounded by axis-aligned boxes. This clustering method requires specification of a separating distance threshold for each axis rather than the expected number of clusters. The result from the principal curves procedure is a set of points defining the principal curves linked in a hierarchical tree-structure. These include three types of *indicator nodes*: a *root node* located at the mean of the volume, *branching nodes* that separate into articulations, and *leaf nodes* at terminal points of the body.

The final step in the NSS procedure projects the intrinsic space principal curve points onto a skeleton curve in the original Euclidean space. We used Shepards interpolation [127] to map principal curve points onto the Euclidean space volume, producing skeleton curve points. The skeleton curve is formed by reapplying the tree-structured linkages of the intrinsic space principal curves to the skeleton curve points.

Other methods for volume skeletonization are available. These approaches include the distance coding [147], boundary peeling [147], and self-organizing feature maps [10]. For our purposes, it is important to ensure that the skeletonization produces a bordered 1-manifold, not necessarily a medial axis that is potentially a 2-manifold.

The skeleton curve found by the NSS procedure will be indicative of the underlying spatial structure of the Euclidean space volume, but may contain a few undesirable artifacts. We handled these artifacts using a skeleton curve refinement procedure. The refinement procedure first eliminates *noise branches* in the skeleton curve that typically occur in areas of small articulation, such as the hands and feet. Noise branches are detected as branches with depth under some threshold. A noise branch is eliminated through merging its skeleton curve points with a non-noise branch.

The refinement procedure then eliminates noise for the root of the skeleton curve. Shell partitions around the mean of the body volume will be encompassed by the volume (i.e., contain a single cluster spread across the shell). The skeleton curve points for such partitions will be roughly located near the volume mean. These skeleton curve points are merged to yield a new root to the skeleton curve. The result is a skeleton curve having a root and two or more immediate descendants.

The minor variations in the topology of the skeleton curve are then eliminated by merging adjacent branching nodes. These are two skeleton points on adjacent spherical

shells with adjacent clusters that both introduce a branching of the skeleton curve. The branches at these nodes are assumed to represent the same branching node. Thus, the two skeleton points are merged into a single branching node.

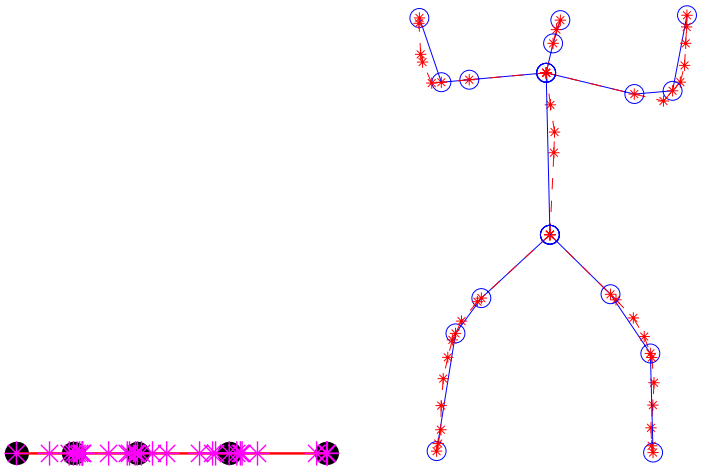
A.1.3 Model and Motion Capture

In this section, we describe the application of NSS within the context of our approach for markerless model and motion capture. The model and motion capture (MMC) procedure automatically determines a common kinematic model and joint angle motion from a volume sequence in a three-pass process. In the first pass, the procedure applies NSS independently to each frame in the volume sequence. From the skeleton curve and volume of each frame, a kinematic model and posture is produced that is specific to the frame. A second pass across the specific kinematic models of each frame is used to produce a single normalized kinematic model with respect to the frames in the volume sequence. Finally, the third pass applies the normalized model to each volume and skeleton curve in the sequence to produce estimated posture parameters.

The described NSS procedure is capable of producing skeleton curve features in a model-free fashion. The skeleton curve is used to derive a kinematic model for the volume in each frame. First, we consider each branch (occurring between two indicator nodes) as a kinematic link. The root node and all branching nodes are classified as joints. Each branch is then segmented into smaller kinematic links based on the curvature of the skeleton curve. This division is performed by starting at the parent indicator node and iteratively including skeleton points until the corresponding volume points become nonlinear. Nonlinearity is tested by applying a threshold to the skewness of the volume points with respect to the line between the first and last included skeleton point. When the nonlinearity occurs, a segment, representing a joint placement, is set at the last included skeleton point. The segment then becomes the first node in the determination of the next link and the process iterates until the next indicator node is reached. The length of these segments, relative to the length of the whole branch, is recorded in the branch. The specific kinematic models derived from the volume sequence may have different branch lengths and each branch may be divided into a different number of links.

In the second pass, a normalization procedure is used across all frame-specific models to produce a common model for the sequence. For normalization, we aim to align all specific models in the sequence and look for groupings of joints. The alignment method we used iteratively collapsed two models in subsequent frames using a matching procedure to find correspondences. The matching procedure uses summed error values of minimum squared distance between branch parents, the difference between angles of branches, and the difference between branch lengths. The normalization procedure finds the mapping that minimizes the total error value. We have also begun to experiment with a simpler alternative alignment procedure. This procedure uses Isomap to align by constructing neighborhoods for each skeleton point that considers its intra-frame skeleton curve neighbors and corresponding points on the skeleton curve in adjacent frames.

Once the specific kinematic models are aligned, clustering on each branch is performed to identify joint positions. Each branch is normalized by averaging the length of the branch and number of links in the branch. The location of the aligned joint locations



(a) (b)

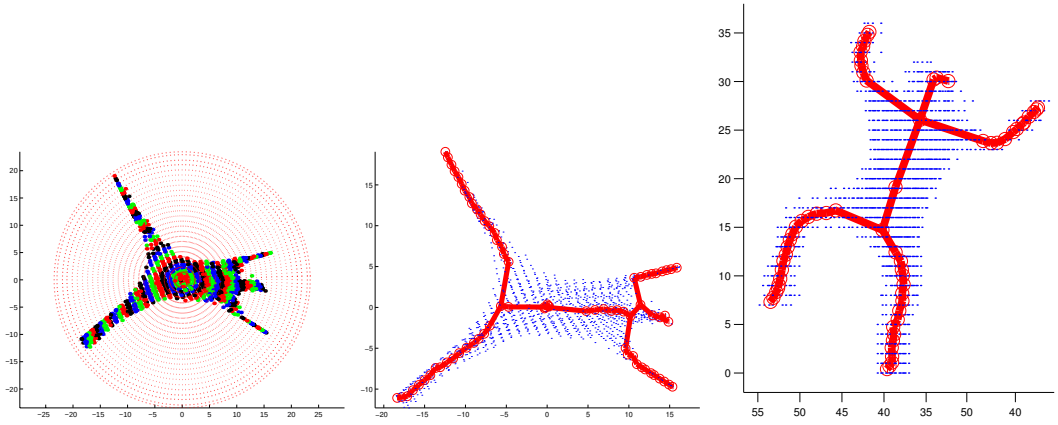
Figure A.3: (a) Aligned segmentation points (as stars) and joints clusters (as circles) of one of the branches in the synthetic data. (b) The normalized kinematic model (circles as joints) with respect to the aligned skeleton curve sequence.

along the branch forms a 1D data sequence. An example is shown (Figure A.3) for a branch with an average number of joints rounded to three. In this figure, the joint positions roughly form three sparse clusters of joint points along the branch, with some outliers. To identify the joint clusters, we used a clustering method that estimates density of all joint locations and places a joint cluster where peaks in the density are found.

In the third pass, the common kinematic model is applied to the skeleton curve in each frame to find the motion of the model (Figure A.3). The coordinate system of the root node of the model is always aligned to the world coordinate system. For every joint, the direction of the link is the Z axis of the joint's coordinate system. The Y axis of the joint is derived by the cross product of its Z axis and its parent's X axis. The cross product of the Y and Z axis is the X axis of the joint. The world space coordinate system for each joint is converted to a local coordinate system by determining its 3D rotational transformation from its parent. The set of these 3D rotations provides the joint angle configuration for the current posture of the derived model.

A.1.4 Results and Observations

In this section, we describe the implementation of our markerless model and motion capture approach and the results from its application to both captured human volume data and synthetic data. The human volume data contain two different motion sequences: waving and jumping jacks. Our approach was implemented in Matlab, with our volume



(a) (b) (c)

Figure A.4: (a) Partitioning of the pose-invariant volume, (b) its tree-structured principal curves, (c) and project back into Euclidean space.

capture implementation in Microsoft Visual C++. The execution of the entire implementation was performed on a 350 MHz Pentium with 128 MB of memory.

For each human motion sequence, a volume sequence was captured and stored for off-line processing by the model and motion capture procedure. Using the Intel Image Processing Library, we were able to capture volumes within a $80 \times 80 \times 50$ grid of cubic $50mm^3$ voxels at 10 Hz. Each volume sequence consisted of roughly 50 frames. Due to our frugal choices for camera and framegrabber options, our ability to capture human volumes was significantly restricted. Our image technology allowed for 320×240 image data from each camera, which produced several artifacts such as incorrectly activated voxels from shadows, occlusion ghosting, and image noise. This limitation restricted our capture motions to exaggerated, but usable, motion, where the limbs were very distinct from each other. Improving our proof-of-concept volume capture system, with more and better cameras, lighting, and computer vision techniques, will vastly improve our capture system, without having to adjust the model and motion capture procedure.

Using the captured volume sequences, our model and motion capture mechanism was able to accurately determine appropriate postures for each volume without fail. We used the same user parameters for each motion, consisting of an Isomap epsilon-ball neighborhood of radius $(50mm^3)^{1/2}$ and 25 for the number of concentric sphere partitions. In addition to accurate postures, the derived kinematic model parameters for each sequence appropriately matched the kinematics of the capture subject. However, for camera captured volume data, a significant amount of noise occurred between subsequent frames in the produced motion sequence. Noise is typical for many instrumented motion capture systems and should be expected when independently processing frames for temporally dependent motion. We were able to clean up this noise to produce aesthetically viable motion using standard low pass filtering.

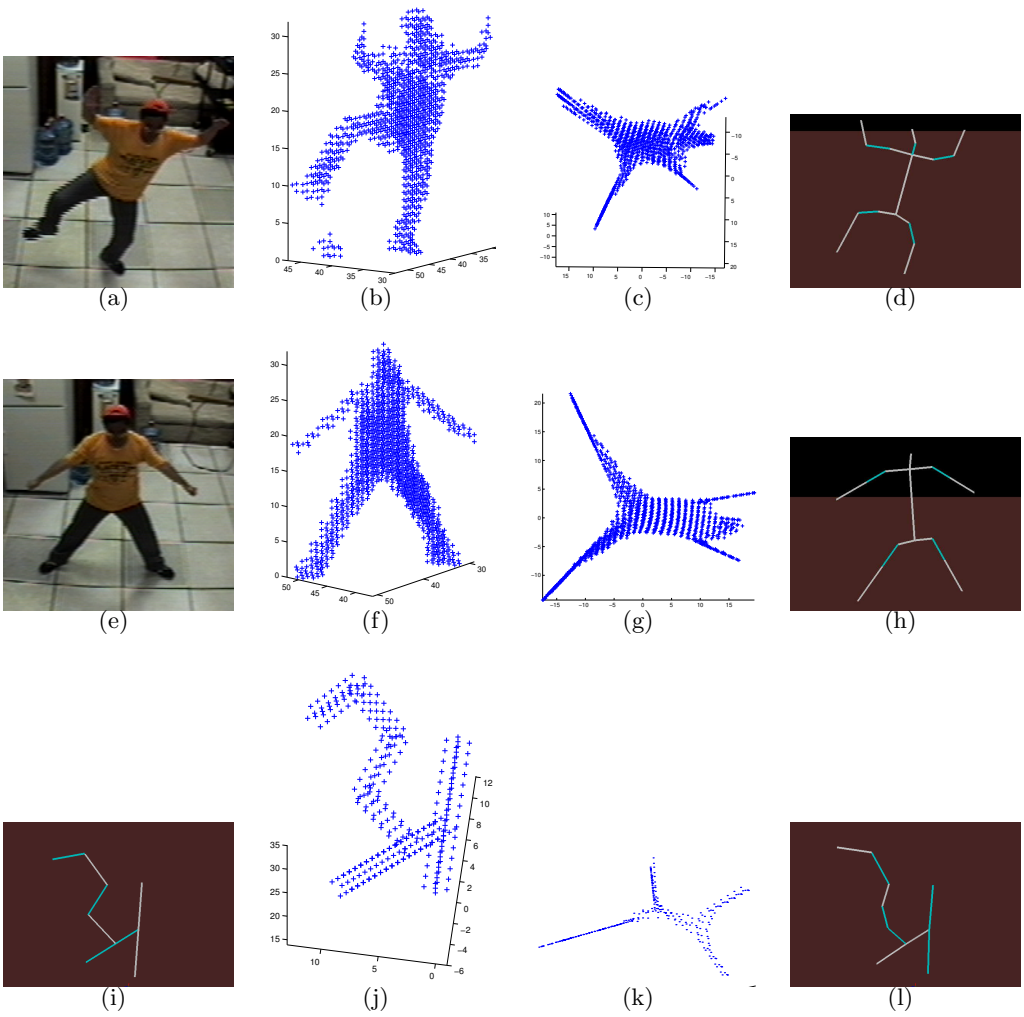


Figure A.5: Results from producing kinematic motion for human waving, jumping jacks and synthetic object motion (rows). The results are shown as a snapshot of the performing human or object, the capture or generated point volume data, the pose-invariant volume, and the derived kinematic posture (columns).

When applied to synthetic data, our method can reconstruct its original kinematic model with reasonable accuracy. This data were subject to the problem of over-segmentation, i.e., joints are placed where there is in fact only one straight link. There are three causes for this problem. First, a joint will always be placed at branching nodes in the skeleton curves. A link will be segmented if another link is branching from its side. Second, the root node of the skeleton curve is always classified as a joint, even if it is placed in the middle of an actual link. Third, noise in the volume data may add fluctuation of the skeleton curves and cause unwanted segments.

Motions were output to files in the Biovision BVH motion capture format. Figure A.5 shows the kinematic posture output for each motion.

In observing the performance of our markerless model and motion capture system, several benefits of our approach became evident. First, the relative speed of our capture procedure made the processing of each frame of a motion tractable. Depending on the number of volume points, the elapsed time for producing a posture from a volume by our Matlab implementation ranged between 60 and 90 seconds, with approximately 90 percent of this time spent for Isomap processing. Further improvements can be made to our implementation to speed up the procedure and process volumes with increasingly finer resolution. Second, our implementation required no explicit model of human kinematics, no initialization procedure, and no optimization of parameters with respect to a volume. Our model-free NSS procedure produced a representative skeleton curve description of a human posture based on the geometry of the volume. Lastly, the skeleton curve may be a useful representation of posture in and of itself. Rigid-body motion is often represented through typically model-specific kinematics. Instead, the skeleton curve may allow for an expression of motion that can be shared between kinematic models, for purposes such as robot imitation.

A.1.5 Extensions for Continuing Work

Using our current work as a platform, we aim to improve our ability to collect human motion data in various scenarios. Motion data are critically important for other related projects, such as the derivation of behavior vocabularies [72]. Areas for further improvements to our capture approach include: *i)* more consistent mechanism for segmenting skeleton curve branches, *ii)* different mechanisms for aligning and clustering joints from specific kinematic models in a sequence, *iii)* automatically deriving kinematic models and motion for kinematic topologies containing cycles (i.e., "bridges", volumes of genus greater than zero), *iv)* and exploring connections between model-free methods for robust model creation and initialization and model-based methods for robust temporal tracking, *v)* extensions to Isomap for volumes of greater resolutions and faster processing of data, *vi)* using better computer vision techniques for volume capture to extend the types subject motion that can be converted into kinematic motion.

A.2 Embedded Motion Capture from Sensor Networks

Sensor networks is a rapidly emerging area of research for distributed sensing in a variety of environments subject to vast amounts of uncertainty. Sensor networks typically consist

of a set of self-sufficient nodes containing wireless networking devices, a set of sensors, and a power source. Each nodes senses the world and relays sensing information across dynamic ad-hoc networks formed over wirelss communication. These networks appear a obvious match to natural motion capture given their aims for distributed, fault-tolerant, and dynamic acquistion of sensory information using small and subtle sensors. We discuss one approach to processing sensor information from such a network to provide 3D instrument locations similar to markers in an optical motion capture system. However, instead of localizing markers from external sensors, we relatively localize all sensors from their local measurements. By placing sensors on various locations on a subject's body, relative localization finds the global positions of the sensors from information local to the sensors.

Relative localization from pairwise proximities is an active topic of research in a variety of domains, such as multi-robot coordination [81], context aware computing [110], and sensor networks [138]. Relative localization is the placement of a set a points in a common coordinate frame such that a set of given pairwise distances are preserved. Common examples of relative localization include map building from inter-city distances and finding the configuration of a group of autonomous robots [58]. We are particularly interested in such domains where only local sensing may be available or appropriate.

Deterministic MDS techniques, like global SDR, are well suited to the problem of relative localization. However, their feasibility is based on the assumption that sensing can provide all-pairs proximity measurements indicative of distances that are not subject to significant amounts of noise or distortion. While local sensing in the real world may not hold to these assumptions, the application of global SDR is an attractively simple and efficient means for relative sensor localization, without the additional machinery of probabilistic MDS [89]. The appeal of such localization techniques is further enhanced when considering the current "sensor explosion" that could lead to more ubiquitous usage of current sensing devices and the development of new sensing modalities, such as the use the received signal strength readings provided by wireless RF Ethernet devices [81]. This sensing modality is particularly interesting when considering emerging sensor network technologies, such as SmartDust motes [55]. These devices may provide only limited local sensing capabilities, but will be small (approx. 1cm) and numerous. If they can provide signal strength measurements indicative of pairwise distances, relative localization can be performed for applications such as on-line capture of geometries for a moving subject, where each node provides a vertex of the subject's surface.

We experimented with localizing a set of Crossbow motes using Isomap on their pairwise signal strength measurements. A set of seven motes were placed in a static planar hexagon configuration with manually measured ground truth coordinates, as shown in Figure A.6. Signal strength measurements were collected over the course of 3783 seconds with no significant variance in the signal strength measurements. We used a single snapshot of these pairwise measurements. Ideally, received signal strength will decrease monotonically with distance. Assuming this is true, nondiagonal elements of a signal strength matrix can be made to form a distance matrix D based on a given maximum signal strength $D_{\max}$:

$$D_{ij} = \begin{cases} 0, & \text{if } i = j \\ D_{\max} - D_{ij}, & \text{if } i \neq j \end{cases} \tag{A.1}$$

We applied Isomap to D using nearest neighbors of $K = 3$ (shortest paths) and $K = N - 1$ (Euclidean distance), where $N =$ the number of motes. In our initial application of Isomap, the produced localizations are observably similar to the hexagon ground truth, but are not accurate 2D localizations. We attributed this inaccuracy to two types of artifacts due to the varied quality of radio communications on the motes. The first artifact is asymmetry in the distance matrix. This asymmetry is caused by the different transmission and reception capabilities between the motes. Symmetries between mote pairs indicate equivalent capabilities, while asymmetries indicate dissimilar capabilities. The second artifact is that the embedding preserves pairwise distances and is reflective of the ground truth, but the localization occurs in 3D. The 3D localization reflects the 2D ground truth only when viewed from a certain orientation. We attribute this artifact to signal strength measurements not reflective of relative distance, which produce matrix asymmetries. Additionally, the viewing orientation needed for the appropriate 2D localization is not readily extractable.

By addressing the problems with distance matrix asymmetries, a relative localization can be approximated for the appropriate dimensionality. Assuming inaccurate signal strength readings produce larger distances, mote pairs with asymmetric distances are arbitrated into symmetry by taking the minimum measured distance between the pair, forming a new matrix $D_s = \min(D, D^T)$. A mote pair is considered symmetric if $|D_{ij} - D_{ji}| < D_\epsilon$. A mote is *significantly symmetric* if it is symmetric with at least N_s other motes. We use motes that exhibit significant symmetry as landmarks in Isomap. By using these landmarks, we reduce the artifacts produced by bad radio communicators and focus on motes with more reliable distance measurements.

With these adjustments, Isomap produces a 2D localization that preserves the topology of the motes and visually approximates the hexagon ground truth. For localizing the hexagon (Figure A.6), we use $D_{\max} = 300$, $D_\epsilon = 15$, and $N_s = \lceil((N-1)/2)\rceil$ for a hexagon signal strength matrix with nonzero elements ranging between 69-196. The naive application of Isomap produced view-dependent localizations in 3D that were sensitive to varying K. From manual inspection, Isomap with symmetry and landmark adjustments produced 2D localizations for both $K = 3$ and $K = N - 1$ that approximate the underlying hexagon structure. The simplicity and relative accuracy of deterministic MDS via Isomap is attractive; however, probabilistic MDS offers several advantages for robustness to noisy and incomplete pairwise distance.

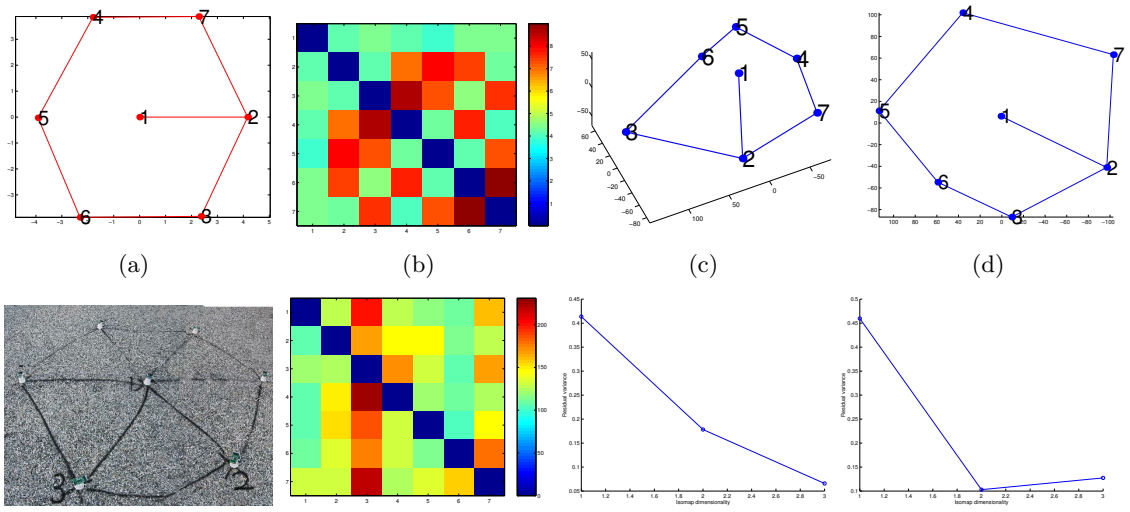


Figure A.6: Group-relative mote localization: (a,e) ground truth hexagon plot and picture of hexagon motes on a graveled roof. (b,f) pairwise ground truth distance and asymmetric signal strength distances (c,g) naive Isomap localization in 3D from best viewing orientation and residual variance. (d,h) adjusted Isomap localization in 2D and residual variance.

Appendix B

Applying Spatio-temporal Isomap to Robonaut Sensory Data

One drawback to our PDBV methodology is that heuristic segmentation must be performed before dimension reduction due to the computational limitations of Isomap. Undesirable artifacts may arise from segmenting motion in this manner and weaken the ability of ST-Isomap for exemplar grouping. Thus, the restriction to working with motion segments is not as desirable as working on the time-series of posture directly. Given this desire to avoid segmentation, we have begun to explore methods for applying sequentially continuous ST-Isomap directly to time-series of human postures.

As part of this effort, we describe joint work with Alan Peters at Vanderbilt University to apply sequentially continuous ST-Isomap to sensory data collected from the NASA Robonaut [1], a humanoid torso robot at the Johnson Space Center. For this joint work, Robonaut was teleoperated to grasp a horizonal wrench at nine different locations within its workspace. Robonaut continuously publishes its sensory and motor information to programs that record this information for further use. We applied sequentially continuous ST-Isomap on sensory data from five of the teleoperated grasps in an attempt to uncover the spatio-temporal structure of the grasp behavior. Data vectors recorded from Robonaut consist of 110 variables for both motor and sensory data. Motor data, including motor actuation forces and joint position and velocity, were zeroed out. The remaining 80 non-zeroed variables contain sensory data, consisting of tactile sensors on the fingers and force sensors on various positions of the robot. Each of these variables were normalized to a common range across all variables.

The embedding of this sensory data by ST-Isomap is shown in Figure B.1 with a comparison to embedding by PCA. The structure of the grasps can be vaguely interpreted in the PCA embedding. In contrast, the structure of the grasps are apparent in the ST-Isomap embedding as two loops. The smaller loop is indicative of reaching from and returning to the zero posture of the robot. The larger loop is indicative of grasp closure and release around the wrench. The points occurring during the grasp are within the smaller cluster.

The structure uncovered for the grasp provides a model that is a description of sensor values during a grasp. This model can also serve to describe sensory data of grasps not included for producing the embedding. To test this hypothesis, we selected and normalized data from a grasp not used for training. Given sensory data for the test grasp, training grasps, and embedded training grasps, interpolation can be used to map

the test grasp on the structure found in the embedding space. For this purpose, Shepards interpolation [127] was used to perform this mapping. The mapped test grasp from interpolation is shown in Figure B.1.

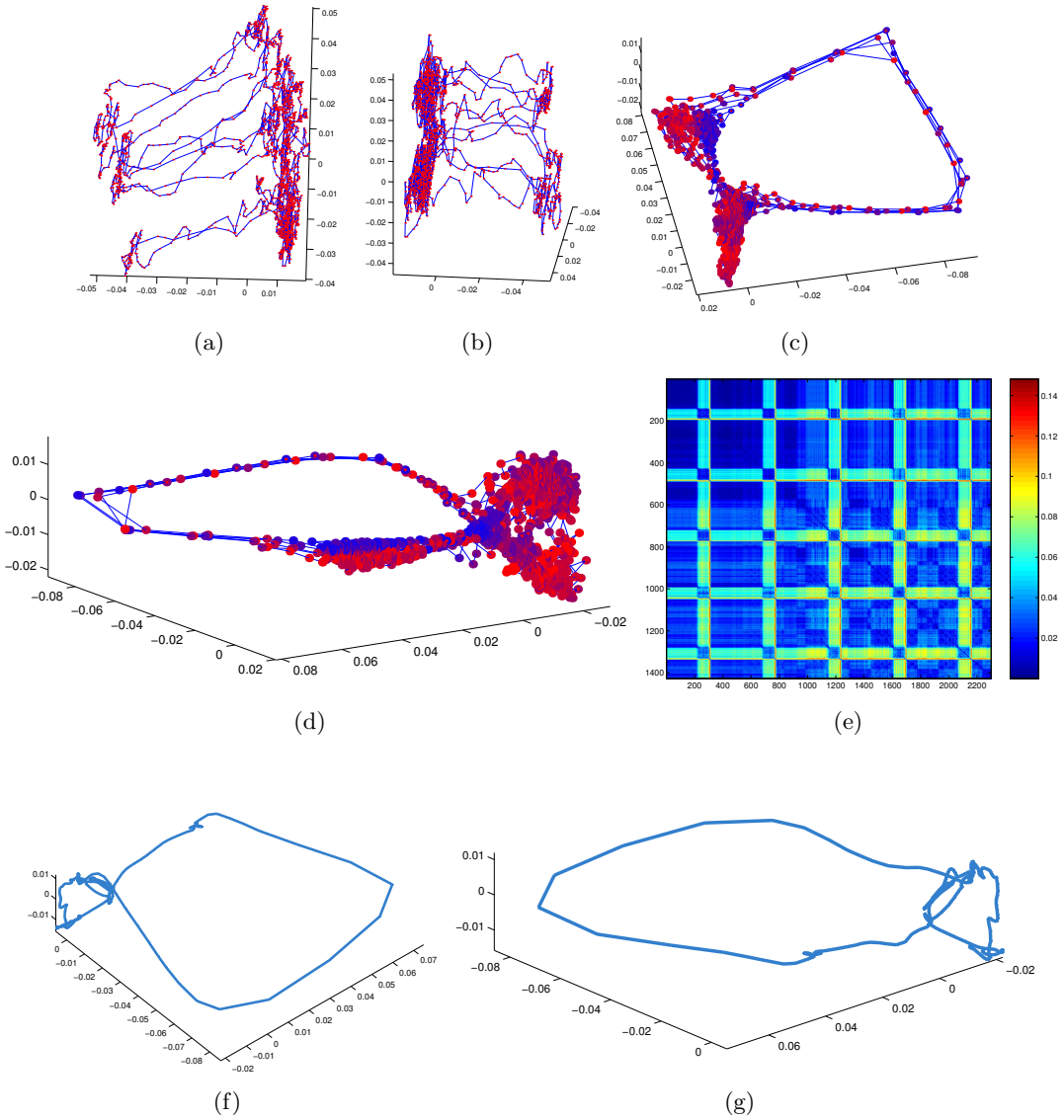


Figure B.1: *(a,b) Two views of the PCA embedding for the grasp data from Robonaut teleoperation. (c,d) Two views of the same data embedded by sequentially continuous ST-Isomap. (e) Distance matrix for the ST-Isomap embedding. (f,g) A test grasp mapped via Shepards interpolation onto the grasp structure in the ST-Isomap embedding.*

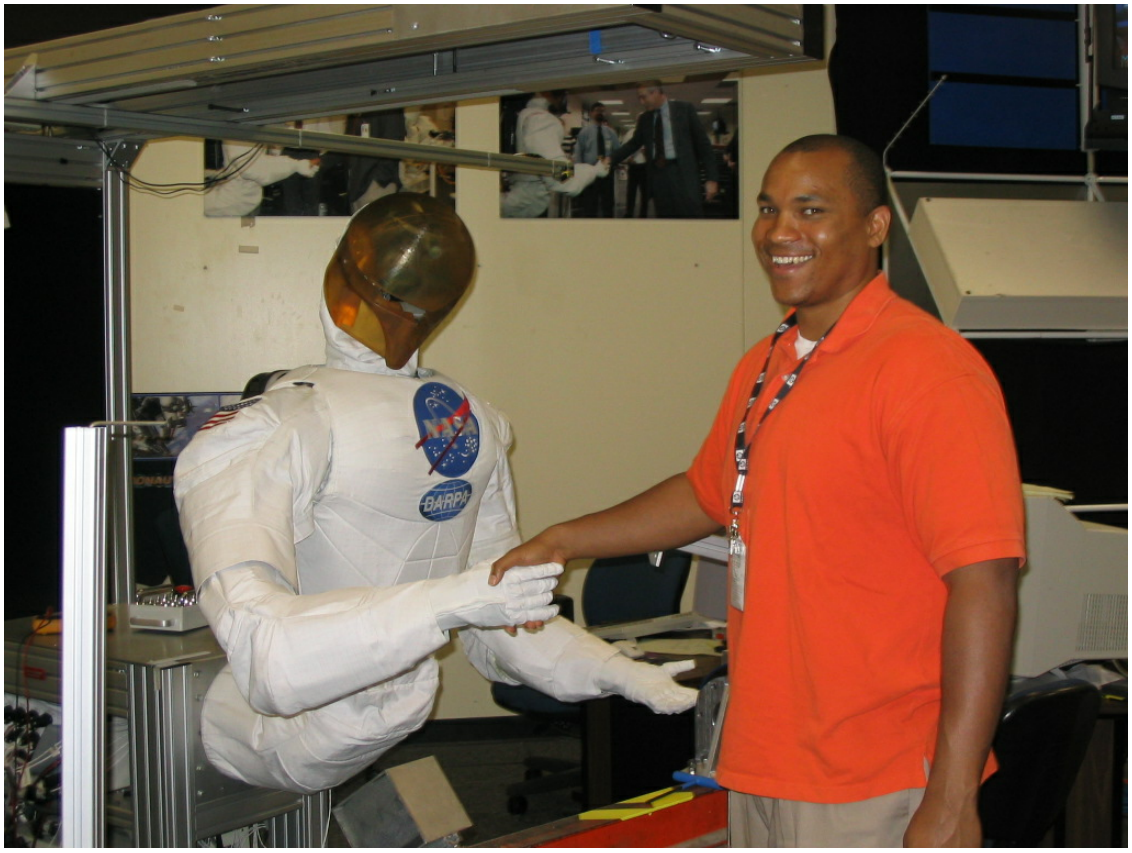


Figure B.2: *Gratuitious picture of the author with the NASA Robonaut. Thanks for reading my dissertation.*

Appendix C

Glossary

agglomerative clustering [68]: iteratively forming new clusters one at a time from the existing clusters.

adjacent temporal neighbors (ATN): data points that are sequentially adjacent within a time-series.

autoencoder [40]: a neural network trained using back-propagation of error, so that the network extracts the principal components of the input.

autonomous humanoid agent: an agent with the embodiment characteristics of a human and the ability to act without external supervision.

back-end motion processing: a process for generalizing motion examples of a module into a behavior.

behavior instance: see motion exemplar.

behavior vocabulary: a capability repertoire of exemplar-based behaviors, with each behavior defined by a generalization of motion exemplars.

blind source separation [60]: the decomposition of a signal or signals into underlying source signals with no prior knowledge of the sources.

capability abstraction: the use of capabilities for a humanoid agent that parsimoniously express the functionalities of the agent without requiring specific values for its degrees of freedom.

capability design: the specification of a modular set of behaviors for a humanoid agent.

capability grounding: the use of a repertoire of capabilities as a common vocabulary for structuring interactions between a humanoid agent and humans or other agents.

capability implementation: the realization of a capability design in the form of controllers for a humanoid agent.

capability repertoire: a set of modular behaviors designed and implemented for a humanoid agent.

capability scalability: the ease and practicality of modifying a capability repertoire.

clusterable proximity: the placement of data points such that distances between intra-cluster points is significantly smaller than inter-cluster points.

common temporal neighbors (CTN): a data point is a common temporal neighbor of given data point if it is within the local neighborhood and shares the same spatio-temporal signature of the given data point or is related through CTN transitivity.

concatenation motion synthesis: the generation of motion from a behavior vocabulary through concatenation of complete motion trajectories produced from a sequence of primitives.

controller encoding: a method for classifying motion into a string-like expression that captures the structure of an observed motion.

CTN component: a subset of data points in which all pairs are common temporal neighbors.

CTN transitivity: the ability to relate two data points as common temporal neighbors given that both share a third point as a common temporal neighbor.

distal correspondence: the correspondence of structurally similar data points that are potentially distal in the input space.

eager evaluation (or speculative evaluation): any evaluation strategy where evaluation of some or all function arguments is started before their value is required. More specific to this dissertation, an evaluation strategy that attempts to precompute the output of a model.

embedding: one instance of some mathematical object contained within another instance.

embedding space: a coordinate system produced as an embedding of an input space.

exemplar merging: the grouping of data points in the same feature group that are exemplars of multiple underlying behaviors.

exemplar space (or parameter space): a low dimensional space containing points corresponding to motion exemplars whose interpolation defines a primitive behavior.

factor analysis [49]: any of several techniques for deriving from a number of given variables a smaller number of different, more useful, variables.

feature group: see primitive feature group or meta-level feature group.

feedback property: motion synthesized from a derived behavior vocabulary will result in similar behaviors as its original input motion.

forward model motion synthesis: the generation of motion from a behavior vocabulary by continually updating desired kinematic postures based on the prediction of an active primitive.

forward model predictor: the use of a primitive behavior as a nonlinear dynamical system in joint angle space to provide predictions independent of a specific function.

front-end motion processing: a process for extracting descriptions of a set of modular behaviors from input motion data.

global spectral dimension reduction [34]: embedding through eigenvalue decomposition on a full matrix of scalar relationships between pairs of data points.

gradient [12]: the rate of increase or decrease of a variable magnitude, or the curve which represents it.

Hidden Markov Model (HMMs) [111]: a variant of a finite state machine with an unobservable current state having a set of states, Q, an output alphabet, O, transition probabilities, A, output probabilities, B, and initial state probabilities, P.

humanoid agent: an agent with the embodiment characteristics of a human.

imitation learning [120]: the acquisition of skills and/or tasks through observation.

independent and identically distributed (IID) [10]: a data set consisting of independent samples from the same underlying distribution.

Independent Components Analysis (ICA) [60]: blind source separation assuming the underlying source signals are statistically independent.

input space: coordinate system in which a set of input data resides.

interpolation [12]: calculation of the value of a function between already known values.

inverse kinematics (IK) [31]: the determination of a kinematic posture from end-effector positions.

Isomap [131]: a method for global spectral dimension reduction using shortest path distances to determine pairwise similarity.

joint angle space: the coordinate space formed by the agent's joint angles.

kernel PCA (KPCA) [125]: a method for global spectral dimension reduction using kernel functions centered on each data point to determine pairwise similarity.

kernel trick [125]: the implicit definition of a nonlinear mapping of data points through pairwise similarity scalars.

Kinematic Centroid Segmentation (KCS): motion segmentation based on treating each limb as a pendulum and segmenting motion based on its swings.

lazy evaluation: An evaluation strategy that evaluates an expression only when its value is needed and remembers this result for subsequent requests.

local neighborhood: a subset of points that are considered to be proximal to a given point.

local sensing: the limitation of an agent to sensing mechanisms provided within the embodiment of a humanoid agent.

local spectral dimension reduction (LSDR) [118]: embedding through eigenvalue decomposition on a sparse matrix of scalar relationships between proximal pairs of data points.

Locally Linear Embedding (LLE) [118]: a method for local spectral dimension reduction using weights from locally linear models centered at each data point to determine pairwise relationships.

K-nearest nontrivial neighbors (KNTN): the k best nontrivial neighbors in a local neighborhood.

marker features: perceptual features that drive the attention of a motion segmentation mechanism.

markerless motion capture: motion capture without instrumentation of the source.

memory model [101]: a model of explicitly remembered experiences from which predictions and generalization can be performed in real time.

merging artifacts: see exemplar merging.

meta-level behavior: a behavior that is sequential combination of primitive behaviors.

meta-level embedding spaces: coordinate systems produced by embeddings beyond the initial embedding.

meta-level feature group: clustering of motion exemplars in a meta-level embedding space representative of sequential combination of primitive behaviors.

mirror neurons [115]: neurons in the motor cortex that activate when performing or observing a certain class of movement.

motion capture: a process by which external devices can be used to capture movement data from various live sources in the world.

motion editing [48]: the modification of previously created or captured motion for new situations.

motion exemplar: a motion that is an example (or instance) of a particular behavior.

motion graph [79]: a graph of static kinematic posture nodes with directed edge transitions between postures.

motion mapping: the production of motion through mapping from a control space to joint angle space.

motion module: a modular description of a single capability in joint angle space.

motion textons [87]: a capability repertoire comprised of linear dynamical systems.

motor level: sensing and actuation mechanisms for achieving desired static configurations.

motor program [93]: a prestructured set of motor commands uninfluenced by feedback.

motor primitives [11]: a proposed biological model for the structure of the motor system as a biological or synthetic repertoire of primitives.

moveme [18]: a primitive building block for structuring motion, analogous to a phoneme for speech.

multidimensional scaling (MDS) [13]: a set of data analysis techniques that display the structure of data, from pairwise relationships, as a geometric picture.

nonlinear spherical shells (NSS) [27]: a method for extracting principal curves for a set of points through nonlinear dimension reduction and clustering on concentric spherical shell partitions.

nonparametric statistics [40]: the branch of statistics dealing with variables without making assumptions about the parameters of their distribution.

Performance-Derived Behavior Vocabularies (PDBV): a method for deriving a behavior vocabulary from kinematic time-series of human motion.

phase space: a 6R dimensional space of R variables described by 3R position and 3R momentum coordinates.

physical embodiment [21]: the realization of an agent in a body that is subject to the physical properties of the real world.

plant level: an agent's embodied interface to the world.

primitive: module that cannot be further subdivided and can be combined using defined operations with other primitives to create more intricate modules.

primitive behavior: a family of trajectories defined by a configuration of a primitive feature group in an exemplar space.

primitive feature group: a group of exemplars defining a primitive behavior

primitive feature group: clustering of motion exemplars with a common spatio-temporal signature in a primitive-level embedding space that is representative of a primitive behavior.

primitive forward model: a primitive behavior with the ability to predict future kinematic states from a current kinematic state.

primitive-level embedding space: the coordinate space produced by the first embedding of an input motion.

primitive support volume: the volume of coordinates in joint angle space for which a primitive forward model can be applied.

principal components analysis (PCA) [12]: a mathematical framework of determining that linear transformation of a sample of points in R-dimensional space which exhibits the properties of the sample most clearly along the coordinate axes.

principal curves [54]: self-consistent smooth curves which pass through the middle of a d-dimensional probability distribution or data cloud.

probabilistic roadmap [75]: a dynamic graph of configurations in the free space of an agent with transition edges between configuration.

proximal disambiguation: the separation of structurally different data points that are proximal in the input space.

proximity-equals-similarity assumption: the assumption that data points that are proximal in the input space are also structurally similar.

sample-atomic (or posture atomic): treating samples of a time series as indivisible units of data.

sampling space: a subspace of an exemplar space used to densely sample a primitive behavior.

segment-atomic: treating features extracted from a time series as indivisible units of data.

segment consistency: a property of segmentation that similar intervals of motion will yield similar segments of motion.

segmented common temporal neighbors (SCTN): determination of common temporal neighbors for segment-atomic data.

segment input space: an input space of segment-atomic motion trajectories of equal length.

segment sensibility: a property of segmentation that motion segments are sensible to a user.

self-organizing topographic map [77]: an unsupervised procedure for embedding using a defined, predetermined topology.

skill level: capabilities that drive motor level mechanisms according to motor program.

spatio-temporal correspondences: distal correspondences for underlying spatio-temporal structure.

spatio-temporal Isomap (ST-Isomap): a spatio-temporal extension of Isomap, with different methods for sample-atomic and segment-atomic data.

split behavior contexts: motion exemplars of a single behavior that are performed in the context of multiple sets of preceding and following behaviors.

spurious transitions: motion segments that are underrepresented transitions between activities performed in an input motion.

stroke: a single complete movement.

support vector clustering (SVC) [6]: a method for clustering data points mapped from an input space to a high dimensional feature space, where the smallest sphere that encloses the feature space data is mapped back into a set of contours in the input space.

task level: control policies for directing skill level capabilities to achieve the objectives of a higher-level task of the agent.

temporal windowing: accounting for temporal properties in time-series data by considering a window of data points about each data point.

time-series data [10]: a series of values of a variable at successive times.

tracking controller [148]: a controller for a humanoid agent that tracks an input motion while potentially being subject to the dynamics of the environment.

trajectory encoding: a method for classifying motion by concatenating the predictions of primitives that provide the best match over different intervals of motion.

trivial matches [26]: points in a local neighborhood that are superseded by a more representative neighbor.

unsupervised learning [40]: learning in which the system parameters are adapted using only the information of the input and are constrained by prespecified internal rules.

Verbs and Adverbs (V-A) [116]: a manually driven method for constructing exemplar-based behavior vocabularies.

video texturing [124]: the process of creating a graph of image nodes with directed edge transitions between images from a video sequence.

zero-posture: the default or resting kinematic posture of a humanoid agent.

z-function segmentation [44]: motion segmentation based on thresholding the sum of squares of the velocity of the degrees of freedom.