In the International Journal of Humanoid Robotics,1 (2), pp. 237-288, 2004

International Journal of Humanoid Robotics

PERFORMANCE-DERIVED BEHAVIOR VOCABULARIES: DATA-DRIVEN ACQUISITION OF SKILLS FROM MOTION

ODEST CHADWICKE JENKINS
MAJA J MATARIĆ

Robotics Research Laboratory,
Center for Robotics and Embedded Systems,
Computer Science Department,
University of Southern California,
941 West 37th Place, SAL 228,
Los Angeles, CA, 90089, USA
{*cjenkins|mataric*}*@usc.edu*

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

Control for and interaction with humanoid robots is often restrictive due to limitations of the robot platform and the high dimensionality of controlling systems with many degrees of freedom. We focus on the problem of providing a "skill-level interface" for a humanoid robot. Such an interface serves as a *i)* modular foundation for structuring task-oriented control, *ii)* parsimonious abstraction of motor-level control (e.g., PD-servo control), and *iii)* means for grounding interactions between humans and robots through common skill vocabularies. Our approach to constructing skill-level interfaces is two-fold. First, we propose a representation for a skill-level interface as a "behavior vocabulary", a repertoire of modular exemplar-based memory models expressing kinematic motion. A module in such a vocabulary encodes a flow field (or gradient field) in joint angle space that describes the "flow" of kinematic motion for a particular skill-level behavior, enabling prediction from a given kinematic configuration. Second, we propose a data-driven method for deriving behavior vocabularies from time-series data of human motion using spatio-temporal dimension reduction and clustering. Results from evaluating an implementation of our methodology are presented along with the application of derived behavior vocabularies as predictors towards on-line humanoid trajectory formation and off-line motion synthesis.

Keywords: motion primitives; motion clustering; spatio-temporal dimension reduction; motion segmentation; humanoid robotics.

1. Introduction and Motivation

Robotic humanoid agents are emerging in a variety of applications spanning several domains. A common theme throughout is the desire to realize humanoid robots as autonomous collaborators in human endeavors. However, current humanoids are closer to unintuitive automatons requiring an abundance of manual supervision. To

achieve the goal of collaboration, humanoid control and interaction methods must address several key questions, including:

- Autonomous control: How can a humanoid robot autonomously control itself?
- External interaction: How can our intentions be communicated to a humanoid?
- Task structure: How can tasks be represented in a manner amenable to autonomous humanoid control?

To illustrate the importance of these questions, consider the example of a human user instructing a robot to clean a room or throw away a piece of trash. First, the user must utilize some interaction modality to communicate to the robot the task to perform and to provide task-specific information, such as "throw away object X." Such collaborations occur at the *task-level*, eventually to be realized as commands for *motor-level* controllers (e.g., PD-servos). A motor-level controller translates commands indicating desired configurations for individual degrees of freedom into forces for the actuators of the robot. A room cleaning robot requires the specification of motor commands such that its task-level objective of a clean room can be achieved. The abstract nature of task directives and the specific nature of motor commands are significantly disparate, making robot control nontrivial and often restricted to specific tasks. In order to realize their role as human collaborators, however, humanoid robots must be able to produce motor-level commands that achieve task-level objectives autonomously for a variety of tasks. Additionally, such humanoids may be required to perform tasks that were not explicitly programmed.

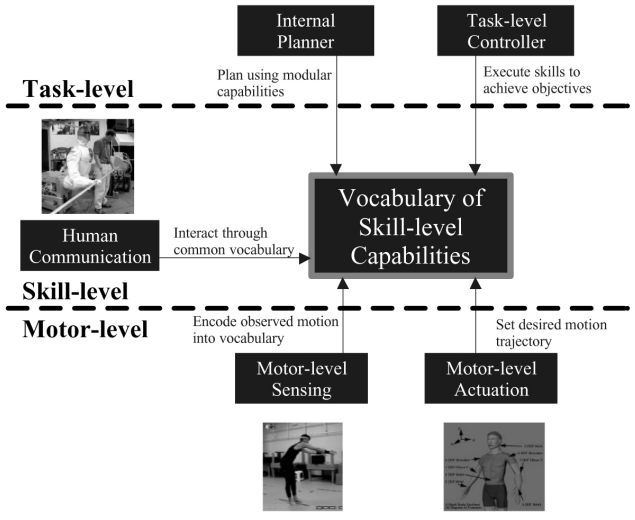


Fig. 1. Illustration of a skill-level interface as a vocabulary for bridging task and motor levels of humanoid robots such as the NASA Robonaut and the dynamically simulated Adonis.

Many of these issues can be addressed through planning-based approaches and/or modularizing the space of robot control. Planning provides flexible autonomy for a variety of tasks, but alone is often too slow for on-line performance. Faster approaches to robot control use modularity, employing a set of behaviors as a repertoire of the robot's capabilities. We refer to such modularity as a *skill-level interface* (Figure 1) that bridges high-level task descriptions and lower-level motor control. An interface of skills provides a parsimonious substrate for task-level operations (abstracting motor-level control and removing the need to specify joint angles directly) and structure for motor-level control (as motor programs). Behaviors in a skill-level interface are discretized to provide parsimonious spaces for planning, learning control policies[1], reacting[2], and distributed behavior-based coordination.[3] Such approaches to control effectively utilize modularity, but provide little insight into how to modularize effectively.

We address the modularization problem in a *data-driven* fashion by extracting structure from human motion data. In the same vein as *learning from demonstration*, we assume human motion is demonstrative of skills humans use to accomplish tasks. By extracting structure from human motion, we bypass explicit *model-based* presuppositions (e.g., domain knowledge).

In this paper, we describe our data-driven method, *Performance-Derived Behavior Vocabularies (PDBV)*, for automatically extracting behaviors underlying human motion data. Motion data acquired from human performance are used as input into PDBV to design and implement a repertoire of skills. A *skill* is either a *primitive* or *meta-level* behavior, defined as follows:

- Primitive behavior: a predictive model of kinematic motion capable of providing a gradient direction Δ_x from a point x in joint angle (or configuration) space.
- Meta-level behavior: a sequential model specifying transition probabilities among a set of primitive behaviors.

PDBV derives such behaviors from kinematic time-series data of human motion that are representative of multiple sequentially performed activities. Clusters in human motion data representative of behaviors are uncovered using *spatio-temporal Isomap (ST-Isomap)*[4]. Primitive behaviors are an extension of Verbs and Adverbs[5] behaviors; each cluster is a memory model[6] of motion trajectories generalized through interpolation. Our primitives are speculatively evaluated to form nonlinear dynamical systems in joint angle space. These models are predictive *perceptual-motor primitives*[7], useful for a variety of robot perception and control functions, including motion synthesis, classification, and imitation.

The remainder of this paper is organized as follows. This section describes issues and background work for representing and developing skill-level behaviors for humanoids. Section 2 describes our PDBV method. Results from evaluating PDBV and derived behavior vocabularies are presented in Section 3 and discussed in Section 4. The paper is concluded in Section 5.

1.1. *Issues in developing humanoid skills*

Several approaches to autonomous humanoid control and learning utilize skill-level capabilities.[2,3,8,1,9] Fully automated or *tabula raza* approaches to learning capabilities are appropriate for learning mappings such as in inverse kinematics[10] (mapping end-effector coordinates to joint angles) and inverse dynamics[11] (mapping actuation forces to control commands). For modularization into behaviors, however, a fully automated learning approach would need both a means to explore the space of possible modularizations and a metric to judge their quality. Given the complexity of humanoids, exhaustive exploration of this space is likely to be intractable and the metric for evaluation is difficult to characterize. Manual development of a skill-level repertoire is an effective approach, but is complex and domain-specific due to the need for *repertoire design* and *controller implementation*.

Any method (automatic or manual) is susceptible to producing a repertoire with errors. Although errors are inevitable, *scalable* approaches to modularization allow them to be corrected through rapid manual or automatic refinement. Scalability factors include the ease of adding new capabilities, removing unnecessary capabilities, modifying existing capabilities, and rebuilding controllers from altered designs.

PDBV is an automated method for initializing a scalable skill-level interface amenable to manual refinement. Our approach to skill acquisition avoids biasing a repertoire by leveraging behaviors inherent (but not immediately extractable) in human motion data. Behaviors derived by PDBV are defined by exemplar motions and, thus, are amenable to manual or automatic modification through changes to these exemplars. An exemplar-based behavior is scalable in that its functionality can be altered by modifying and automatically reevaluating constituent exemplars.

1.2. *Issues in representing motion capabilities*

We consider the primary form of humanoid expression to be rigid-body kinematic motion, although other modes are plausible (e.g., speech, facial gestures). Methods for representing motion capabilities fall into three non-mutually exclusive categories:

- motion mappings (e.g. inverse kinematics[12]);
- motion graphs (e.g., probabilistic roadmaps[13,14]);
- motion modules (e.g., behaviors[15,16,17]).

A *motion mapping* consists of two (or more) coordinate spaces, one of which serves as a "control space". The control space, a parsimonious and/or accessible coordinate system, is combined with a mechanism to map into a "command space", from which actuator commands can be generated. Similar to Eigenfaces[18] by Turk et al., Fod et al.[19] use *Principal Components Analysis (PCA)* to construct motion mappings between the space of joint angle trajectories and lower-dimensional principal coordinates. A significant problem is the difficulty and overspecification involved in producing a control space that is both parsimonious and accessible. Additionally, Sidenbladh et al.[20] present a similar PCA-based approach that creates a

binary tree index structure for vision-based kinematic tracking. While this method is used for both perception and control, they aim for enabling probabilistic search rather than modularization for autonomy.

A *motion graph* is constructed in a humanoid's configuration space to represent possible valid motion trajectories. Graph nodes are valid configurations connected by edges specifying a valid transition between configurations. Trajectories are specified as paths in the graph. Motion graphs can be constructed using a variety of techniques. *Tabula raza* approaches use exploration to produce variations on *probabilistic roadmaps.*[13] They require no *a priori* knowledge, but become intractable as the dimensionality of the configuration space increases. Alternatively, computer animation approaches[14] avoid exponential exploration by augmenting nodes from motion capture data with "splicing" transitions determined by a similarity metric. Work by Choi et al.[21] attempts to construct motion graphs in high dimensions using motion capture as a starting point for exploration. A single motion graph, however, is an inherently monolithic structure that: *i)* requires a potentially expensive search for trajectory formation, *ii)* is limited to control functions, and *iii)* is not suitable for non-deliberative control.

In contrast to monolithic motion graphs, motion capabilities for a humanoid robot can be represented as a set of modules. Analogous to deliberation and reaction[3], motion graphs are suited for top-down search and motion modules are suited for bottom-up coordination. Proposed approaches to modularization can be characterized by different properties such as the scope and form of each module, inter-module interaction, and repertoire design, construction, and functionality. We discuss these issues with respect to the following characteristics (specified in italics).

Input feature specificity refers to techniques that modularize motion specific to non-kinematic features extracted from human motion, such as points[22], contours[23], and color blobs.[24] Methods subject to *overmodularization*, such as movemes[24], are specific to an individual rigid body, resulting in a large number of modules that requiring explicit coordination across all bodies to represent motion. Overmodularization becomes problematic when the difficulty in coordinating modules becomes significantly greater than a set of modules individually. In contrast, *coarse modularization* is due to a small number of modules with overly broad parameter spaces, such as discrete point-to-point motion[25,26], rhythmic oscillation[27,17], or their combination.[28]

Model specificity refers to methods that rely on specific hypotheses of human motion, such as EMOTE proposed by Chi et al.[29] Methods for modularization subject to *heavy user supervision* require significant manual interaction in behavior design and/or implementation. These include the "control basis" approach[30], hand coded behaviors[31], behaviors tailored from motion[32], vocabularies for user annotation[33], and Verbs and Adverbs behaviors.[5]

Automated methods focused on generalizing a single behavior are restricted to *single class automation*, such as associative memory [34], clustering[35,36], and Hidden Markov Models.[37] Ijspeert et al.[17,26] proposed a single class method that encodes

a primitive module as a nonlinear attractor for a demonstrated trajectory. Collectively, a set of such primitives can express a more general behavior, but the means for grouping primitives into such collections is not provided. Recent work by Okada et al.[38] proposes a technique for automatically constructing interpolation spaces and realizing dynamical systems given a set of motions expressing the same skill. This method utilizes an exemplar-based skill representation similar to Verbs and Adverbs. As a potential extension, our PDBV method could be used to cluster motion into behaviors that are individually generalized by Okada's more specific mechanism.

In contrast to batch processing in PDBV, methods for incremental modularization[39,40,41] exist. Such methods suffer from *limited correspondence ability* in that two motions that do not currently appear similar may be similar given future motions. Additionally, methods that modularize through clustering, such as Fod et al.[19], are subject to limited correspondence when proximity in the input space is not indicative of underlying structural similarity. Methods such as *motion textures*[16] yield modules with *limited interpretation*. Motion textures represents modules as linear dynamical systems, providing an elegant framework for modularization, but are difficult to use without complex controllers.

Our PDBV methodology is based on establishing *spatio-temporal correspondences* in motion data. Similar in spirit, Ilg et al.[42] presented *Spatio-Temporal Morphable Models (STMMs)* as a means to automatically extract primitives as prototypical motion trajectories. STMMs use dynamic time warping to establish spatio-temporal correspondences between input motion data and prototypes. Similar to work by Ijspeert et al.[17], the STMM approach works well when training data are sparse and represent a primitive behavior with respect to a single motion trajectory. Consequently, combinations of such primitives are likely to represent style variations on a single behavior rather than a complete repertoire.

Recent work by Kovar and Gleicher[43] utilizes a similar approach to PDBV for corresponding motions using spatio-temporal relationships. Kovar and Gleicher present a new spatio-temporal distance metric utilized to construct manifold-like *match webs* across a set of motion data. Match webs, however, is a generalization of their motion graphs work aimed towards query-based search for producing highly controlled animations. In contrast, motion modules extracted by PDBV are expressed as dynamical systems for use with various autonomous perception and control functions.

The PDBV methodology we present is a modularization of motion into exemplar-based behaviors. PDBV establishes pairwise spatio-temporal correspondences between motion examples in the input data. These correspondences allow a module to internally span across stylistic variations of an underlying behavior. Variations on each module are indexable through a motion mapping between the space of motion trajectories and a lower-dimensional "exemplar space". Though not explicitly discussed in this paper, PDBV provides a generalization of motion data amenable to motion graph construction.

2. Performance-Derived Behavior Vocabularies

Performance-Derived Behavior Vocabularies (PDBV) (Figure 2) is a methodology for automatically deriving modular skills from kinematic motion data. PDBV takes as input a single continuous time-series of kinematic configurations. As output, PDBV produces a *behavior vocabulary* of kinematic motion modules composing a repertoire of *primitive* and *meta-level* behaviors. Primitive behaviors are exemplar-based memory models that generalize clusters of motion extracted from an input motion into dynamical systems in joint angle space. The essence of a primitive behavior B is the ability to provide prediction in the form of:

$$\Delta_x = B(x) \tag{1}$$

where x is a point in joint angle space and Δ_x is vector direction from x. Meta-level behaviors are sequential models of transitioning between a set of primitives. More specifically, a meta-level behavior M is an $W \times W$ matrix specifiying the transition probabilities between a set of W primitives. Each element $M_{i,j}$ specifies the probability of a primitive B_j producing motion once another primitive B_i reaches completion. Assuming a demonstrated input motion is representative of all motion encountered by a system, M can be stated as a first-order Markov model[44]:

$$M_{i,j} = P(b_k = B_i | b_{k-1} = B_j) \tag{2}$$

where B is a set of behaviors and b is a sequential list specifying the sequence of performed behaviors with respect to a motion.

PDBV extracts primitive and meta-level behaviors from free-space motion data on the assumption that these data are structured by an underlying spatio-temporal process. This assumption allows us to equate the problem of behavior extraction to finding spatio-temporal structure in data. For behavior extraction, we use segmented ST-Isomap[4] to uncover spatio-temporal structure with the following four-step PDBV procedure:

(1) Motion preprocessing (Section 2.1): producing time-normalized motion segments from the input motion data;
(2) Exemplar grouping (Section 2.2): clustering motion segments into behaviors based on common spatio-temporal signatures;
(3) Behavior generalization (Section 2.3): forming nonlinear dynamical systems for behaviors from grouped exemplars;
(4) Meta-level behavior grouping (Section 2.4): grouping exemplars of lower-level behavior based on higher-level spatio-temporal signatures.

Each step is described in detail.

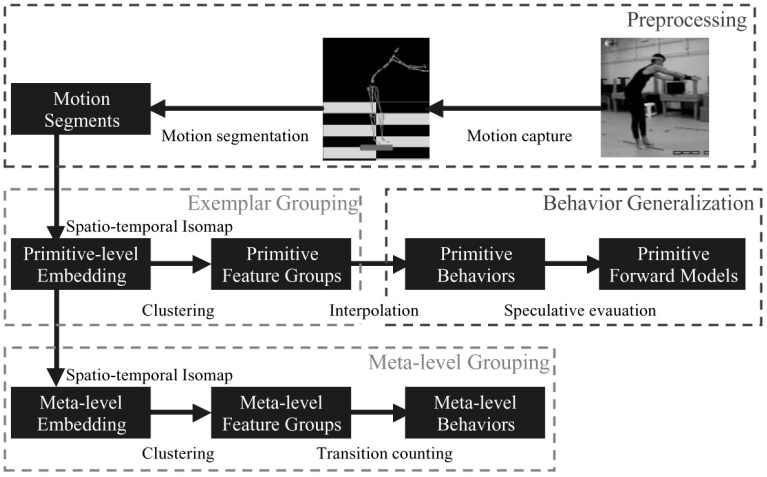


Fig. 2. Performance-Derived Behavior Vocabularies consist of four main steps: motion preprocessing, exemplar grouping, behavior generalization, and meta-level behavior grouping. Preprocessing produces a data set of motion segments from real-world human performance. Exemplar grouping uses ST-Isomap to cluster motion variations of the same underlying behavior. Exemplars of a behavior are generalized through interpolation and eager evaluation. Common sequences of primitives in an input motion are found as meta-level behaviors using additional ST-Isomap iterations.

2.1. *Motion preprocessing*

The first step in PDBV is the preprocessing of an input motion $m(t) = [\theta_1 ... \theta_{N^{\mathtt{DOF}}}] \in \Theta$ to produce a data set of motion segments with constant dimensionality. Preprocessing consists of segmentation followed by time normalization.

The result from any segmentation method (manual or automatic) is a sequential set of segments of various lengths l_i. The dimensionality of the i^{th} segment is $d_i = l_i \times N^{\mathtt{DOF}}$, where l_i is the length of segment i and $N^{\mathtt{DOF}}$ is the number of regarded DOF. Dimension reduction techniques considered in this work require data points of equal dimensionality. Thus, each segment of S is normalized to a constant length l by constructing a cubic spline and interpolating for l uniformly spaced samples.

The result from preprocessing is a sequential set of motion segments S. Each segment $S_i \in \Theta_T$ represents a $d_s = l_s \times N^{\mathtt{DOF}}$ dimensional point in the space of joint angle trajectories Θ_T.

2.1.1. *Kinematic Centroid Segmentation*

Kinematic Centroid Segmentation (KCS), illustrated in Figure 3, is our heuristic method for automatically segmenting free-space motion (i.e., without external object or environment interactions). Based on assumptions similar to those of Cutting and Profitt[45], KCS performs "swing detection" by treating a kinematic substructure of a performer as a pendulum, placing segment boundaries at the beginning and end

of pendulum swings. A kinematic substructure is a set of DOF that are always in coordination for a common purpose, such as an arm, that can work in coordination with or independently of other substructures. The rationale for swing segmentation is that the revolute joints and joint limits of a human restrict the reachable space of a kinematic substructure. Due to these constraints, the distance a substructure can achieve from its initial position is bounded. Thus, a substructure moving away from its initial position must eventually reach a local extreme and move back toward its initial position. KCS is an alternative to common "stop detection" techniques, such as the z-function of Fod et al.[19], that search for velocity zero crossings across the DOFs and are best suited for segmenting discrete point-to-point motion.

KCS iteratively segments one kinematic substructure at a time in a greedy fashion. Segmentation of a single substructure abstracts the kinematic configuration as a *centroid feature*, the average location of features along the arm, with respect to a *base feature*. KCS is described in the following procedure:

(1) Initialize the list of segment boundaries S as the input motion m;
   (a) $S = \{1, |m|\}$;
(2) Iterate over each kinematic substructure k:
   (a) $i = 1$;
   (b) Iterate over each segment S_j in the current set of segments S:
      i. while $(s_i \neq S_{j+1})$
         A. Set current segment s_i to the first frame of S_j;
         B. Compute distance between centroid at s_i and the centroid at every subsequent frame t upto S_{j+1};

$$m_d(t) = \texttt{dist}_c(m_{\texttt{centroid}_k}(t) - m_{\texttt{base}_k}(t), m_{\texttt{centroid}_k}(s_i) - m_{\texttt{base}_k}(s_i)) \tag{3}$$

         C. Find first local maximum s_{i+1} in centroid distance function;

$$s_{i+1} = \texttt{min}(t_b) : \quad m_d(t_b) - m_d(s_i) \geq \tau; m_d(t_b) \geq m_d(t_s), \forall t_b - \epsilon_{\texttt{KCS}} \leq t_s \leq t_b + \epsilon_{\texttt{KCS}} \tag{4}$$

         D. $i = i + 1$;
   (c) Update the segment list $S \leftarrow s$; Clear s
(3) Remove last segment boundary from S;

The above instantiation of KCS assumes the following: S is initialized as a single segment from the beginning to the end of input motion m, $m_d(t)$ is the value of the centroid distance function at time t, $\texttt{dist}_c$ is the metric for computing distances between centroids (Euclidean as default), $m_{\texttt{centroid}}(t)$ and $m_{\texttt{base}}(t) = m_{\texttt{shoulder}}(t)$ are the Cartesian positions of the centroid and base for the kinematic substructure at a given time t, a local maximum in $m_d(t)$ is greater than some threshold τ and maximal across a temporal window of length $(2{\cdot}\epsilon_{\texttt{KCS}})+1$ (to avoid issues with noise).

$m_{\texttt{centroid}}(t)$ is not resticted to a specific definition, but we used the following in our implementation for arm motion:

$$m_{\texttt{centroid}}(t) = \frac{m_{\texttt{wrist}}(t) + m_{\texttt{elbow}}(t) + m_{\texttt{shoulder}}(t)}{3} - m_{\texttt{base}}(t) \tag{5}$$

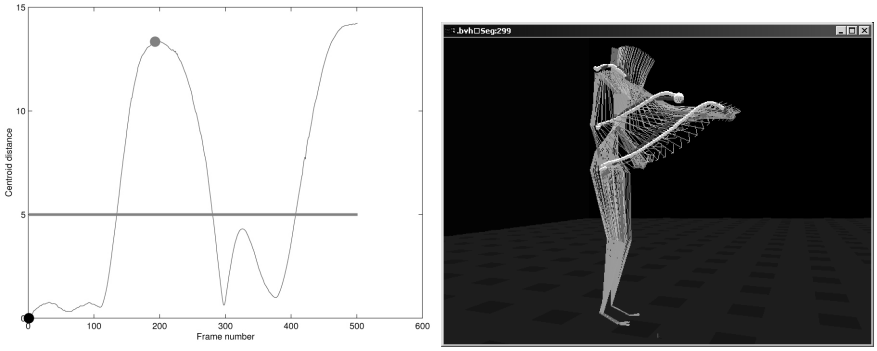


Fig. 3. Illustration of Kinematic Centroid Segmentation for a two-arm reaching motion. (Left) A plot of the kinematic centroid distance over 500 frames from the beginning of the reach. A horizonal line is placed to mark the "significant motion" threshold τ. The segment boundary is noted by the dot in the middle of the plot. (Right) The dotted trajectories are shown for the base (shoulder), centroid, and hand of the right arm. The segment boundary is highlighted by a larger sphere in the centroid trajectory.

2.2. *Exemplar grouping into primitives*

Exemplar grouping is the process of clustering motion segments such that each group contains variations on a common theme. For behavior extraction, each group contains motion variations with a common spatio-temporal structure, representative of an underlying behavior. For this purpose, we use ST-Isomap to transform a data set of motion segments into clusterable feature groups. We consider a motion segment as a point in a $N^{\texttt{DOF}} \times l$ dimensional *segment input space*, where $N^{\texttt{DOF}}$ is the number of regarded performer 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. In the remainder of this section, we briefly describe *Sequentially Segmented ST-Isomap* and its use for clustering motion data.

2.2.1. *Sequentially Segmented Spatio-temporal Isomap*

ST-Isomap[4] is our spatio-temporal extension of Isomap[46] nonlinear dimension reduction. ST-Isomap is an unsupervised method for uncovering structure in segmented or continuous data with *i)* potentially high input dimensionality, *ii)* spatial

nonlinearity, and *iii)* temporal ordering. For PDBV, we focus on *Sequentially segmented ST-Isomap* for *segmented* data, where data points are intervals of the input and temporally adjacent data points are not necessarily spatially proximal.

Isomap is a method for nonlinear dimension reduction through constructing a full shortest-path distance matrix that is embedded through eigendecomposition. We paraphrase the general framework of Isomap from Tenenbaum et al.[46] as follows:

(1) Create a sparse $|S|\times|S|$ matrix D^n nonzero elements representing edges between local neighbors:

$$D^0_{i,j} = \begin{cases} \texttt{dist}(S_i, S_j) & \text{if } S_j \in \texttt{nbhd}(S_i) \\ \infty & \text{otherwise} \end{cases} \tag{6}$$

(2) Complete D into a full shortest-path distance matrix such that:

$$D^g_{i,j} = \begin{cases} D^0_{i,j} & g = 0 \\ \texttt{min}(D^{g-1}_{i,j}, D^{g-1}_{i,k} + D^{g-1}_{k,j}) & g \geq 1 \end{cases} \tag{7}$$

(3) Perform multidimensional scaling (MDS) to embed into d_e-dimensional coordinates that preserve the pairwise distance relationships in D such that the following error is minimized:

$$E = |D^g - D_e|_{L^2} \tag{8}$$

where `dist()` is the distance between two segments (typically Euclidean distance), `nbhd()` are the local neighbors of given segment (typically K-nearest neighbors or an ϵ-radius), D_e is the matrix of Euclidean distances in the embedding, and $\|A\|$ is the L^2 matrix norm of a given matrix A. It is crucial to note that the essence of Isomap is that shortest-path distances are assumed to be reflective of the distances with respect to the underlying topology of the data. The MDS step serves to translate shortest-path distances into coordinates. Thus, we can think of Isomap as preserving distances indicative of the underlying topology while removing configuration-specific nonlinearity.

ST-Isomap retains the general framework of spatial Isomap for constructing a full distance matrix that is embedded through eigendecomposition. However, ST-Isomap takes into consideration temporal dependencies between sequentially adjacent points for:

- *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.

Figure 4 illustrates our approach to the problems of proximal disambiguation and distal correspondence using three arm waving motions. The two low waving motions are relatively proximal in joint angle space but are structurally different due to moving in opposite directions. The low and high motion waving the same

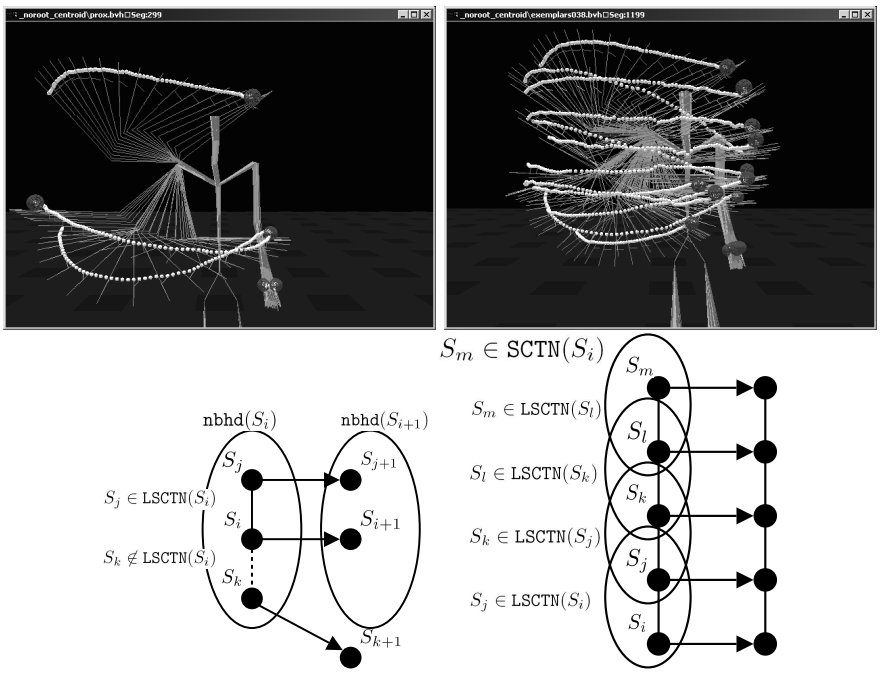


Fig. 4. An illustration of proximal disambiguation and distal correspondence. (Top left) Three waving motions with hand trajectories shown as a dotted trail. The beginning of each trajectory is marked with a large sphere. (Top right) A set of exemplars connected the low and high waving motions through similarly structured motions. (Bottom left) A diagram of disambiguating two proximal points of S_i using LSCTN(). S_i has a spatio-temporal correspondence with S_j and not S_k. (Bottom right) A diagram of corresponding two distal points, S_i and S_m, through transitivity using the definition of SCTN().

direction are structurally corresponding but are distal in joint angle space. By establishing correspondences between structurally similar waving motions, the high and low waving can be distally corresponded through transitivity, as illustrated in Figure 4.

Sequentially segmented ST-Isomap (heretofore referred to as simply ST-Isomap) consists of the following five steps, with our enhancements of Isomap in bold:

(1) **segment preprocessing: partition input data into intervals, replacing input data with $|S|$ higher dimensional segments**;
(2) compute sparse distance matrix D^l from local neighborhoods nbhd(S_i) about each segment S_i as K-nearest neighbors using Euclidean distance;
(3) **identify local segmented common temporal neighbors LSCTN(S_i) of each segment S_i**;

$$S_j \in \texttt{nbhd}(S_i) \text{ and } S_{j+1} \in \texttt{nbhd}(S_{i+1}) \Leftrightarrow S_j \in \texttt{LSCTN}(S_i) \tag{9}$$

(4) **reduce distances in $D^l_{S_i,S_j}$ between points with common and adjacent**

temporal relationships

$$D^0_{S_i,S_j} = \begin{cases} D^l_{S_i,S_j}/(c_{\texttt{CTN}}c_{\texttt{ATN}}) & \text{if } S_j \in \texttt{LSCTN}(S_i) \text{ and } j = i+1 \\ D^l_{S_i,S_j}/c_{\texttt{CTN}} & \text{if } S_j \in \texttt{LSCTN}(S_i) \\ D^l_{S_i,S_j}/c_{\texttt{ATN}} & \text{if } j = i+1 \\ D^l_{S_i,S_j} & \text{otherwise} \end{cases} \tag{10}$$

(5) complete D^0 into full all-pairs shortest-path distance matrix $D = D^g$ (Dijkstra's algorithm), such that $g \geq |S|$;
(6) embed D into d_e-dimensional embedding space through MDS.

Described later in this section, $c_{\texttt{CTN}}$ and $c_{\texttt{ATN}}$ are user defined constants for scaling distances between data pairs with temporal relationships.

The initial step in ST-Isomap is the abstraction of the *pose-atomic* input motion data into *segment-atomic* motion intervals through segmentation. 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. Our framework for ST-Isomap is not specific to any particular segmentation mechanism, boundary event definition, or time normalization. In order to produce a structurally appropriate embedding, however, segments produced from 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 subsequence of the input data). Mutually exclusive segments (i.e., not overlapping in time) are recommended, but not required.

After preprocessing the input data, *hard* spatio-temporal correspondences are found between motion segments as *segmented common temporal neighbors (SCTN)*. Local segmented common temporal neighbors (LSCTN) are found as local neighbors that temporally transition to the same neighborhood. The idea behind SCTN is that two segments that share a common spatio-temporal structuring A will be followed by segments from a different spatio-temporal structure B. By corresponding LSCTN locally, global components of SCTN with distal correspondences will be extracted via shortest-path computation and embedded into *clusterable proximity*. Clusterable proximity results from our consideration of common temporal correspondences as locally symmetric and globally transitive:

$$S_j \in \texttt{LSCTN}(S_i) \Leftrightarrow S_j \in \texttt{LSCTN}(S_i) \tag{11}$$

$$S_j \in \texttt{LSCTN}(S_k) \ \texttt{and} \ S_k \in \texttt{LSCTN}(S_i) \Rightarrow S_j \in \texttt{SCTN}(S_i) \tag{12}$$

As the value of $c_{\texttt{CTN}}$ increases, the distance between two points with a SCTN relationship decreases. Thus, the shortest-path distance between any SCTN-connected pair $D^g_{S_i,S_j}$ will be significantly smaller than the distance between any non-SCTN-connected pair $D^g_{S_i,S_k}$, for a significantly large value of $c_{\texttt{CTN}}$. A set of points with

all-pairs SCTN paths forms a *CTN component*. CTN components could be extracted without embedding via MDS, but we use embeddings as a means for visualization. Clusterable proximity is more formally stated as:

$$S_j \in \texttt{SCTN}(S_i) \ \texttt{and} \ S_k \notin \texttt{SCTN}(S_i) \Rightarrow D^g_{S_i,S_j} \ll D^g_{S_i,S_k} \tag{13}$$

Members of a cluster for A are found implicitly using the members of a temporally adjacent cluster B. Consequently, a clustered embedding provides a structure like $A \rightarrow B \rightarrow C$, where A, B, and C are clusters. This structure is indicative of spatial variations within each cluster and temporal transitions between clusters. In Section 2.4, we also reduce distances between *adjacent temporal neighbors (ATN)* to extract higher-level modules from previously extracted exemplar groups, such as a single cluster ABC from multiple $A \rightarrow B \rightarrow C$ clusters.

2.2.2. *Exemplar grouping using ST-Isomap and bounding box clustering*

An embedding produced by ST-Isomap places motion segments with a common spatio-temporal structure into clusterable proximity. Such embeddings for primitive behaviors are produced by setting $c_{\texttt{CTN}}$ to some significantly large value (100 for our validation results), adjusted through manual tuning, and $c_{\texttt{ATN}} = 1$.

Given an embedding of significant dimensionality d_e, such clusters K of motion exemplar groups will be separable through a variety of methods[47].

To avoid issues in estimating cluster cardinality $|K|$, we perform clustering using the "sweep-and-prune" technique[48] from the collision detection literature. Sweep-and-prune partitions data based on axis-aligned bounding boxes, using a partition threshold distance $\epsilon_{\texttt{SAP}}$ rather than an *a priori* estimate for $|K|$. The partitioning procedure begins with a single cluster and iteratively divides this cluster based on projections of the data onto an individual axis of the embedding. For a given projection, an existing cluster is sorted along the axis and partitioned when the distance between two adjacent points is greater than $\epsilon_{\texttt{SAP}}$.

Once clustering is performed, the points of a single cluster are identified as a *primitive feature group*. Because of their spatio-temporal correspondence, motion segments in each primitive feature 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 input space of motion segments. Together, however, the exemplars of a group are indicative of an infinite span of variations that are structurally common to a single behavior.

2.3. *Generalizing exemplars into predictive flow fields*

Primitive behaviors in PDBV are similar to and inspired by *verb behaviors* proposed by Rose et al. for Verbs and Adverbs vocabularies.[5] Similar to a primitive behavior, a verb behaviors is defined by a set of exemplar motion trajectories K_i and generalized through interpolation. For both primitive and verb behaviors, each exemplar

group K_i has a one-to-one correspondence with points in a lower dimensional *exemplar space* R_i, or *adverb space*. For our implementation, the coordinates for points in R_i are taken directly from a given cluster K_i in the primitive ST-Isomap embedding. However, the location of points in R_i could be specified through a variety of other methods (e.g. manually[5], model fitting[38], dimension reduction through PCA, Isomap, etc.). This correspondence serves to define mappings G_i for primitive B_i between its exemplar space R_i and the space of joint angle trajectories Θ_T:

$$G_i : R_i \mapsto \Theta_T \tag{14}$$

The motion mapping G_i expresses an infinite span of motion variations, or *motion support volume*, for a primitive behavior B_i. Although we use Shepard's interpolation[49] to approximate G_i and G_i^{-1}, the specific mapping mechanism between input and exemplar spaces is methodologically independent.

Unlike PDBV-derived primitive behaviors, verb behaviors enjoy a manually defined R_i, and thus human-intuitive parameterization. For a verb behavior, exemplars are manually positioned in an exemplar space such that the parameterization of the behavior is intuitive. Consequently, a human user can reliably produce desired motions through *lazy evaluation* (i.e., evaluation when needed) of a motion mapping G_i, avoiding complicated or exhaustive search procedures.

By virtue of their automatic derivation, primitive behaviors are not guaranteed to have human-intuitive parameterizations. To uncover useful parameterizations, we employ *speculative (or eager) evaluation* (i.e., evaluation in advance) on G_i to approximate an explicit mapping between a primitive's exemplar space and resulting motion trajectories in joint space. Similar to Isomap[46], we make a *motion manifold assumption* that the span of motion trajectories for a primitive behavior actually lies on a low-dimensional manifold in joint angle space Θ. We term this manifold as the *primitive motion manifold*. Speculative evaluation of G_i, without this assumption, is likely to be infeasible due to an exponential increase in necessary samples.

For speculative evaluation of G_i, a set of $N_{\texttt{SAM}}$ points V are randomly placed within an exemplar space R_i. They are evaluated for the set of trajectories $\hat{V}$ that will represent the primitive motion manifold p:

$$\hat{V}_j = G_i(V_j) \tag{15}$$

An example of a primitive motion manifold produced through speculative evaluation is shown in Figure 5.

To enable prediction, we treat primitive motion manifolds as flow/gradient fields. We frame prediction using Jordan's formalism[11] for a dynamical system as a next-state and an output equation:

$$x[n+1] = f(x[n], u[n]) \tag{16}$$

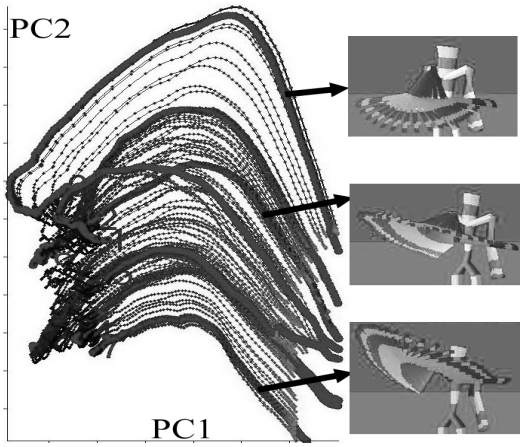


Fig. 5. A primitive flow field in joint space for a horizontal arm waving behavior, shown with respect to its first two principal components PC1 and PC2. The flow field moves forward along the trajectories, roughly from left to right. Exemplars from the input motion are shown in bold. Motion for selected exemplars of this primitive are shown on the right.

$$y[n] = g(x[n]) \tag{17}$$

where $x[n]$ and $x[n+1]$ are the current and desired kinematic states at time n, $u[n]$ is control input, f is a function predicting the next state from the current state, and g is a function mapping states to observations $y[n]$. For simplicity, we consider observations in joint angle space and, thus, the output equation is an identity function $x[n] = g(x[n])$. However, observations are not required to be in joint angle space, as in the case of using Cartesian endeffector observations with our primitives[50].

Fitting with Jordan's description, a primitive motion manifold B_k provides next-state prediction with respect to (a potentially nonlinear) manifold within some volume of support on $x[n]$:

$$x[n+1] = f_k(x[n], u[n]) = x[n] + u[n] \cdot B_k(x[n]) \tag{18}$$

The volume of support for B_k is defined by its component trajectories in joint angle space $\hat{V}_k$. More specifically, B_k is comprised of a set of kinematic configurations p_i with sequential relationships $p_{i+1} \xrightarrow{\texttt{follows}} p_i \Leftrightarrow p_i \in \hat{V}_k$ and $p_{i+1} \in \hat{V}_k$. In most cases $p_{i+1} \xrightarrow{\texttt{follows}} p_i$ have a sequential relationship; although, for some samples $p_{i+1} \xrightarrow{\texttt{does not follow}} p_i \Leftrightarrow p_i \in \hat{V}_k$ and $p_{i+1} \in \hat{V}_j$ such that $k \neq j$. In other words, p_i is at the end of a trajectory and p_{i+1} is the start of a new trajectory. The sequential relationships are reflective of the dynamics of a primitive demonstrated in an input or generalized motion trajectory:

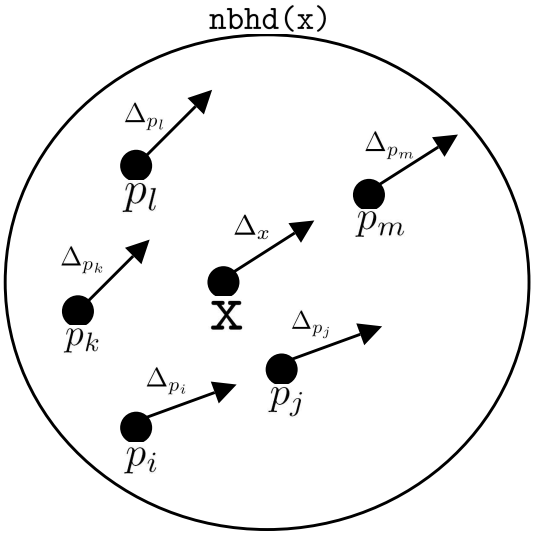


Fig. 6. A illustration of a gradient Δ_x for a point x on a primitive flow field using neighboring samples of p and their gradients.

$$p_{i+1} \xrightarrow{\texttt{follows}} p_i \Rightarrow p_{i+1} = u_i \cdot B_k(p_i) \tag{19}$$

where u_i is the velocity magnitude. Because time normalization is performed after segmentation, such magnitudes have been skewed. Thus, we leave $u[n]$ (i.e., step length) as a user parameter. Beyond the scope of this paper, u_i from the input motion m can be reintroduced as a separate field or through variable length gradients at each p_i. Through their sequential relationships, we phrase the normal-length gradient Δ_{p_i} (or direction of displacement) for each sample p_i as:

$$B_k(p_i) = \Delta_{p_i} = \frac{p_{i+1} - p_i}{\|p_{i+1} - p_i\|} \tag{20}$$

Illustrated in Figure 6, we phrase the normalized gradients Δ_x for a point x on the flow field through the weighted average of gradients from neighboring trajectory samples `nbhd(x)`:

$$B_k(x) = \Delta_x = \frac{\sum_{y \in \texttt{nbhd}(x)} w_y \Delta_y}{\| \sum_{y \in \texttt{nbhd}(x)} w_y \Delta_y \|} \tag{21}$$

where w_y is the weight for neighboring point y with respect to x. In our implementation, we specified w_y using the reciprocal distance or a Gaussian radial basis function:

$$w_y = \exp\left(-\frac{(y-x)^2}{\sigma^2}\right) \tag{22}$$

For prediction, $f_k(x[n], u[n])$ is used to dynamically step from $x[n]$ by length $u[n]$. A primitive B_k can be traversed from $x[1]$ by stepping to $x[2]$ followed by $x[3]$ and so on up to some $x[l]$. However, a primitive does not necessarily rely on $x[n+1]$ being achieved immediately after $x[n]$. In the context of robot control, a predicted $x[n+1]$ may not be achieved from $x[n]$. The robot may instead achieve $\hat{x}[n+1]$, which we assume will roughly follow $\Delta_{x[n]}$. A primitive will use the robot's current configuration to resume prediction as $f_k(\hat{x}[n+1], u[n])$. Similar to Ijspeert et al.[17], this incremental prediction ability is beneficial for added robustness for autonomous control. Similar to rigid body dynamics[51], however, our method of incremental prediction is prone to instability for large $u[n]$ step lengths.

2.4. *Meta-level exemplar grouping*

Given primitive feature groups described in Section 2.2, we now describe using additional iterations of ST-Isomap to group exemplars of meta-level behaviors. To restate, meta-level behaviors are sequential models of transitions (like Markov chains) between lower-level behaviors, indicative of higher-level patterns in motion. Primitive feature groups found in the embedding space have a spatial and temporal relationship that forms an $A \rightarrow B \rightarrow C$ structure of clusters. This structure can 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 (e.g., $A \rightarrow B \rightarrow C$) into a single higher-level behavior (e.g., ABC). Similar to primitive behaviors, meta-level behaviors are derived through extracting *meta-level feature groups* using ST-Isomap.

Meta-level feature groups are found as clusters within a *q^{th}-level embedding space*, where $q > 1$. A meta-level behavior $M_{i,j}^{q,k}$ for the k^{th} feature group describes the probability of transitioning from one lower-level behavior $M^{q-1,i}$ to another $M^{q-1,j}$:

$$M_{i,j}^{q,k} = P(S_{l+1} \in K^{q-1,j} \text{ and } S_{l+1} \in K^{q,k} | S_l \in K^{q-1,i} \text{ and } S_l \in K^{q,k}) \tag{23}$$

where $K^{q,i}$ is the k^{th} feature group for the q^{th}-level embedding. Meta-level behaviors are complimented by the transition probability matrix $\hat{M}_{i,j}^{q}$ between all feature groups of an embedding:

$$\hat{M}_{i,j}^{q} = P(S_{l+1} \in K^{q,j} | S_l \in K^{q,i}) \tag{24}$$

While no meta-level behaviors occur at the primitive level $q = 1$, $\hat{M}_1$ does contain transition probabilities between all primitives with respect to an input motion.

An $(q+1)^{\text{th}}$-level embedding is produced by applying ST-Isomap to the data in the q^{th}-level embedding, such that the q^{th}-level features are preserved. 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 transitions between feature groups.

Lower-level feature groups are preserved in higher-level embeddings by constructing local neighborhoods to include only points from a single established feature group. Such neighborhoods are constructed using an ϵ-radius based on the bounding radius of the corresponding feature group at the previous level. Thus, the neighborhood function for the $(q+1)^{\text{th}}$-level embedding is:

$$\texttt{nbhd}_{q+1}(S_i) = \{S_j : \|S_i - S_j\| < \tau_q\} \tag{25}$$

where τ_q is the bounding radius of the largest cluster in the q^{th}-level embedding.

Assuming group integrity and an appropriately large value for $c_{\texttt{ATN}}$, lower-level feature groups are collapsed into clusterably proximal higher-level feature groups by reducing distances between ATNs of feature groups with strong transitions. The strength of transition from one feature group to another is given by the number of ATNs expressing the transition.

3. Validation Results

We evaluated our methodology for deriving behavior vocabularies through an implementation of PDBV in MATLAB. This system was applied to three multi-activity input motions. 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 motions.

3.1. *Input motions and preprocessing*

Each input motion[a] $m_i(t)$, individually referred to as Input Motion i, was collected from a human subject performing a series of scripted activities centered around upper-body movement. Input Motions 1 and 2, containing 22,549 and 9,145 frames, respectively, were scripted to contain multiple activities, including punching, dancing, and arm waving. Input Motion 3, containing 9,394 frames, was scripted to contain a single two-arm reaching activity between a *zero-posture* and various Cartesian locations in the subject's reachable space. The input motions consisted of 42 kinematic DOF. The original performer motion contained 69 DOF, but less relevant DOF were removed to avoid DOF weighting issues and minimize kinematic mismatches with a humanoid robot. The streams are available for viewing at *http://robotics.usc.edu/∼cjenkins/motionmodules/*.

[a]Motion data used in this paper were obtained using a Vicon optical motion capture system and were graciously provided by Jessica Hodgins and her motion capture group at Carnegie Mellon University.

The input motions were segmented using three different methods: manual, z-function[19], and KCS. Manual segmentation was performed to place segment boundaries at the end of perceived *strokes*. Because the definition of a stroke can vary[52], manual segmentation was constrained such that linear interpolation between segment boundaries would produce a coarse approximation of the input motion. Each segment was normalized to 100 frames and ST-Isomap was applied using $c_{\texttt{CTN}} = 100$. As shown in Tables 1 and 2, the number of motion segments and derived behaviors produced from using KCS exhibits greater resemblance to manual segmentation than z-function segmentation. These tables illustrate that the z-function is more prone to missing *true negative* segments in dynamic motion (as with Input Motions 1 and 2) and detecting *false positive* segments in discrete point-to-point motion.

Table 1. Motion segmentation statistics.

	num. segments	mean	st. dev.	min.	max.
input motion 1 (manual)	250	90.19	113.72	15	970
input motion 1 (KCS)	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 (KCS)	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 (KCS)	64	142.45	40.44	7	261
input motion 3 (z-function)	84	104.52	48.02	5	259

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

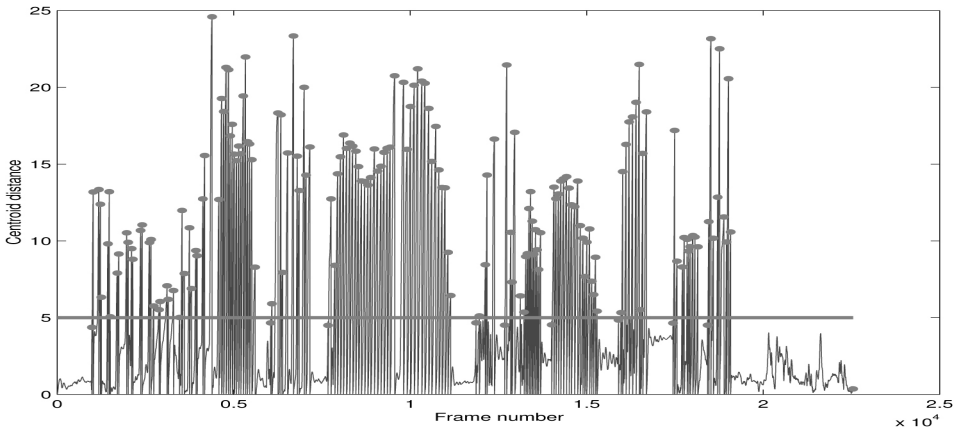
Table 2. Numbers of derived primitives per input motion.

	manual	KCS	z-function
Input Motion 1	84	78	32
Input Motion 2	62	37	20
Input Motion 3	7	5	10

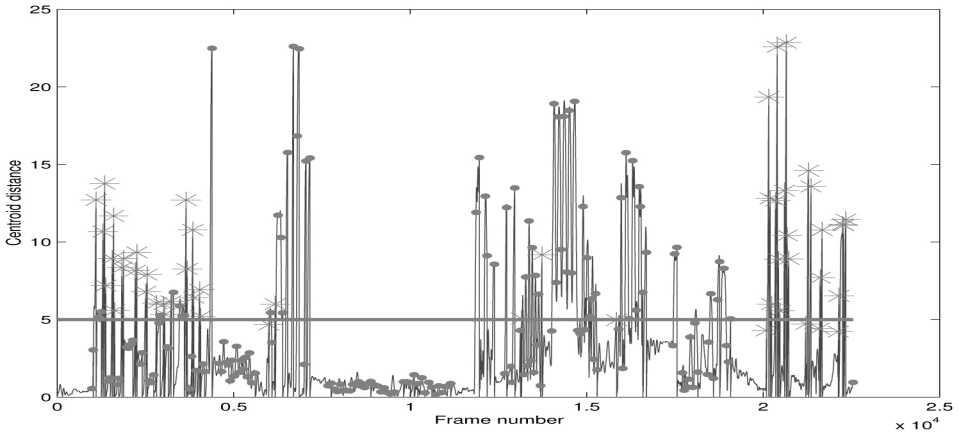
Note: Number of primitives derived for each input motion and each segmentation procedure.

3.2. *Humanoid robot control and motion feedback*

We evaluated the feasibility of PDBV-derived behavior vocabularies for on-line and off-line robot control through the use of a basic task-level arbitration mechanism.



(a)



(b)

Fig. 7. Plots of KCS applied to Input Motion 1. (Top) The kinematic centroid distance over the frames of the input motion for the first kinematic substructure, the right arm. Segment boundaries are shown as dots in the plot. The centroid distance is zeroed after each segment boundary. (Bottom) The kinematic centroid distance for the second kinematic substructure, the left arm. Segment boundaries remaining from the right arm are shown as dots in the plot. New segments introduced by the left arm are shown as stars.

For control, we assume a behavior vocabulary is accompanied by a task-level controller and a motor-level interface. The motor-level interface provides reliable DOF proprioception and translation of desired configurations to actuator forces. The task-level controller indexes into behaviors of a vocabulary to output motor-level configuration commands. Several task-level control mechanisms for purposeful control exist, such as behavior-based methods proposed by Nicolescu and Matarić.[8] We used a simple *random walk* arbitration mechanism, for simplicity, using a Markov

model described in Section 2. Additionally, motion produced from control is fed back as input into PDBV to test for derivation consistency. Regardless of the task-level control mechanism, PDBV-derived behavior vocabularies can be used to synthesize motion for off-line or on-line trajectory formation.

For our random walk arbitrator, a single primitive is initially activated to drive the control process. It continually updates the current motor-level configuration desireds in time through prediction with a fixed gradient magnitude. When the activated primitive approaches completion, the arbitrator switches activation to another primitive on transition probabilities among a set of primitives or a meta-level behavior.

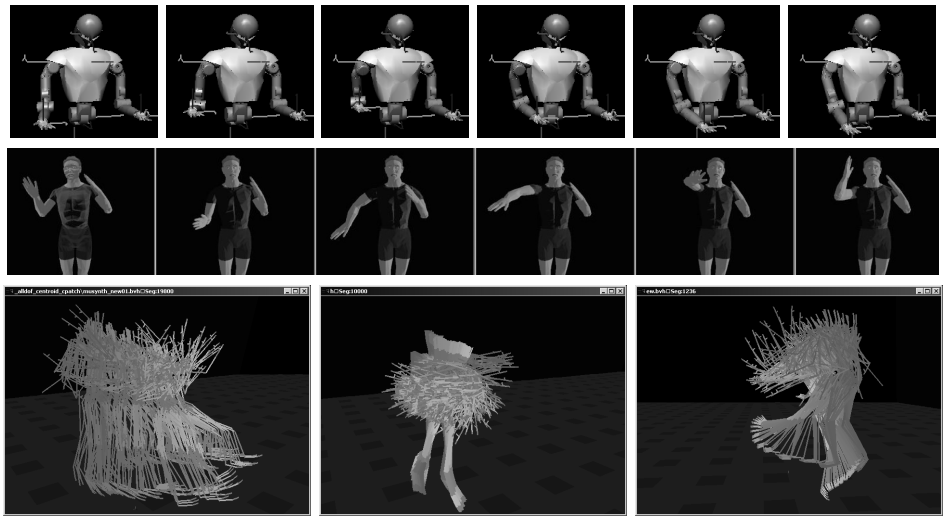


Fig. 8. Snapshots from (Top) on-line right-arm control of a "cabbage patch" dance by a simulated Robonaut and (Middle) off-line trajectory synthesized for a punching behavior. (Bottom) Postures from off-line trajectory formation using derived vocabularies for (right to left) the "cabbage patch" dance, horizontal arm waving and combined jab and uppercut punching.

Using the random walk arbitrator, we were able to autonomously control RoboSIM, a simulation of the NASA Robonaut,[53] on-line. The same client/server interface is used to control both Robonaut and RoboSIM. We used various behavior vocabularies to control the right arm of RoboSIM, including vocabularies for vertical arm waving, jab punching, and the "cabbage patch" (illustrated in Figure 8), over several executions of the activity without user intervention. Similar to Pollard et al.[54], adapting motion produced by human-based behaviors required significant manual tuning to deal with the kinematic mismatch between the robot and human performer.

Additionally, we used the same control mechanism to synthesize motion off-line in an incremental fashion for several derived vocabularies. The synthesis procedure

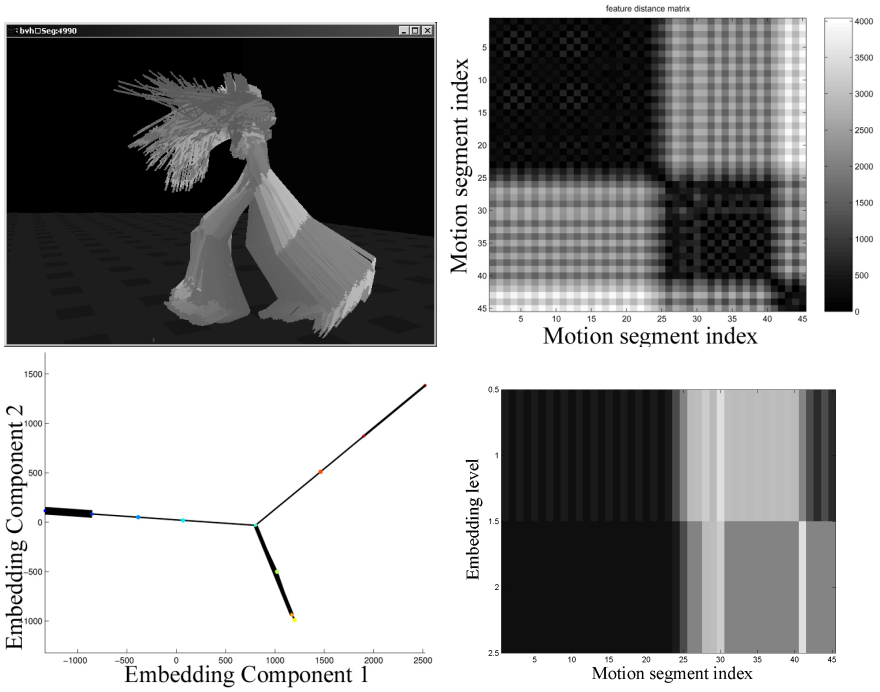


Fig. 9. (Top left) Postures from off-line trajectory formation for jab punching. Results from feeding back jab synthesized motion into PDBV as the resulting distance matrix (top right), clustered embedding (bottom left), and cluster assignments to motion segments (bottom right). Darker elements in the distance matrix indicate segments with greater similarity. The embedding is shown as a branching 1-manifold. Thicker lines indicate higher a transition probability between clusters on the manifold. The intensity of an elements in the cluster assignment matrix indicates the cluster associated with each motion segment.

came to completion at the specified number of frames for horizontal arm waving (3000 frames), jabbing (5000 frames), and "cabbage patch" (20000 frames) vocabularies, shown in Figure 8. The synthesis process for these vocabularies was manually stopped, but could continue for a theoretically infinite duration. However, this motion synthesis procedure is susceptible to premature termination for behavior vocabularies with limited support volumes. Selected motions synthesized off-line were actuated by a dynamical humanoid simulation, Adonis,[55] 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. Snapshots of its performance of a synthesized punching motion are shown in Figure 8.

Conceptually, synthesized motion should be structurally similar to the original input motion. Thus, the *feedback property* should hold such that a behavior vocabulary derived from synthesized motion should be similar to the originally de-

rived vocabulary. To evaluate this property, we fed motion synthesized from the "jab" vocabulary (chosen for its simplicity) back into PDBV. Figure 8 shows a checkerboard distance matrix and alternating feature groups with one meta-level behavior produced from feedback. These visuals illustrate structural similarity to the original input motion through the alternation between two primitives. However, the alternation pattern applies only on submatrix blocks along the diagonal. This partitioning is due to the divergence in the motion synthesis mechanism towards a "downward punch". Even with this divergence, the motion alternates between "jab" and "return" primitives.

3.3. *Feature extraction and exemplar grouping*

PDBV was applied to each of the three input motions, producing separable clusters of motions. For brevity, we focus on the results for Input Motion 1 at various stages in the PDBV procedure, shown in Figure 10. Data pairs (of motion segments) within the same CTN component are visualized in the pre-embedding distance matrix as dark elements. 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 are visualized as checkerboard-like patterns indicating structural transitions between CTN components. For example, the activity with alternating primitives is visualized as a block checkerboard. A checkerboard pattern results from the waving activity, performed by alternating between "waving outward" and "waving inward" primitives. Although the waving activity was performed over two non-adjacent intervals (segments 60-75 and 105-120), these motions were clustered together, as indicated by the off-diagonal blocks in the distance matrix. Similar to the primitive-level, second-level feature groups formed separable clusters in the second-level embedding, visualized in Figure 10.

3.4. *Primitive speculative evaulation*

For PDBV, we assume that a family of motion variations is a low dimensional manifold in the joint space of the capture subject. We are able to visualize these primitive motion manifolds by applying PCA and viewing the projection onto its first 3 principal components (PCs). Figures 12-19 illustrate meta-level behaviors and primitive motion manifolds derived from Input Motions 1 and 3. These manifolds required no more than 3 PCs (or ECs meaning Embedding Components) for accurate viewing and typically formed bordered 2-manifolds in joint angle space. In this case, ECs are equivalent to PCs.

We observed from these visualizations that speculative evaluation through sampling provides a representative realization of a primitive's span of variation, even when using basic Shepard's interpolation. Through manual inspection, we determined that PDBV appropriately extracted and generalized primitive behaviors for activities from Input Motion 1 including: arm waving (Figures 12 and 13), punching (Figure 14), and dancing the "Cabbage Patch" (Figure 15). Additionally, the single

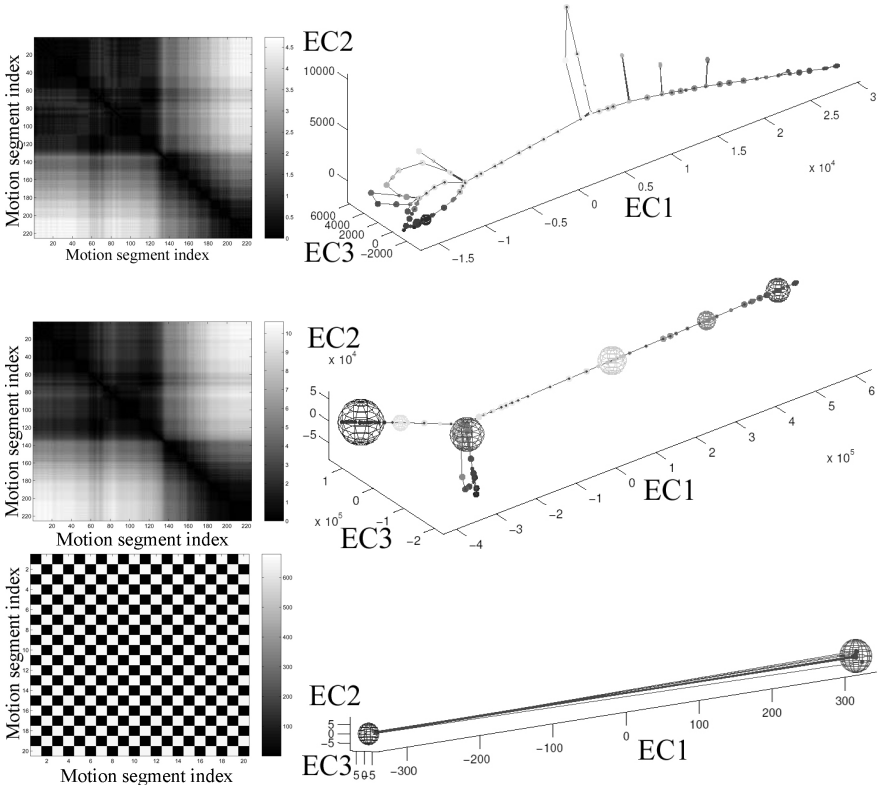


Fig. 10. Results of primitive-level (top), second-level (middle), and isolated waving activity (bottom) exemplar grouping for Input Motion 1. The left image in each row are embeddings produced as a 1-manifold (the dark line) connecting the sequence of motion segments and clusters (bounding spheres). On the right, distance matrices (dark elements indicate greater similarity) produced by ST-Isomap are shown for each embedding.

activity in Input Motion 3, reaching (Figure 16), was appropriately extracted and generalized. However, our implementation of PDBV demonstrated some artifacts in the extraction process, as shown for hand circling with the left arm (Figure 17) and dancing "the Twist" (Figure 18) activities. The "swing up" primitive for hand circling inappropriately merged exemplars for circling with the left and right arms. While the primitive generalizes appropriately and is not a practical problem, the modularization is not consistent. The "twist left" primitive inappropriately merged an exemplar performing "the Swim". This merge is an artifact from only one performance of dancing "the Swim" in Input Motion 1. Additionally, a single performance of the "Itsy Bitsy Spider" occurred during Input Motion 1, yielding a single primitive (Figure 19). While not practical as a primitive behavior, the "spider" primitive represents an appropriate modularization that should also happen for a "The Swim" primitive.

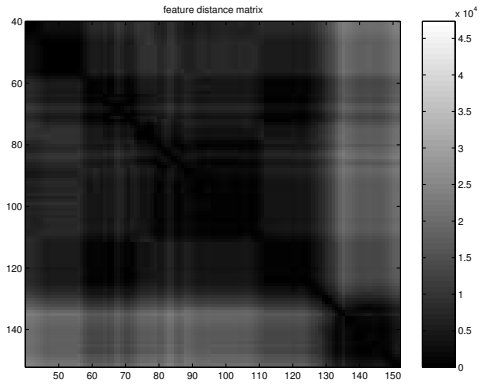


Fig. 11. A magnified view of the primitive-level distance matrix for Input Motion 1, which illustrates the off-diagonal blocks that correspond non-adjacent performances of the same primitive behavior.

4. Discussion

The quality of structures derived by PDBV is dependent on the preprocessing mechanism and appropriateness of parameters used for ST-Isomap. Preprocessing of an input motion prepares data for dimension reduction and, thus, has a direct impact on ST-Isomap. The user-defined parameters of ST-Isomap define the manner in which spatio-temporal structure and underlying behaviors are uncovered from motion segments. We discuss these two critical issues in the remainder of this section.

4.1. *Consistency and sensibility in motion preprocessing*

The aim in motion preprocessing 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. Sensibility means that each segment is *i)* a significant expression of 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.[16] In contrast, "long segments" may contain behaviors that are specific to a sequence of sub-behaviors, providing a more modular description. However, such segments that are structurally similar may be difficult to place into clusterable proximity due to greater, potentially more distal spatial signatures and sparseness between common temporal neighbors. Sensible segments are in the middle ground between short and long segments, large enough to describe meaningful motion but small enough to provide modularization.

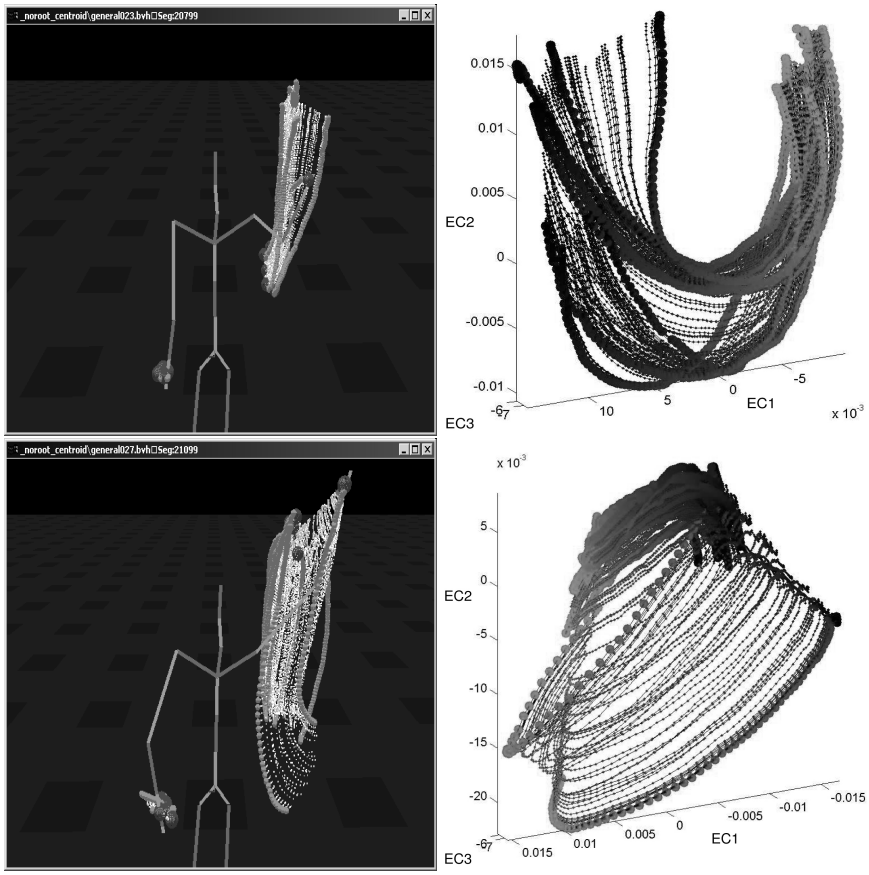


Fig. 12. Illustration of the "vertical waving" meta-level behavior from Input Motion 1. Vertical waving consists of two primitives for "wave up" (top) and "wave down" (bottom).

4.2. *Parameter tuning for ST-Isomap and exemplar grouping*

Parameter tuning is a significant issue for ST-Isomap and clustering motion segments with a common spatio-temporal structuring. ST-Isomap parameters are particularly important because they define the local neighborhood of each point, and consequently define distal spatio-temporal correspondences. 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 CTN 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 mixture of exemplars from different behaviors in a given neighborhood. The potential for such

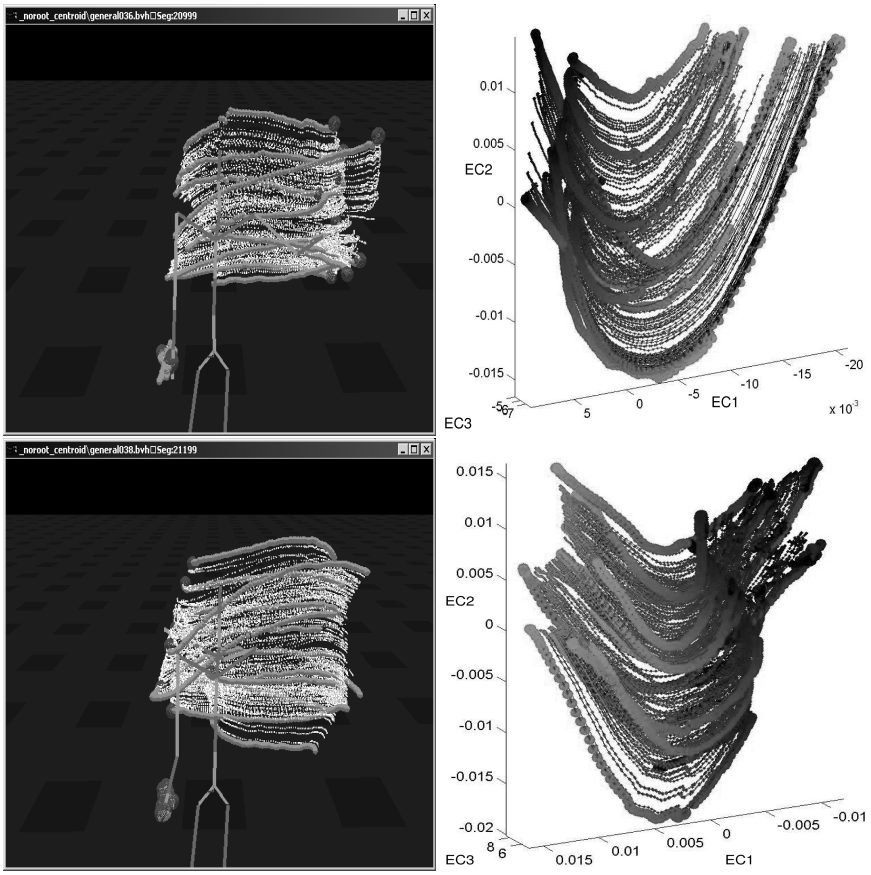


Fig. 13. Illustration of the "horizontal waving" meta-level behavior from Input Motion 1. Horizontal waving consists of two primitives for "wave inward" (top) and "wave outward" (bottom). The motion manifold for each primitive is shown with respect to a kinematic figure in Cartesian space (left) and its first 3 ECs in joint angle space. Each manifold is illustrated with exemplars in bold and interpolations in small dots. The Cartesian figure marks the beginning of exemplars with a large sphere.

nonrepresentative local spatial neighborhoods is at the root of merging artifacts, but also plays a role in *splitting artifacts*. These occur when exemplars of a single underlying behavior are placed into multiple CTN components. Splitting artifacts are due to underrepresentation and sparseness of exemplars for a particular underlying behavior. 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.

Behavior splitting occurs in two forms: split instances and split behavior contexts. *Split instances* of an underlying behavior are uncovered as multiple CTN components. Simply, there exists two instances not connected to each other through the

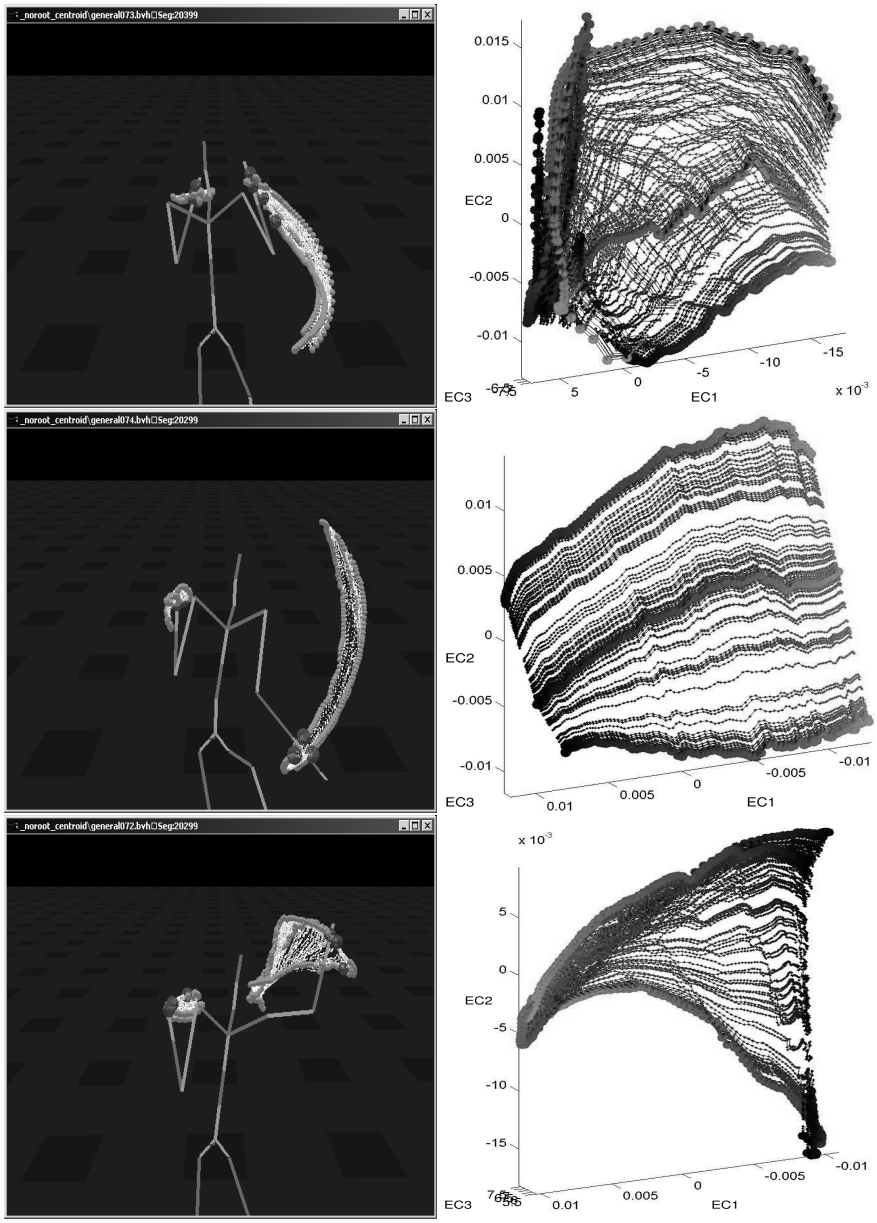


Fig. 14. Illustration of the "punching" meta-level behavior from Input Motion 1. Punching consists of three primitives for "retract from ready posture" (top), "punch" (middle), and "return to ready posture" (bottom).

neighborhoods of other instances in the CTN component, typically due to exemplar sparseness. The problem of *split behavior contexts* occurs when instances of the same

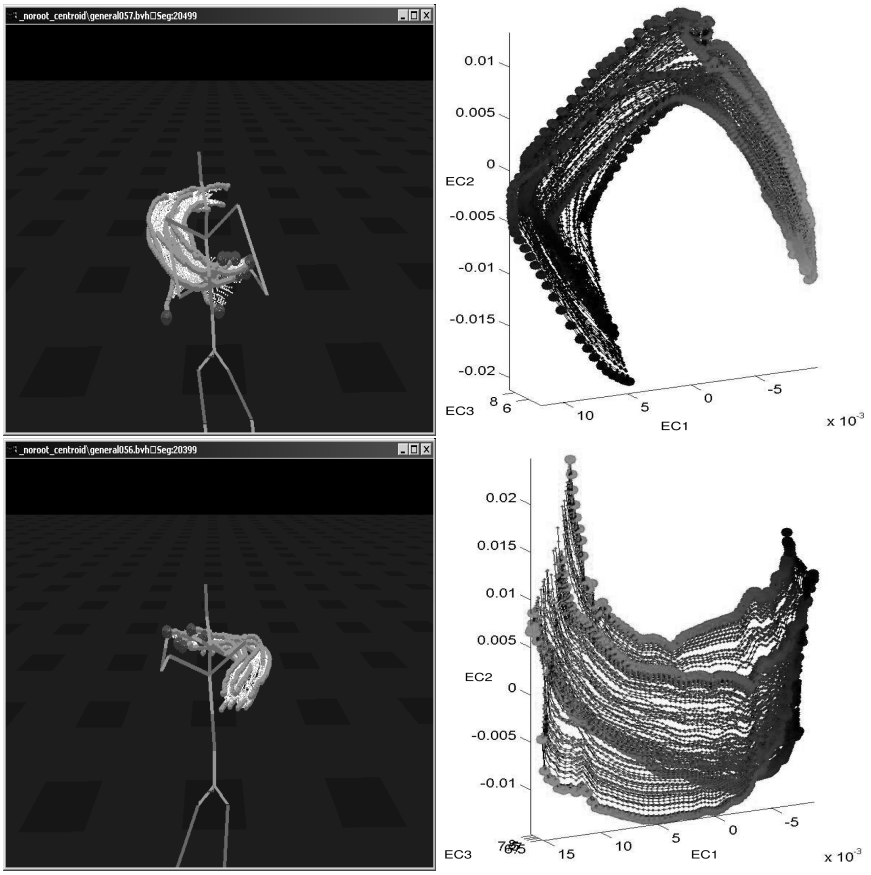


Fig. 15. Illustration of the "Cabbage Patch" meta-level behavior from Input Motion 1. The Cabbage Patch is a dance consisting of two primitives for "swing inward" (top) and "swing outward" (bottom).

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, a direct result from the definition of common temporal neighbors.

We do not propose PDBV as a replacement for human judgment or effort, but rather as a means for automated initialization. Because our method is model-free and exemplar-based, every step in the PDBV process is easily amenable to manual refinement and postprocessing.

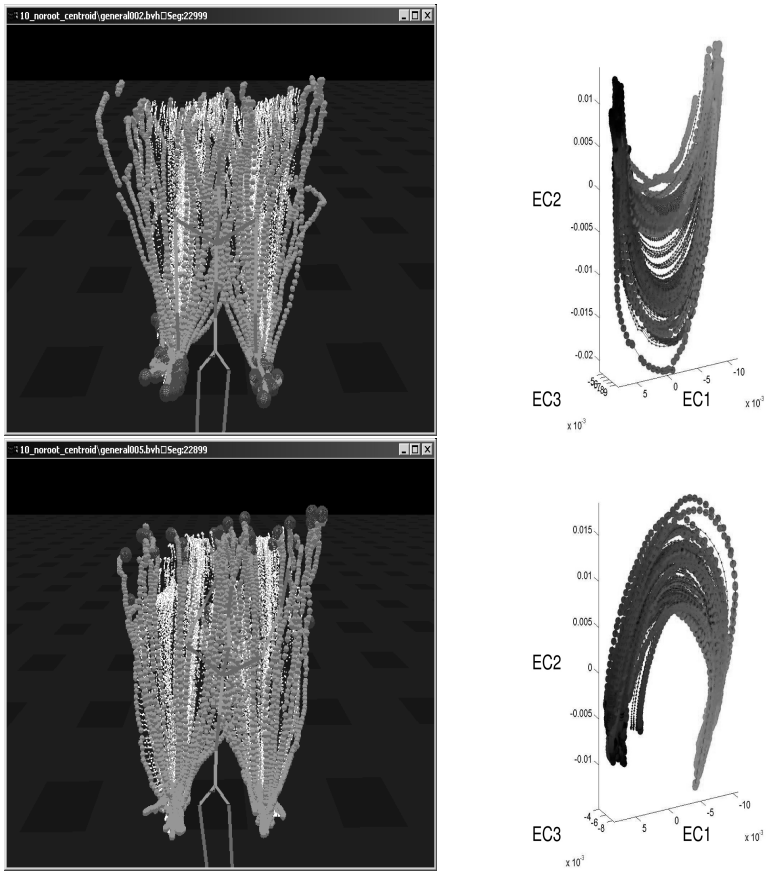


Fig. 16. Illustration of the single meta-level behavior from Input Motion 3. This behavior is for reaching and consists of two primitives for "reach to position" (top) and "return to rest posture" (bottom).

5. Conclusion

We have presented Performance-Derived Behavior Vocabularies (PDBV) as a methodology for automatically deriving behavior vocabularies to serve as skill-level interfaces for autonomous humanoids. Modular behavior vocabularies derived by our methodology are substrates for endowing humanoids with autonomy for a variety of functions, such as those used in constructing perceptual-motor algorithms. Our methodology derives these skills from motion data of human performance, leveraging the structure underlying human motion in a data-driven manner. Behaviors underlying human motion data are uncovered through unsupervised learning, using our extension of Isomap for spatio-temporal data dependencies. Behavior vocabularies are realized as a modular set of exemplar-based behaviors. The nonlinear dynamics of each behavior are expressed as a primitive flow field in the joint space

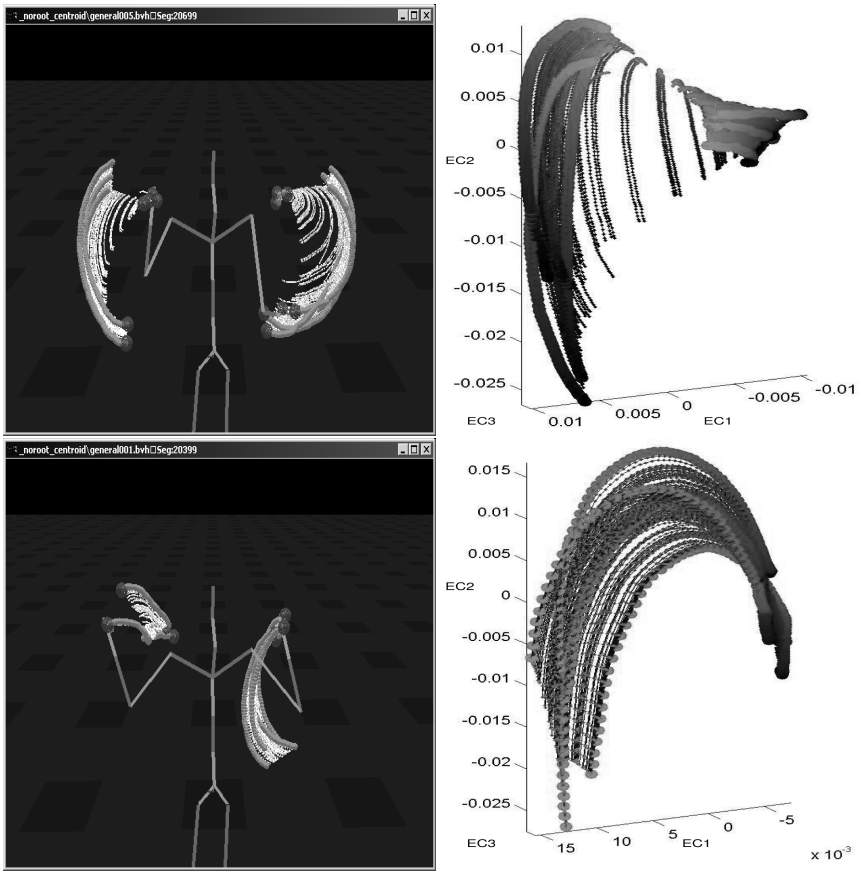


Fig. 17. Illustration of the "hand circle" meta-level behavior from Input Motion 1. Circling consists of two primitives for "circle up" (top) and "circle down" (bottom).

of the human. In future work, we will investigate the use of attractors as a means of incorporate user constraints with our primitive flow fields.

The PDBV methodology provides an automated data-driven means for modularizing motion into behaviors. We currently limit the scope of PDBV to free-space motion, allowing us to disregard issues of *i)* incorporating sensory information and *ii)* handling potential physical interactions between a humanoid and its environment. One area for future research is determining what types of information could be included in the derivation process to produce skills that address these issues, similar in aim to work by Peters et al.[56]

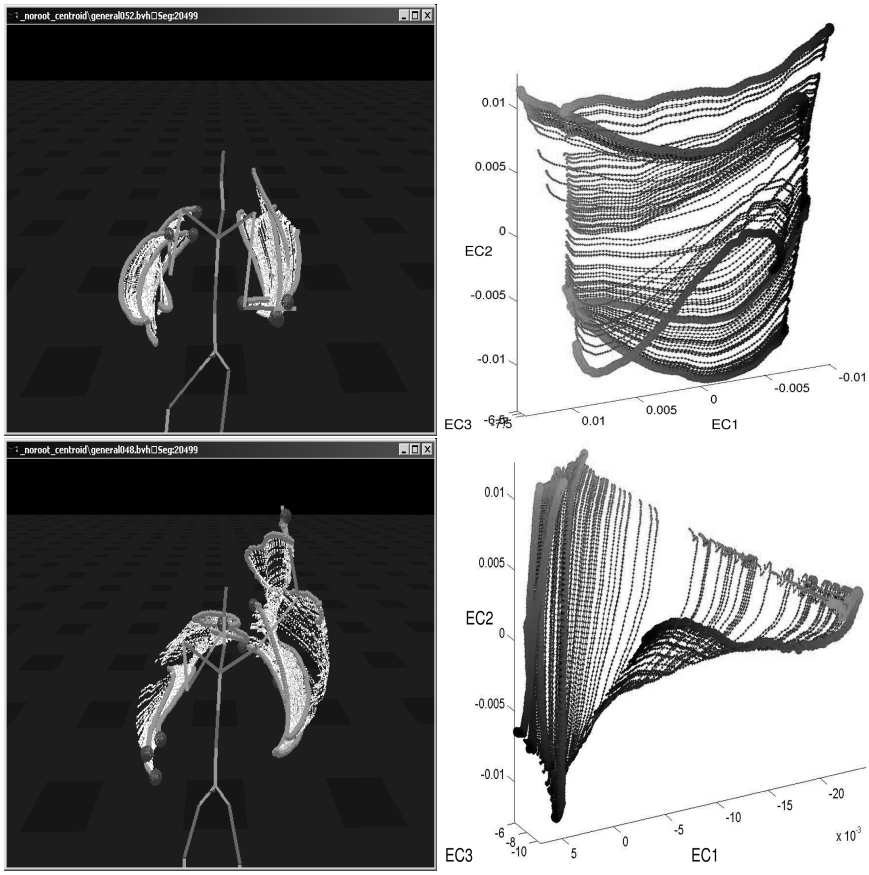


Fig. 18. Illustration of the "Twist" meta-level behavior from Input Motion 1. The Twist is a dance consisting of two primitives for "twist right" (top) and "twist left" (bottom).

Acknowledgements

This research was supported by the DARPA MARS Grant DABT63-99-1-0015, DARPA MARS2020 Grant 5-39509-A, and ONR MURI Grant N00014-01-1-0354. The authors are grateful to Jessica Hodgins for human motion data, and Sarah R. Jenkins, Dylan Shell, Jon Eriksson, and Roger D. Sealion for insightful feedback.

References

1. M. Huber and R. A. Grupen. A hybrid architecture for learning robot control tasks. *Robotics Today, Robotics International/Society of Manufacturing Engineers*, 13(4), 2000.
2. R. A. Brooks. Intelligence without representation. *Artificial Intelligence Journal*, 47:139–159, 1991.
3. M. J. Matarić. Behavior-based control: Examples from navigation, learning, and group

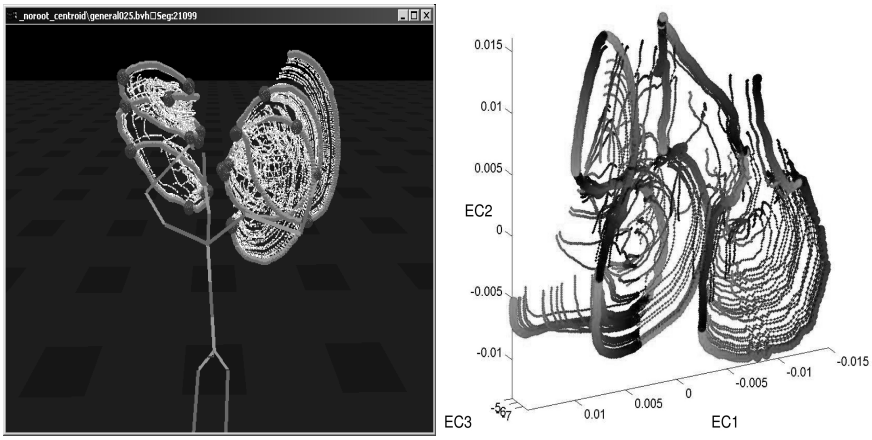


Fig. 19. Illustration of the "Itsy Bitsy Spider" behavior from Input Motion 1.

behavior. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2-3):323–336, 1997.

4. O. C. Jenkins and M. J. Matarić. A spatio-temporal extension to isomap nonlinear dimension reduction. In *To appear in the International Conference on Machine Learning (ICML 2004)*, Banff, Alberta, Canada, 2004.
5. C. Rose, M. F. Cohen, and B. Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics & Applications*, 18(5):32–40, Sep-Oct 1998. ISSN 0272-1716.
6. A. 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, Apr 1995.
7. M. J. Matarić. Sensory-motor primitives as a basis for imitation: Linking perception to action and biology to robotics. In C. Nehaniv and K. Dautenhahn, editors, *Imitation in Animals and Artifacts*, pages 392–422. MIT Press, 2002.
8. M. N. Nicolescu and M. J. Matarić. Natural methods for robot task learning: Instructive demonstration, generalization and practice. In *Autonomous Agents and Multi-Agent Systems (AAMAS 2003)*, pages 241–248, Melbourne, AUSTRALIA, July 2003.
9. D. C. Bentivegna and C. G. Atkeson. Learning from observation using primitives. In *IEEE International Conference on Robotics and Automation*, pages 1988–1993, Seoul, Korea, May 2001.
10. A. D'Souza, S. Vijayakumar, and S. Schaal. Learning inverse kinematics. In *IEEE Intelligent Robots and Systems (IROS 2001)*, pages 298–303, Maui, HI, USA, 2001.
11. 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.
12. J. J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley, Reading, MA, USA, 1989.
13. 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.
14. L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. *ACM Transactions on Graphics*,

21(3):473–482, 2002.
15. O. C. Jenkins and M. J. Matarić. Automated derivation of behavior vocabularies for autonomous humanoid motion. In *Autonomous Agents and Multiagent Systems (AAMAS 2003)*, pages 225–232, Melbourne, Australia, Jul 2003.
16. Y. Li, T. Wang, and H.-Y. Shum. Motion texture: a two-level statistical model for character motion synthesis. *ACM Transactions on Graphics*, 21(3):465–472, 2002.
17. J. Ijspeert, J. Nakanishi, and S. Schaal. Learning rhythmic movements by demonstration using nonlinear oscillators. In *IEEE Intelligent Robots and Systems (IROS 2002)*, pages 958–963, Lausanne, Switzerland, Oct 2002.
18. M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.
19. A. Fod, M. Matarić, and O. Jenkins. Automated derivation of primitives for movement classification. *Autonomous Robots*, 12(1):39–54, Jan 2002.
20. H. Sidenbladh, M. J. Black, and L. Sigal. Implicit probabilistic models of human motion for synthesis and tracking. In *European Conference on Computer Vision*, volume 1, pages 784–800, Copenhagen, Denmark, 2002.
21. M. G. Choi, J. Lee, and S. Y. Shin. Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Transactions on Graphics*, 22(2):182–203, 2003.
22. L. G. Y. Song and P. Perona. Unsupervised learning of human motion. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(7):814–827, 2003.
23. J. Rittscher and A. Blake. Classification of human body motion. In *IEEE International Conference on Computer Vision*, pages 634–639, Kerkyra, Corfu, Greece, Sep 1999.
24. C. Bregler. Learning and recognizing human dynamics in video sequences. In *IEEE Computer Vision and Pattern Recoginition*, pages 568–574, San Juan, Puerto Rico, 1997.
25. M. Williamson. Postural primitives: Interactive behavior for a humanoid robot arm. In *International Conference on Simulation of Adaptive Behavior*, pages 124–131, Cape Cod, MA, USA, 1996. MIT Press.
26. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *IEEE International Conference on Robotics and Automation (ICRA 2002)*, pages 1398–1403, Washington, D.C., USA, May 2002.
27. M. Williamson. *Robot Arm Control Exploiting Natural Dynamics*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1999.
28. S. Schaal and D. Sternad. Programmable pattern generators. In *International Conference on Computational Intelligence in Neuroscience*, pages 48–51, Research Triangle Park, NC, USA, Oct 1998.
29. D. Chi, M. Costa, L. Zhao, and N. Badler. The emote model for effort and shape. In *Conference on Computer Graphics and Interactive Techniques*, pages 173–182. ACM Press/Addison-Wesley Publishing Co., 2000.
30. R. A. Grupen, M. Huber, J. A. C. Jr., and K. Souccar. A basis for distributed control of manipulation tasks. *IEEE Expert*, 10(2):9–14, 1995.
31. M. J. Matarić, V. B. Zordan, and M. 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, Mar 1999.
32. O. C. Jenkins, M. Matarić, and S. Weber. Primitive-based movement classification for humanoid imitation. In *IEEE International Conference on Humanoid Robots (Humanoids 2000)*, Cambridge, MA, USA, Oct 2000.
33. O. Arikan, D. A. Forsyth, and J. F. O'Brien. Motion synthesis from annotations. *ACM Transactions on Graphics*, 22(3):402–408, 2002.

34. A. Billard and M. J. Matarić. Learning human arm movements by imitation: Evaluation of a biologically inspired connectionist architecture. *Robotics and Autonomous Systems*, 37(2):145–160, Nov 2001.
35. L. Molina-Tanco and A. Hilton. Realistic synthesis of novel movements from a database of motion capture data. In *IEEE Workshop on Human Motion (HUMO 00)*, pages 137–142, Austin, Texas, USA, Dec 2000.
36. R. Bowden. Learning statistical models of human motion. In *IEEE Workshop on Human Modelling, Analysis and Synthesis*, Hilton Head Island, SC, USA, 2000.
37. M. Brand and A. Hertzmann. Style machines. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 183–192. ACM Press, Jul 2000.
38. M. Okada and Y. Nakamura. Design of the continuous symbol space for the intelligent robots using the dynamics-based information processing. In *IEEE International Conference on Robotics and Automation (ICRA 2004)*, pages 3201–3206, 2004.
39. W. Iba. Learning to classify observed motor behavior. In *International Joint Conference on Artificial Intelligence (IJCAI 1991)*, pages 732–738, Sydney, New South Wales, Australia, 1991.
40. Y. Demiris and G. 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.
41. D. M. Wolpert and M. Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11(7-8):1317–1329, 1998.
42. W. Ilg, G. H. Bakir, J. Mezger, and M. A. Giese. On the representation, learning and transfer of spatio-temporal movement characteristics. In *IEEE International Conference on Humanoid Robotics (Humanoids 2003)*, Karlsruhe, Germany, Oct 2003.
43. L. Kovar and M. Gleicher. Automated extraction and parameterization of motions in large data sets. *To appear in ACM Transactions on Graphics*, 23(3), 2004.
44. L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–285, Feb 1989.
45. J. Cutting, D. Proffitt, and L. Kozlowski. A biomechanical invariant for gait perception. *Journal of Experiemental Psychology: Human Perception and Performance*, 4(3):357–372, Aug 1978.
46. J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.
47. A. Jain and R. Dubes. *Algorithms For Clustering Data*. Prentice Hall, Englewood Cliffs, NJ, USA, 1988.
48. J. D. Cohen, M. C. Lin, D. Manocha, and M. K. Ponamgi. I-COLLIDE: An interactive and exact collision detection system for large-scale environments. In *Symposium on Interactive 3D Graphics*, pages 189–196, 218, Monterey, California, USA, 1995.
49. D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the ACM National Conference*, pages 517–524. ACM Press, 1968.
50. 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, 2003.
51. D. Baraff and A. Witkin. Physically-based modeling, principles and practice. Technical Report Course 34, SIGGRAPH 1997 Course Notes, ACM, Aug 1997.
52. D. Sternad and S. Schaal. Segmentation of endpoint trajectories does not imply segmented control. *Experimental Brain Research*, 124(1):118–136, 1999.
53. 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, Jul-Aug. 2000.

54. N. Pollard, J. K. Hodgins, M. Riley, and C. G. Atkeson. Adapting human motion for the control of a humanoid robot. In *IEEE International Conference on Robotics and Automation (ICRA 2002)*, pages 1390–1397, May 2002.
55. 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, MN, USA, 1998. ACM Press.
56. R. Peters, C. Campbell, W. Bluethmann, and E. Huber. Robonaut task learning through teleoperation. In *IEEE International Conference on Robotics and Automation (ICRA 2003)*, pages 2806–2811, 2003.

Chad Jenkins is a postdoctoral researcher at the Center for Robotics and Embedded Systems at the University of Southern California. Chad will be an assistant professor of Computer Science at Brown University as of Fall 2004. He earned his B.S. in Computer Science and Mathematics at Alma College (1996), M.S. in Computer Science at Georgia Tech (1998), and Ph.D. in Computer Science at the University of Southern California (2003). His dissertation pertained to model-free methods for capture, analysis, and modeling of kinematic motion. His research interests include humanoid robotics, machine learning, computer animation, computer vision, autonomous agents, and interactive entertainment.

Maja Matarić is an associate professor in the Computer Science Department and the Neuroscience Program at the University of Southern California, Founding Director of the USC Center for Robotics and Embedded Systems, and Co-Director of the USC Robotics Research Lab. She received her Ph.D. in Computer Science and Artificial Intelligence from MIT in 1994, her M.S. in Computer Science from MIT in 1990, and her B.S. in Computer Science from the University of Kansas in 1987. She is a recipient of the NSF Career Award, the IEEE Robotics and Automation Society Early Career Award, the MIT TR100 Innovation Award, and the USC School of Engineering Junior Research Award. She is an associate editor of three major journals and has published over 31 journal articles, 17 book chapters, 4 edited volumes, 91 conference papers, and 22 workshop papers, and has two books in the works with MIT Press. Her research is aimed at endowing robots with the ability to help people and involves systems ranging from individual assistants (for convalescence, training, education, companionship, etc.) to cooperative robot teams (for habitat monitoring, emergency response, etc.) and problems of intelligent control and learning in complex, high dimensional/high degree of freedom systems that integrate perception, representation, and interaction with people. Research details are found at http://robotics.usc.edu/interaction/