In *Autonomous Robots, 12(1), Jan. 2002, pp. 39-54*

Automated Derivation of Primitives for Movement Classification

Ajo Fod, Maja J Mataric, and Odest Chadwicke Jenkins

University of Southern California, Los Angeles CA, USA,
mataric|cjenkins@cs.usc.edu,
http://robotics.usc.edu/~agents/imitation.html

Abstract. We describe a new method for representing human movement compactly, in terms of a linear superimposition of simpler movements termed *primitives*. This method is a part of a larger research project aimed at modeling motor control and imitation using the notion of perceptuo-motor primitives, a basis set of coupled perceptual and motor routines. In our model, the perceptual system is biased by the set of motor behaviors the agent can execute. Thus, an agent can automatically classify observed movements into its executable repertoire. In this paper, we describe a method for automatically deriving a set of primitives directly from human movement data.

We used movement data gathered from a psychophysical experiment on human imitation to derive the primitives. The data were first filtered, then segmented, and principal component analysis was applied to the segments. The eigenvectors corresponding to a few of the highest eigenvalues provide us with a basis set of primitives. These are used, through superposition and sequencing, to reconstruct the training movements as well as novel ones. The validation of the method was performed on a humanoid simulation with physical dynamics. The effectiveness of the motion reconstruction was measured through an error metric. We also explored and evaluated a technique of clustering in the space of primitives for generating controllers for executing frequently used movements.

1 Introduction

Programming and interacting with robots, especially humanoid ones, is a complex problem. Using learning to address this problem is a popular approach, but the high dimensionality of humanoid control makes the approach prohibitively slow. Imitation, the process of learning new movement patterns and skills by observation, is a promising alternative. The ability to imitate enables a robot to greatly reduce the space of possible trajectories to a subset that approximates that of the observed demonstration. Refinement through trial and error is still likely to be required, but in a greatly reduced learning space.

We have developed a model for learning by imitation [16, 23, 19], inspired by two lines of neuroscience evidence: *motor primitives* [5] and *mirror neurons* [14]. Motor primitives are biological structures that organize the underlying mechanisms of complete movements, including spinal fields [5] and central pattern generators [32]. In a computational sense, primitives can be viewed as a basis set of motor programs that are sufficient, through combination operators, for generating entire movement repertoires. Several properties of human movement patterns are known, including smoothness [12, 9, 35], inter-joint torque constraints [10], and the "$2/3$ power law" relating speed to curvature [36]. It is not clear how one would go about creating motor primitives with such knowledge alone.

Mirror neurons are structures that link the visual and motor systems in the context of complete, generalized movements, such as reaching and grasping. In our model, the ability to imitate is based on such a mapping mechanism; the system automatically classifies all observed movements onto its set of *perceptuo-motor primitives*, a biologically-inspired basis set of coupled perceptual and motor routines. This mapping process also serves as a substrate for more complex and less direct forms of skill acquisition. In this view, primitives are the fundamental building blocks of motor control, which impose a bias on movement perception and facilitate its execution, i.e., imitation [19].

Determining an effective basis set of primitives is thus a difficult problem. Our previous work has explored hand-coded primitives [22, 16]. Here, we describe a method for automatically generating a set of arm movement primitives from human movement data. We hypothesize that the trajectory executed by the arm is composed of segments in which a set of principal components are active. We first segment movement data and then apply principal component analysis on the resulting segments to obtain primitives. The eigenvectors corresponding to a few of the highest eigenvalues provide us with a basis set of such primitives. The projection of a new segment onto this subspace contains most of the information about that segment, but is a lower-dimensional, more compact representation of it. To evaluate the method of movement encoding in terms of eigenmovement primitives, and the subsequent reconstruction of the original movements or entirely novel ones, we used a mean squared deviation

metric. Finally, we also demonstrated the movements on a dynamic humanoid simulation, reconstructing both the training set and entirely novel test movements.

The rest of the paper is organized as follows. In Section 2, we place our work in the context of relevant research. A brief description of our imitation model is given in Section 3. The psychophysical experiment from which the movement data were drawn is described in Section 4. The methods used to analyze the data are presented in Section 5. Section 6 discusses controller design and movement clustering. Section 7 introduces the evaluation metric and the performance of the movement reconstruction. Section 8 describes the validation of the derived primitives on a humanoid simulation. In Section 9 summarizes the paper and our continuing research.

2 Motivation and Related Work

Much of human visual experience is devoted to watching other humans manipulate the shared environment. If humanoid or similarly articulated robots are to work in the same type of environments, they will need to recognize other agents' actions and to dynamically react to them. To make this possible, motor behaviors need to be classified into higher-level representations. The perceptuo-motor primitives we present are such a representation. Understanding motor behavior, then, becomes a process of classifying the observed movements into the known repertoire, a natural basis for imitation.

In our approach, the motor repertoire consists of a collection of movement primitives, behaviors that accomplish complete goal-directed actions [20, 18, 19]. These behaviors are used to structure movement as well as bias and classify movement perception. Thus, the primitives, in our imitation model, are perceptuo-motor, unifying visual perception and motor output, through a direct mapping between the two representational spaces. While this model is inspired by neuroscience evidence, its specific structure and the methods we describe for deriving the primitives are not intended to model any particular biological processes.

To move, the vertebrate central nervous system (CNS) must transform information about a small number of variables to a large number of signals to many muscles. Any such transformation might not be unique. Bizzi et al [5] performed experiments to show that inter-neuronal activity imposes a specific balance of muscle activation. These synergies lead to a finite number of force patterns, which they called convergent force fields. The CNS uses such synergies to recruit a group of muscles simultaneously to perform any given task. This evidence has inspired work in motor control of mobile robots using schemas [3, 2, 4] and our own work on basis behaviors [20, 18]. Sanger [29] demonstrated such motor synergies by applying PCA to analyze trajectories followed by a human arm while the wrist traces a curve on a plane. We use a similar approach here, but on higher-dimensional free movements, for the purposes of movement encoding for subsequent recognition and classification. In mobile robotics, Pierce and Kuipers [27] proposed a method of abstracting control of a system using PCA to determine motor features that are associated with sensory features.

Other studies and theories about motor primitives [24, 25, 14] suggest that they are viable means for encoding humanoid movement. Thoroughman and Shadmehr [33] provide neuroscience evidence in support of motor primitives. They argue for a localized sensori-motor map from position, velocity, and acceleration to the torques at the joints. In contrast, our method for generating primitives produces a fixed set of trajectories for which controllers can be generated. Schaal et al [31, 30] discuss a set of primitives for generating control commands for any given motion by modifying trajectories appropriately or generating entirely new trajectories from previously learned knowledge. In certain cases it is possible to apply scaling laws in time and space to achieve the desired trajectory from a set of approximations. The described elementary behaviors consist of two types of movements, discrete and rhythmic. While these are well suited for control, it is not obvious how the robot would generalize from the input space of visual data. In contrast, in our model, primitives are the representation used for generalizing visual inputs, i.e., inspired by the function of mirror neurons, they combine perceptual and motor components.

Segmentation and classification become key inter-related processes of movement interpretation. Segmentation of visual data is a general problem in computer vision. Brand [6] discusses a method for "gisting" a video by reasoning about changes in motions and contact relations between participants. In that work, usually only the agent (typically a part of a human arm) and any objects under its control were segmented. Our other work [16] applies a method of segmenting visual data manually into primitives. Most related to the work presented here is the approach to face classification and recognition using so-called "eigenfaces" [34]. There, principle component analysis was applied to a set of static images of faces; in this work we apply PCA toward classification and encoding of time-extended multi-joint movements.

The current testing environment for our research is a humanoid simulation with dynamics, Adonis. As the development of physical humanoid systems advances [15, 11, 1], such physical testbeds will become more accessible for further evaluation and application of our research.

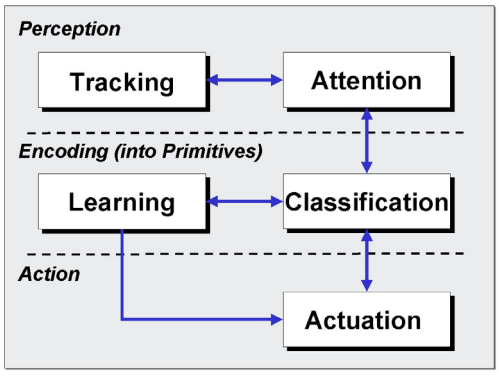

Fig. 1. *Our imitation model.*

3 The Imitation Model

Here, we describe in more detail our model for imitation using perceptuo-motor primitives [19]. The model consists of five main subcomponents: Tracking, Attention, Learning, Classification, and Actuation. These components are divided into three layers: Perception, Encoding, and Action. Figure 1 illustrates the structure of the model.

3.1 Perception

The Perception layer consists of two components, Tracking and Attention, that serve to acquire and prepare motion information for processing into primitives at the Encoding layer. The Tracking component is responsible for extracting the motion of features over time from the perceptual inputs.

In general, the choice of a primitive set is constrained by the types of information that can be perceived. Naturally, extending the dimensionality and types of information provided by the Tracking component increases the complexity of other components. For instance, if 3D location information is used, the primitive set generated by the Learning component must provide more descriptions to appropriately represent the possible types of motion.

The Attention component is responsible for selecting relevant information from the perceived motion stream in order to simplify the Classification and Learning components. This selection process is performed by situating the observed input into meaningful structures, such as a set of significantly moving features or salient kinematic substructures. Various attentional models can be applied in addition to our segmentation method [26].

3.2 Classification and Learning

The Encoding layer encompasses Classification and Learning, which classify observed motions into appropriate primitives. The primitives, in turn, can be used to facilitate segmentation. The Learning component serves to determine and refine the set of primitives. The Classification component then uses the updated set of primitives for movement encoding.

The Encoding layer provides two outputs to the Action layer: 1) the sequence of segments representing a motion (from Classification), and 2) a set of constraints for creating motor controllers for each primitive in the primitive set (from Learning). The list of segments is especially important because it describes intervals in time where a given primitive is active.

A discussion of key issues, including time and space invariance of the primitives and the choice of their representation for facilitating classification and learning is found in [16].

3.3 Action

The final layer, Action, consists of a single component, Actuation, which performs the imitation by executing the list of segments provided by the Classification component. Ideally, primitive controllers should provide control commands independently, which can then be combined and/or superimposed through motor actuation. The design of such controllers is one topic of research we are pursuing.

As noted above, the motor controller components of the primitives may be manually derived or learned. In both cases, to be general, they must be characterized in parametric form. In this paper we address learning of an initial set of primitives directly from movement data.

In this paper, we address three components of the above-described imitation model: Attention, Learning, and Classification. Attention is implemented in the form of a method for motion segmentation. Learning is addressed only at the level of deriving an initial set of primitives from the segmented data, not subsequent expansion and refinement of that set. The latter is another active area of research we are pursuing. Classification of new motion is performed for encoding the motion into the primitives and for reconstruction of the original observed movement.

4 Human Movement Data

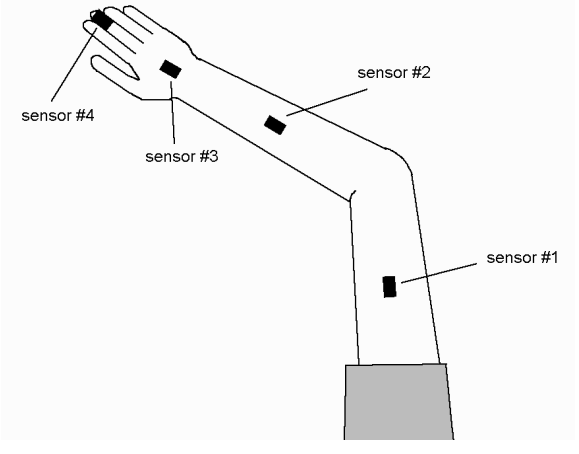

Fig. 2. Location of Polhemus FastTrak sensors on the subject's arm.

We used human movement data as the basis for deriving the movement primitives. The data were collected in a psychophysical experiment in which 11 college-age subjects watched and imitated 20 short videos (stimuli) of arm movements[1] [28]. Each stimulus was a 3 to 5-second long sequence of arm movements, presented against a black background, shown on a video monitor. The subjects were asked to imitate a subset of the presented movements with their right arm, in some cases repeatedly, and their imitations were tracked with the Polhemus FastTrak motion tracking system. Position sensors, about $2\times1\times1$ cm in size, were fastened with elastic bands at four points on each subject's right arm, as shown in Figure 2:

1. The center of the upper arm,
2. The center of the lower arm,
3. Above the wrist,
4. The phalanx of the middle finger.

Thus, the tracking data consisted of the joint angles over time of the 4 DOF of a human arm: the shoulder flex extend (SFE), shoulder adduction abduction (SAA), humeral rotation (HR), and elbow rotation (ER) [17]. SFE is the up-and-down movement of the arm; SAA represents movement of the projection of the arm onto the horizontal plane; HR is rotation about an axis passing through the upper arm, and ER is the flexion/extension of the elbow joint.

The positions of the sensors were measured every 34ms, with an accuracy of about 2mm. The measured 3D coordinates of the four sensors were recorded for each of the subject's imitations, for all subjects, along with a time stamp for each measurement. The details of the psychophysical experiment and its results are described in [28].

[1] The data were gathered by M. Matarić and M. Pomplun in a joint interdisciplinary project conducted at the NIH Resource for the Study of Neural Models of Behavior, at the University of Rochester.

5 Methods

This section describes our approach and the methods involved. First, we applied inverse kinematics to transform extrinsic Cartesian 3D marker coordinates into intrinsic joint angle representation. We applied filtering to remove the incidental random zero-velocity points caused by noise. Next, we segmented the movement data so that finite-dimensional vectors could be extracted for subsequent principal component analysis (PCA). By applying PCA, we obtained a set of eigenvectors, a small subset of which represents almost all the variance in the data. The details of each of these methods are given in the following sections. Figure 3 summarizes the key steps of the approach, and relates them to our general imitation model.

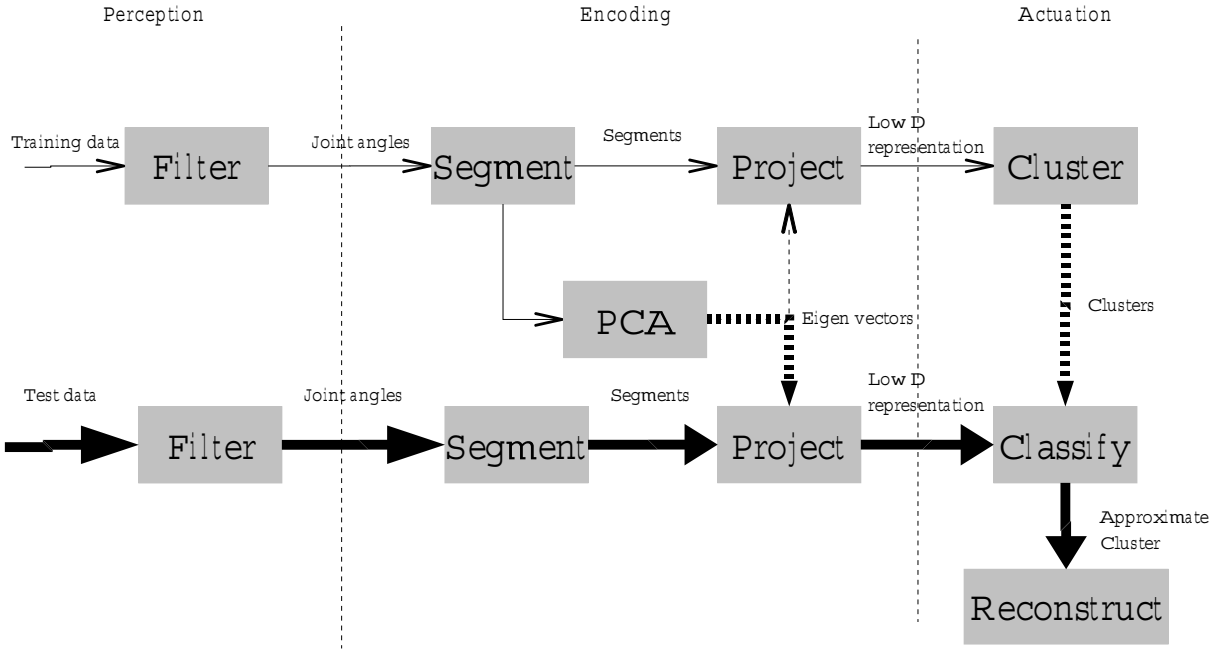

Fig. 3. Components of our system. Thin lines represent the flow of data in the training phase. Thick lines represent the flow of data in the testing phase. All dashed lines represent the use of previously computed data. The three layers of the imitation model are shown vertically, as they relate to the specific system components.

5.1 Inverse Kinematics

The raw movement data we used were in the form of 3D Cartesian marker positions as a function of time. Due to human kinematic constraints imposed on the joints, all the data can be compressed into fewer variables than required by a 3D representation of the wrist and elbow. For example, since the elbow lies on the surface of a sphere centered at the shoulder, we need only two degrees of freedom to completely describe the elbow when the position of the shoulder is known. Similarly, the wrist can only lie on the surface of a sphere centered at the elbow. Thus, the configuration of the arm can be completely represented by four variables henceforth referred to as angles.

The Euler angle representation we used describes the position of the arm in terms of a sequence of rotations around one of the principal axes. Each rotation transforms the coordinate frame and produces a new coordinate frame in which the next transformation is performed. The first transformation $R_{z\gamma}$ rotates the coordinate frame through γ about the z-axis. Subsequently, the rotations $R_{y\beta}$ about the y-axis and $R_{z\alpha}$ determine the configuration of the shoulder. If the rotation of the elbow is specified, the position of the elbow and the wrist relative to the arm are completely determined.

5.2 Filtering

The movement data we used contained two distinct types of noise that had to be filtered out: drift and high frequency components. Drift, the low frequency component of the noise, is primarily caused by the movement of the subject's shoulder. We assumed that it affected all marker positions on the arm uniformly, and made the shoulder the origin of our coordinate frame, thereby eliminating this noise component. The high frequency component of noise can be attributed to the elastic vibrations in the fasteners used to attach the markers to the arm. Passing the signal through a Butterworth filter of order 10 and normalized cutoff frequency of 0.25 reduced this component of the noise considerably (based on an empirically-derived level of tolerance). After these transformations, the resulting angles are a smooth function of time.

5.3 Segmentation

Signal segmentation is notoriously difficult, and thus segmentation of the arm movement data was one of the most challenging aspects of this work. In our approach, it is necessary to segment the movement data so PCA can be applied to extract common features. In the context of movement representation and reconstruction, the segmentation algorithm needs to have the following properties:

1. *Consistency*: Segmentation produces identical results in different repetitions of the action. For this, we defined a set of features which could be used to detect segment transitions.
2. *Completeness*: Ideally, segments do not overlap and a set of segments contains enough information to reconstruct the original movement completely.

A great deal of previous work has dealt with segmentation of time-series data. Brand et al [6, 7] present an event-based method for continuous video segmentation; segments are introduced when a certain feature or contact configuration changes. Wada et al [37] introduce a via-point method of segmenting handwriting, speech, and other temporal functions. Pomplun & Matarić [28] use a segmentation scheme based on the root mean squared (RMS) value of velocity in different joints. To reduce the distance between two movements that could be similar, they propose a modified scheme that minimizes that distance to produce the best segmentation. We consider a comparatively simpler approach to segmentation here which proved sufficient.

Our first attempt at segmentation involved the use of the 4D parametric curve plotting the change of the joint angles over time. The radius of curvature is defined as the radius of the largest circle in the plane of the curve that is tangential to the curve. Curvature has the convenient property of being a scalar function of time. Thus, when the radius is low, it could imply a change of desired state and hence can be used as a segmentation cue. Unfortunately, calculation of the radius of curvature of a trajectory becomes intractable as the dimensionality of the data increases. Furthermore, choosing an appropriate threshold for segmenting the curvature is difficult. As a result, we abandoned this segmentation method for two simpler alternatives, presented next.

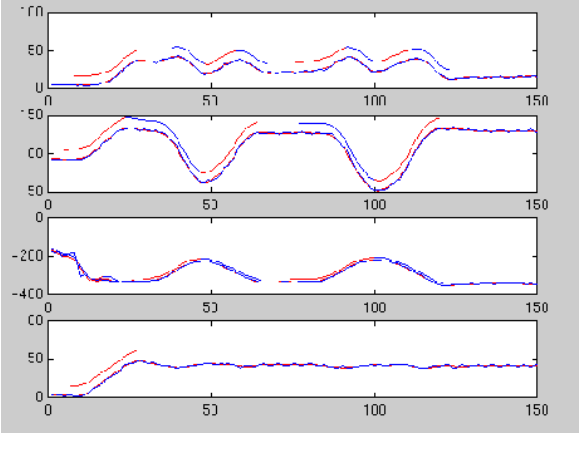

Fig. 4. Segmenting using zero-velocity crossings. Variables are plotted along the y-axis against time on the x-axis. The four sub-plots are of the four angles that we analyzed. The curved lines above the movement plots represent individual extracted segments.

The segmentation routines we designed are based on the angular velocity of different DOFs. The velocity reverses in response to the subject changing direction of movement. Whenever this happens, the point is labeled as a zero-velocity crossing (ZVC). Each ZVC is associated with one DOF. We assume that all movement in any DOF occurs between ZVCs. If the movement is significant, based on an empirically-determined threshold, it is recorded as a segment. This thresholding aids in eliminating small oscillations. ZVCs could flank a segment where almost no movement occurs. Ideally, these segments should be discarded since they contain no movement information other than that the angle needs to be held constant. Spurious ZVCs could also be introduced by noise. Figure 4 shows an example of the result of ZVC segmentation of a subject's imitation of a stimulus video. Significant movement is assumed to occur when the change in the angle is above a certain threshold. The short curves above the continuous curves indicate the instances when such segments are extracted.

Toward the goal of deriving primitives, we seek properties common to all DOFs. Fortunately, most of the time, the ZVCs in different DOFs coincide in many of the movement samples. Due to this property of continuous smooth movement in particular, we segmented the data stream at the points where more than one DOF has a ZVC separated by less than 300ms, an empirically-derived threshold; we call this the SEG-1 routine. To avoid including random segments, whose properties cannot be easily determined, we kept only those segments whose start and end both had this property. This results in dropping some segments, thereby compromising the completeness property of the segmentation algorithm; we would be unable to reconstruct all movements completely. The movements represented in such dropped segments have certain common properties. Specifically, the end-points are likely to involve little movement, and there is a directed effort to get to a new configuration of joint angles in the interim. It is also very likely that the limb is moving in a nearly constant direction during such segments. The direction reversals are likely to be points where intentions and control strategies change.

The segmentation routine SEG-1 we described is not complete, i.e., it does not guarantee that any input movement will be completely partitioned. There is considerable overlap when two joint angles cross their zeros simultaneously while and the other two cross their zeros at a different point in time. For some actions, the zero-crossings may not coincide for the majority of the movement. This is possible either when the some of the zeros are missed by the zero marking procedure or when the zeros fail to align. The latter problem arises for specific classes of movements in which at least one of the DOFs is active when another is at its ZVC, such as in repetitions of smooth curves. For example, repeatedly tracing out a circle would result in no boundaries detected by SEG-1.

Though SEG-1 explains the formation of primitives in well-defined segments, it cannot reliably reconstruct outside those segments because the segmentation does not result in a complete partition of the movement space. It might be possible to improve the segmentation for reaches. However, that would not solve the problems caused by potential misalignment of the zero-crossings themselves. It might also be possible to arrive at a different segmentation scheme for the parts that are not amenable to this form of segmentation. This would essentially divide the space of movement into reaches and non-reaches, consistent with literature in motor control [31].

In order to produce a more complete segmentation, we developed a second segmentation algorithm, called SEG-2, based on the sum of squares of angular velocity z, as follows:

$$z = \dot{\theta_1^2} + \dot{\theta_2^2} + \dot{\theta_3^2} + \dot{\theta_4^2} \tag{1}$$

where θ_i is an angle. Figure 5 plots the variation of z over time for a single movement stimulus. The SEG-2 algorithm segments whenever z is lower than a threshold we derived empirically from the movement data. As noted above, the subjects were asked to repeat some of the demonstrations multiple times in a sequence. We adjusted the threshold so as to obtain about as many segments in each movement as the number of repetitions of the action in the sequence.

The resulting segments had the common property that z was high within a segment and low at the boundaries. Thus, low values of z mark segment boundaries. Because of the way we determined the threshold, this method of segmentation was highly reliable. The resulting segments do not encode the entire movement because there are portions where z is low for an extended period of time, indicating a stationary pose.

In Section 7 we compare the performance of the two segmentation algorithms.

5.4 Principal Component Analysis

The segmentation algorithms above converted the input movement data into a list of segments of varying lengths. In order to perform PCA on the segments, we first converted the movement to vector form. For each DOF, the movement within a segment is interpolated to a fixed length, in our implementation to 100 elements. The elements are then represented in vector form and the vectors for all DOFs concatenated to form a single vector $\mathbf{s}$ of 400 elements. This is effectively a 400D representation of the movement within each segment.

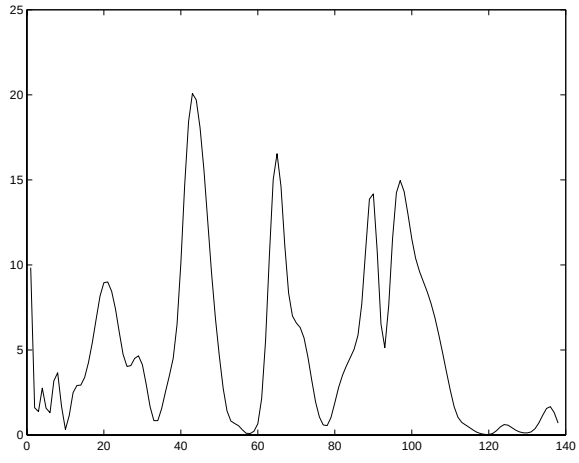

Fig. 5. A plot of the variation of the sum of squares of velocities (y-axis) as a function of the time (x-axis).

The mean of N segments $\overline{\mathbf{s}}$ in the input vector is computed as follows:

$$\overline{\mathbf{s}} = \frac{\sum_{i=1}^{N} \mathbf{s_i}}{N} \tag{2}$$

Since we have about 250 vectors in our data set, the covariance matrix K can be written as

$$\mathbf{K} = \frac{\sum_{i=1}^{N} (\mathbf{s_i} - \overline{\mathbf{s}})(\mathbf{s_i} - \overline{\mathbf{s}})^T}{N} \tag{3}$$

where T is the transpose operator and N is the number of segments.

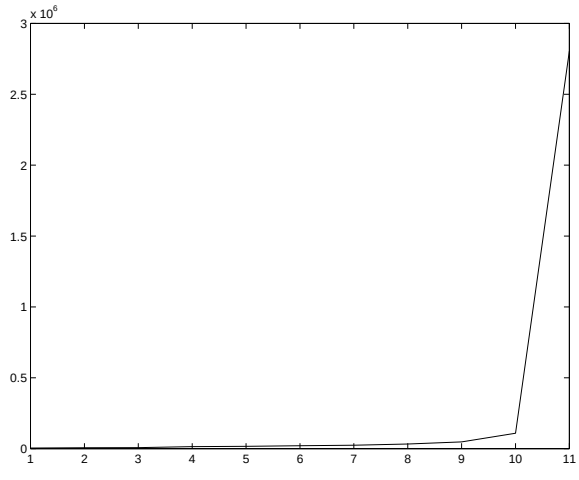

Fig. 6. The magnitude of the most significant 11 eigenvalues for our training set. X-axis presents the index of the eigenvectors, the y-axis the associated eigenvalue.

The principal components of the vector are the eigenvectors of **K**. If **s** is d-dimensional, then there are d such eigenvectors. Most of the variance in the movements is captured by the few eigenvectors (say f vectors) that have the highest eigenvalues. Figure 6 illustrates this property; it shows the most significant 11 eigenvectors obtained by sorting the 400 principal components of **K** by increasing magnitude. Thus, we can describe the vector **s** in terms of the projection of the vector along these f eigenvectors $\mathbf{E_f}$, in other words, an f-dimensional vector **p**:

$$\mathbf{p} = \mathbf{E_f^T s} \tag{4}$$

This results in an f-dimensional representation of each segment. The first four of these eigenvectors are shown in Figure 7.

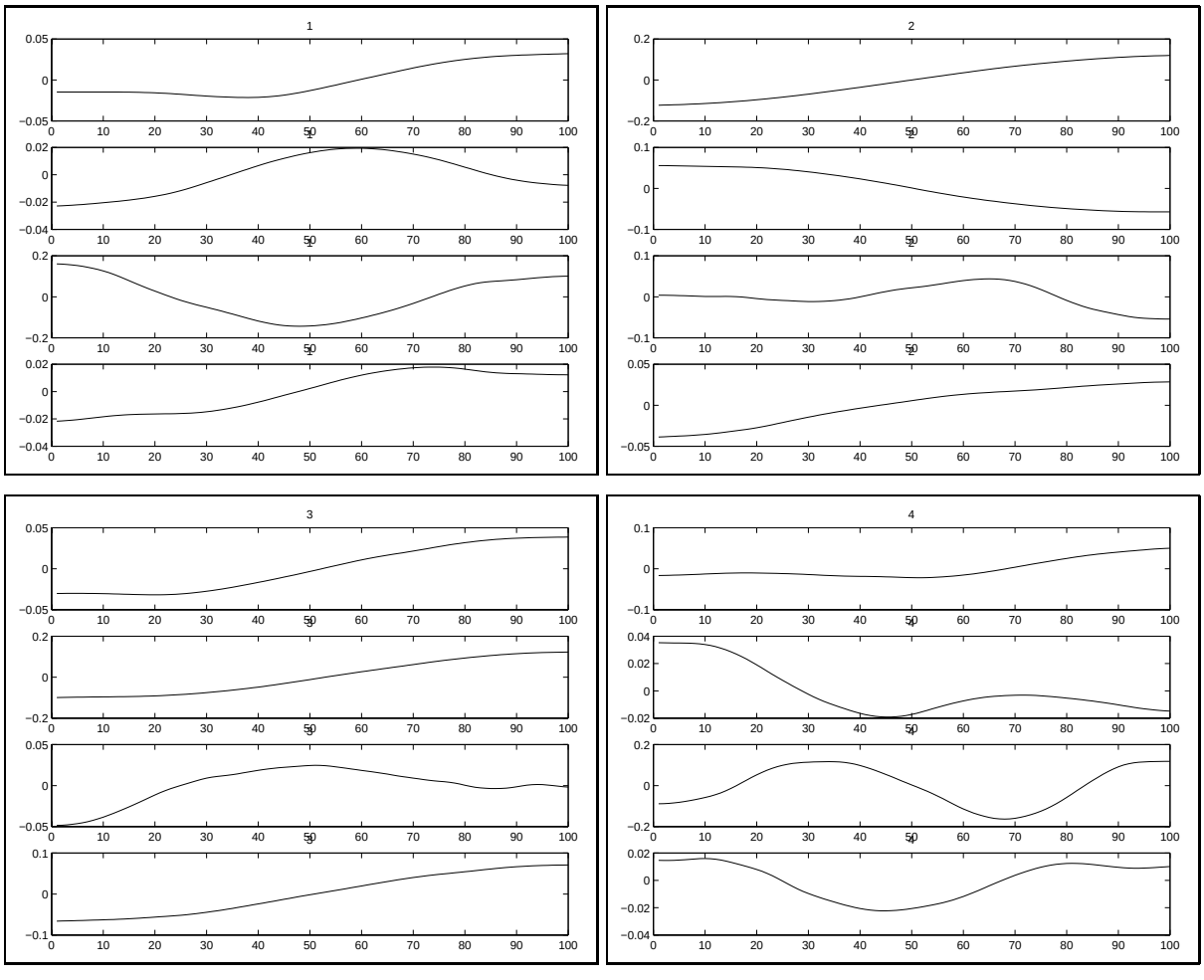

Fig. 7. The first 4 eigenvectors. Each figure shows one eigenvector. Each sub-plot shows the variation of an angle (y-axis) over time (x-axis) in each of the eigenvectors.

The number of eigenvectors chosen for further processing determines the accuracy and the computational overhead involved in deriving the movement primitives. More eigenvalues produce higher accuracy of the resulting primitive representation, but involve increased computational overhead in the derivation.

5.5 Segment Reconstruction

The vector **p** mentioned above can be used to reconstruct the original movement segment, as follows. **p** is essentially the projection of the movement segment onto the eigenvector space. To reconstruct, we need to expand the original vector in terms of this set of basis vectors. Formally, projecting back consists of:

$$\mathbf{s} = \mathbf{E_f p} \tag{5}$$

where vector **s** is in the same space as the original set of vectors. The resulting vectors can now be split into 4 components for the 4 DOF needed for reconstruction. We split the vector into 4 parts of length 100 each, thereby decomposing the movement in the 4 joint angles in time. This is the movement on a normalized time scale.

While encoding the movement in this format, we record the time at the beginning and the end of each segment to aid in the reconstruction of the movement. Expanding **p** for every segment gives us a set of time-normalized segments. These need to be resized temporally to obtain the original movement. This is done by cubic spline interpolation between data points from the KL expansion.

Because of the segmentation approach we used, the derived primitives, when executed in isolation, constitute strokes, movements of the arm from one pose to another [30]. High frequency components contribute increasingly in lower eigen-valued primitives. When combined through sequencing, the strokes generate a reconstruction of a given test movement.

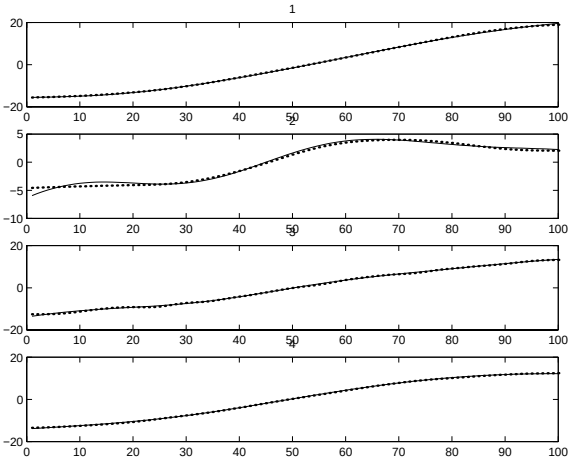

Fig. 8. Reconstruction of a segment. Each plot shows the variation of one angle (y-axis) over time (x-axis). The solid line shows the original movement, the dotted line the reconstruction.

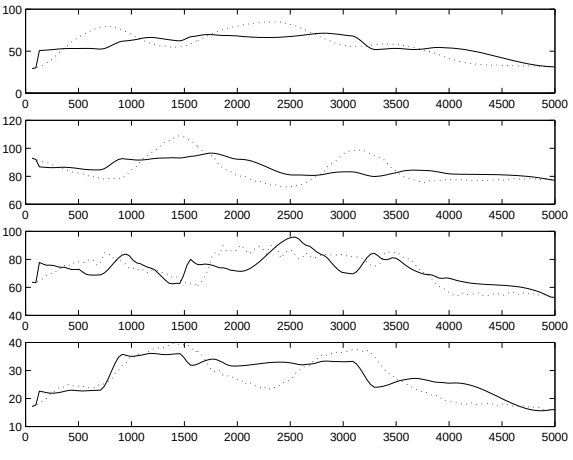

Fig. 9. Reconstruction of an entire movement composed of multiple segments. Each subplot shows one angle (y-axis) as a function of time (x-axis). The solid line shows the original movement, the dotted line the reconstruction.

6 Movement Controller Design

As described in Section 3.3, our imitation model involves a set of controllers which actuate the derived primitives. We explored the following method for generating such controllers by clustering motion segments in the reduced eigenvector space.

Each segment is projected onto a f-dimensional space by the subset of eigenvectors we chose to retain. We then cluster the points using K-means [8] into k clusters, where k is a predetermined constant. The algorithm is initialized with a set of k random estimates of cluster vectors. In every iteration, each point that is closer to one

of the vectors than others is grouped into a cluster belonging to that vector. The vector is then recalculated as the centroid of all the points belonging to the vector's cluster.

The choice of the number of clusters, k, is based on the error that can be tolerated in approximating a movement by a cluster. If the tolerable error is large, we can do well with few clusters. For more accuracy, we increase the number of clusters; this reduces the error but increases the computation time as well as the memory requirements of the clustering algorithm. Fortunately, clustering is performed in the training phase, while the primitives are being derived, not during their on-line use for real-time movement classification and imitation.

To evaluate the above clustering technique, we replaced each point in eigenvector (primitive) space with its corresponding cluster point. We then used those cluster points to reconstruct the training movements (i.e., the data used to create the clusters). The quantitative analysis of the reconstruction error is presented in Section 7.

We can view clustering as a naive means of controller parameterization. Thus, this analysis represents the worst-case scenario, providing an upper bound on the reconstruction error. Any more sophisticated controller design method will produce improved movement reconstruction. Next we discuss one such more sophisticated approach to controller design.

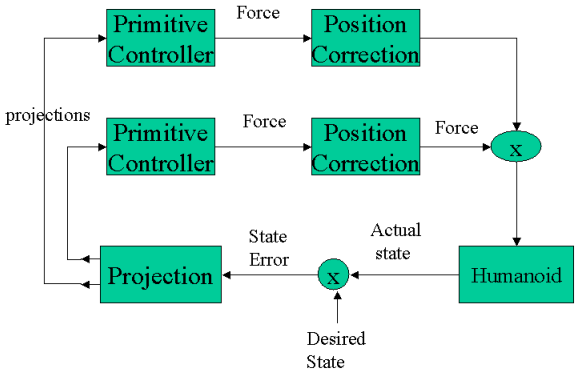

Fig. 10. A controller model. The Primitive Controller is an individual controller for each primitive. The Position Correction module corrects for changes in the location of the center of the stroke. The Projection Module projects the errors back onto the eigenmovement space.

A simple schematic representation of a primitives-based controller is shown in Figure 10. Desired inputs are fed into the control system and the error is calculated, then projected onto the primitives derived above. This process gives us certain projection coefficients which generate the desired control inputs for the primitive controllers.

As shown in the figure, each of the Primitive Controllers executes an individual primitive. Given the projection coefficients, the controller generates a force signal. The Position Correction modules correct for the difference in the center of the stroke and the consequent change in the dynamics of the humanoid. The force signals are then sent to the humanoid. The resulting angle measurements are compared with the desired angles. The error is transformed into a correction signal and sent to the Primitive Controllers.

This approach is one solution for the Actuation component of the Action layer of our imitation model. Our continuing research addresses the design and learning of such controllers. However, the evaluation of the primitives approach we present here has so far been performed with a simpler Actuation component, using PD servos on the humanoid simulation. The results are presented in Section 8. The next section presents an analytical evaluation of the complete method, including the accuracy of primitives-based movement reconstruction, with and without the use of clustering.

7 Results and Evaluation

We chose mean squared error (MSE), one of the most commonly used error metrics for linear systems, to evaluate the performance of our encoding and reconstruction of movement. MSE is attractive for the following reasons:

1. It penalizes large deviations from the original more severely than smaller ones.
2. It is additive. Error in one part of the reconstruction will not cancel the error in another part.
3. It is mathematically tractable.

Representing segments in terms of eigenvectors allows us to quantify the error in the subsequent reconstruction. The squared error between the reconstructed vector $\mathbf{s}_r$ and the original vector $\mathbf{s}$ is given by:

$$\epsilon = (\mathbf{s} - \mathbf{s_r})^T (\mathbf{s} - \mathbf{s_r}) \tag{6}$$

If we tolerate a root mean squared error of θ degrees at each sample point in $\mathbf{s}$, the required bound on the expectation of ϵ, i.e., MSE, is given by:

$$E(\epsilon) \leq N\theta^2 \tag{7}$$

The Karhunen-Louve (KL) expansion of the reconstructed vector in terms of the eigenvectors is:

$$\mathbf{s_r} = \sum_{i=1}^{f} p_{r,i}\mathbf{e_i} \tag{8}$$

where r is the cluster that is used for the reconstruction and $p_{r,i}$ are the coordinates of the cluster points.

The difference between the original and the reconstructed vectors is:

$$\mathbf{s} - \mathbf{s_r} = \sum_{i=1}^{f}(p_i - p_{r,i})\mathbf{e_i} + \sum_{i=f+1}^{N} p_i\mathbf{e_i} \tag{9}$$

The first term in the computed reconstruction error is the *clustering error*, while the second term is the *projection error*.

Our formulation of the clustering error is based on approximating any segment by its closest cluster point. A torque strategy can be developed for each cluster point. If we use the torque strategy corresponding to the nearest cluster, our error in reconstructing the movement would be lower, bounded by the error as formulated above. If, instead, we make use of a local approximation of the movement-torque map around the cluster point, this error can be reduced further.

Since the eigenvectors are a set of orthonormal basis vectors, the MSE is a sum of the MSE of each projection along the individual eigenvectors. The MSE of individual projections along each of the dropped eigenvectors is:

$$E(\epsilon)_e = \sum_{i=f+1}^{N} \lambda_i \tag{10}$$

where the eigenvectors $f+1$ through N were dropped. The error in projection is a function of the number of eigenvectors kept for reconstruction. Table 1 compares the projection error for different numbers of eigenvectors for the segmentation algorithms SEG-1 and SEG-2.

We used thirty eigenvectors ($f = 30$) in our projection of the input vector.

No. of eigenvectors used	MSE_p using SEG-1	MSE_p using SEG-2
10	$1.22 * 10^7$	$6.47 * 10^7$
20	$1.05 * 10^5$	$1.48 * 10^6$
30	$2.82 * 10^3$	$7.64 * 10^4$
40	$1.92 * 10^2$	$5.39 * 10^3$

Table 1. A comparison of the MSE in projection using SEG-1 and SEG-2 segmentation routines.

The clustering error is orthogonal to the projection error and is given by:

$$E(\epsilon)_c = E\left(\sum_{i=1}^{f}(a_i - p_{r,i})^2\right) \tag{11}$$

where f is the number of dimensions in which clustering is done, a_i is projection of the data point along the i^{th} eigenvector, and $p_{r,i}$ is the projection of the cluster vector along the same. The above is also the average squared distance from the representative vector when clustering is done. Table 2 shows a tabulation of the MSE_c introduced by clustering as a function of the number of clusters used, comparing the two segmentation algorithms we used, SEG-1 and SEG-2.

No. of clusters	MSE_c using SEG-1	MSE_c using SEG-2
10	$1.45 * 10^9$	$6.12 * 10^5$
20	$3.05 * 10^5$	$3.52 * 10^5$
30	$2.34 * 10^5$	$2.84 * 10^5$
100		$9.63 * 10^4$
170		$7.54 * 10^4$

Table 2. A comparison of the MSE in clustering using the SEG-1 and SEG-2 segmentation routines.

Since the error in clustering is orthogonal to the error in dropping smaller eigenvalues, the total error introduced by the approximation steps is given by:

$$E(\epsilon) = E(\epsilon)_p + E(\epsilon)_c \tag{12}$$

A tabulation of this total error as a function of the number of clusters used is given in Table 3. The total error is calculated assuming that 30 eigenvectors were used in the KL expansion.

No. of clusters	Total Error Using SEG-1	θ (degrees) Using SEG-1	Total Error Using SEG-2	θ (degrees) Using SEG-2
10	$1.45 * 10^6$	60.28	$6.88 * 10^5$	41.42
20	$3.05 * 10^5$	26.68	$4.28 * 10^5$	32.71
30	$2.34 * 10^5$	24.43	$3.50 * 10^5$	29.50
100			$1.633 * 10^5$	20.22
170			$1.514 * 10^5$	19.42

Table 3. Angular error corresponding to reconstruction with 30 eigenvectors using SEG-1 and SEG-2 segmentation routines.

As can be seen, SEG-2 is a more complete segmentation algorithm than SEG-1, which partitions the movement data into complete non-overlapping segments more suitable for reconstruction. While using SEG-2, the magnitude of the eigenvectors does not decline as fast as in SEG-1. As a result of this, the number of eigenvectors required to reconstruct the original segment is larger, as shown in Tables 2 and 3.

The majority of the training movements segmented into 2, 3, or 4 segments. Figure 8 shows an example of a reconstructed movement, a reproduction of one segment from the input data. Figure 9 shows a reconstruction of an entire movement composed of multiple segments. Each sub-plot shows one angle as a function of time. In both figures, solid lines show original movements, dotted lines the reconstruction.

To compare the performance of the algorithm in reconstructing data from the training set with that in reconstructing movement outside the training set, we split our movement data into two sets. Then, 75% of the data were used as a training set, and the remaining 25% as a test set, i.e., as novel movements. The results show that reconstruction is marginally better with the training set (see Figure 11). The median RMS reconstruction error is 19.4 degrees for the training data, and 22.27 degrees for the test data. Both reconstructions, observed by the naked eye, qualitatively appear to have very high fidelity.

The choice of the model parameters directly affects the error in the movement representation and thus reconstruction. By altering those parameters, we can guarantee a higher accuracy in representation or achieve greater generalization. For example, higher accuracy can be obtained by either increasing the number of eigenvectors in the representation or by increasing the number of cluster points. This choice involves a necessary tradeoff between the processing time, memory, and desired accuracy.

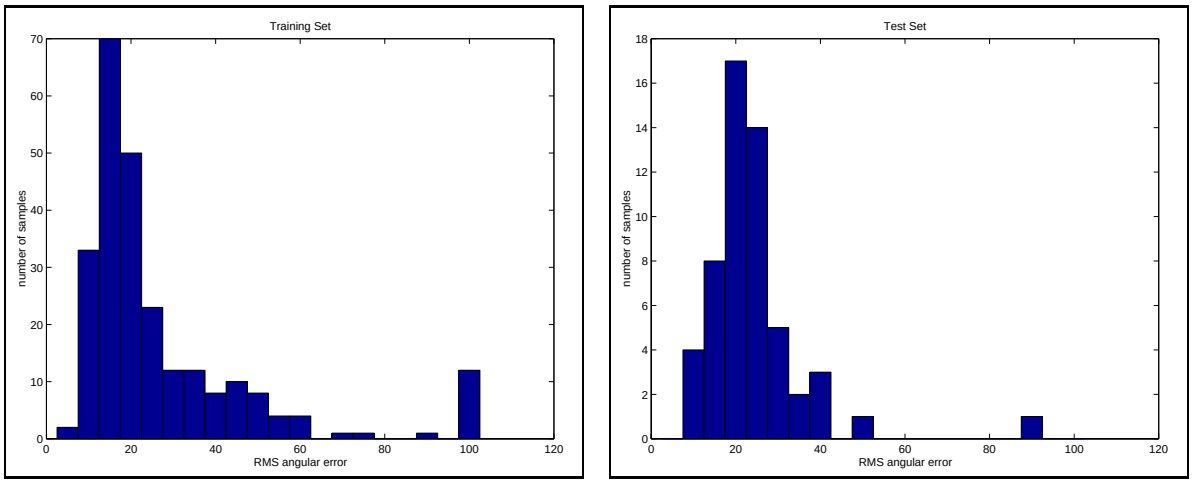

Fig. 11. A comparison of the histograms of RMS angular errors in the reconstruction of movements in the training (left plot) and test (right plot) data sets.

8 Humanoid Simulation and Validation

To demonstrate an application of the derived primitives, we used them to control a humanoid simulation with full dynamics. Here we describe the simulator test-bed, the controller implementation, and finally the results of the validation.

8.1 The Humanoid Simulation Test-Bed

Our validation of the approach was performed on Adonis [22, 21], a rigid-body simulation of a human torso, with static legs (see Figure 12). Adonis consists of eight rigid links connected in a tree structure by a combination of hinge, gimbal, and ball joints. The simulator has a total of 20 DOF. Each arm has 7 DOF, but movement of one or both arms generates internal torques in the rest of the DOFs. The static ground alleviates the need for explicit balance control.

Mass and moment of inertia information is generated from the geometry of body parts and human body density estimates. Equations of motion are calculated using a commercial solver, SD/Fast [13]. The simulation acts under gravity and accepts other external forces from the environment. Collision detection, with the body itself and with the environment, is implemented on Adonis, but was not used in the experiments presented here, as it was not relevant; the training and test movements involved no collisions or near-collisions.

8.2 Motor Control in the Simulation Test-Bed

We used joint-space PD servos to actuate the humanoid in order to execute the derived primitives, after converting the movement segments into a quaternion representation. Specifically, the segments were presented as a sequence of target joint angle configurations. If the set of desired points falls on a smooth trajectory, the resulting motion has a natural appearance. Based on the target joint angle configuration, the torques for all joints on the arm were calculated as a function of the angular position and velocity errors by using a PD servo controller:

$$\tau = k(\theta_{desired} - \theta_{actual}) + k_d(\dot{\theta}_{desired} - \dot{\theta}_{actual}) \tag{13}$$

where k is the stiffness of the joint, k_d the damping coefficient, $\theta_{desired}$, $\dot{\theta}_{desired}$ are the desired angles and velocities for the joints, and θ_{actual}, $\dot{\theta}_{actual}$ are the actual joint angles and velocities.

In this simple validation, Adonis is programmed to continually reach target points in its joint space. This can be thought of as setting a region about the target point into which the trajectory must fall before the next target is

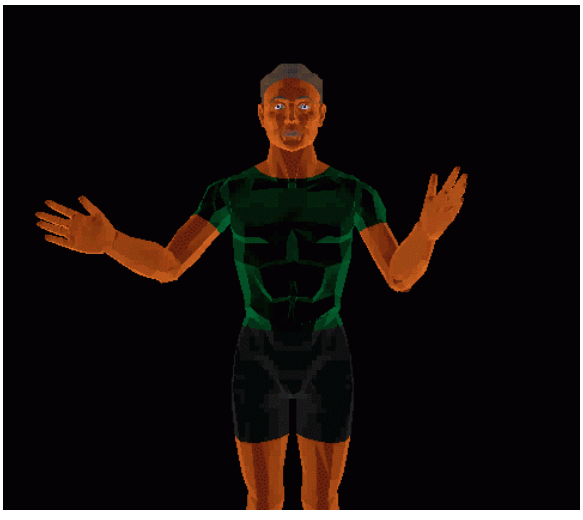

Fig. 12. The graphical display of Adonis, the dynamic humanoid simulation we used for validation.

tracked. This approach has the potential of over-constraining the trajectory. However, its purpose here is merely to validate the reconstruction of the primitives and to visualize them. As discussed in Section 6, our model mandates the use of motor controllers associated with each primitive [19, 16], an area of research we are currently pursuing.

9 Summary and Continuing Research

We have presented a method that can be used to derive a set of perceptuo-motor primitives directly from movement data. The primitives can be used as a lower-dimensional space for representing movement, as proposed in our imitation model. By projecting observed movement onto a lower-dimensional primitives-based space, we can facilitate perception in a top-down manner, as well as classification and imitation. Thus we only need to store a small number of control strategies and use those as a basis for sequencing. In addition to sequencing primitives, our representation also allows them to be concurrently executed, since the eigenvectors form superimposable basis functions.

Briefly, the derivation process consists of the following steps. Movement data in the form of 3D locations of the arm joints are converted into an Euler representation of joint angles using inverse kinematics, and filtered to facilitate segmentation. Next, one of the two segmentation algorithms we proposed was applied. Principal components analysis was performed on the resulting segments to obtain the "eigenmovements" or primitives. The K-means algorithm was used to cluster the points in the space of the projections of the vectors along the eigenvectors corresponding to the highest eigenvalues to obtain commonly used movements. Reconstruction was performed to retrieve the original movements as an evolution of Euler angles in time from a representation in terms of primitives. A MSE-based error metric was used to evaluate the resulting reconstruction.

The described research is one of several of our projects aimed at primitives-based motor control and imitation, described in more detail in [19, 16]. The areas of direct application of our work are in robotics, physics-based animation, and Human Computer Interface (HCI). In all of these areas, the need to control and interact with complex humanoid systems, whether they be physical robots or realistic simulations, involves managing complex motor control, where the number of interdependent variables makes designing control strategies formidable. By using human movement data, we can find a set of relationships between the different degrees of freedom, as our eigenvectors do, to reduce the dimensionality of the problem. This can further be used for controller design, as well as more general structuring of perception, motor control, and learning.

The presented method for deriving primitives is the basis for our work on primitives-based motor control and imitation [19, 16]. The notion of motor primitives is gaining popularity in both the motor control and the animation communities. For example, [30] suggests a learning mechanism where a few movement primitives compete for a demonstrated behavior. He uses an array of primitives for robot control, each of which has an associated controller. Learning is performed by adjusting both the controller and the primitives. Our method for deriving primitives could be used either in place of the learning algorithm or to help bootstrap it.

Our continuing research focuses on methods for generating controllers for the derived primitives. We are also working on the policies needed to modify these controllers for changes in duration and couplings with other controllers that are simultaneously active. This is an important part of our model, which involves simultaneous execution as well as sequencing of the primitives for both movement perception and generation [19]. Finally, we are exploring alternative segmentation methods, in particular a means of using existing primitives as top-down models.

Acknowledgments

The research described here was supported in part by the NSF Career Grant IRI-9624237 to M. Matarić, by the DARPA MARS Program Grant DABT63-99-1-0015, by the National Science Foundation Infrastructure Grant CDA-9512448, and by the USC All-University Predoctoral Fellowship to Chad Jenkins. The data used for this analysis were gathered by M. Matarić and M. Pomplun in a joint interdisciplinary project conducted at the National Institutes of Health Resource for the Study of Neural Models of Behavior, at the University of Rochester. The humanoid simulation was obtained from Jessica Hodgins.

References

1. R. O. Ambrose, H. Aldridge, R. S. Askew, R. R. Burridge, W. Bluethmann, M. Diftler, C. Lovchik, D. Magruder, and F. Rehnmark. Impedance control: An approach to manipulation. *IEEE Intelligent Systems*, 15(4):57–63, 2000.
2. M. Arbib. Schema theory. In S. Shapiro, editor, *The Encyclophedia of Aritificial Intelligence, 2nd Edition*, pages 1427–1443. Wiley-Interscience, 1992.
3. R. C. Arkin. Motor schema based navigation for a mobile robot: An approach to programming by behavior. In *Proceedings of IEEE Intl. Conf. on Robotics and Automation*, pages 264–271, Raleigh, NC, April 1987.
4. R. C. Arkin. *Behavior-Based Robotics*. MIT Press, Cambridge, MA, USA, 1998.
5. 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.
6. M. Brand. Understanding manipulation in video. In 2^{nd} *International Conference on Face and Gesture Recognition*, pages 94–99, Killington, VT, 1996.
7. 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.
8. A. D.Gordon. *Classification*. Chapman and Hall, 1999.
9. T. Flash and N. Hogan. The coordination of arm movements: An experimentally confirmed mathematical model. *Journal of Neuroscience*, 5:1688–1703, 1985.
10. G. L. Gottlieb, Q. Song, Hong D-A, G. L. Almeida, and D. Corcos. Coordinating movement at two joints: a principle of linear covariance. *Journal of Neurophysiology*, 75:1760–1764, 1996.
11. S. Hashimoto, S. Narita, H. Kasahara, K. Shirai, T. Kobayashi, A. Takanishi, S. Sugano, J. Yamaguchi, H. Sawada, H. Takanobu, K. Shibuya, T. Morita, T. Kurata, N. Onoe, K. Ouchi, T. Noguchi, Y. Niwa, S. Nagayama, H. Tabayashi, I. Matsui, M. Obata, H. Matsuzaki, A. Murasugi, T. Kobayashi, S. Haruyama, T. Okada, Y. Hidaki, Y. Taguchi, K. Hoashi, E. Morikawa, Y. Iwano, D. Araki, J. Suzuki, M. Yokoyama, I. Dawa, D. Nishino, S. Inoue, T. Hirano, E. Soga, S. Gen, T. Yanada, K. Kato, S. Sakamoto, Y. Ishii, S. Matsuo, Y. Yamamoto, K. Sato, T. Hagiwara, T. Ueda, N. Honda, K. Hashimoto, T. Hanamoto, S. Kayaba, T. Kojima, H. Iwata, H. Kubodera, R. Matsuki, T. Nakajima, K. Nitto, D. Yamamoto, Y. Kamizaki, S. Nagaike, Y. Kunitake, and S. Morita. Humanoid robots in waseda university – hadaly-2 and wabian. In *Proc. of First IEEE-RAS International Conference on Humanoid Robots*, Cambridge, MA, USA, 2000.
12. N. Hogan. An organizing principle for a class of voluntary movements. *Journal of Neuroscience*, 4:2745–2754, 1984.
13. M. Hollars, D. Rosenthal, and M. Sherman. Sd/fast user's manual. Technical report, Symbolic Dynamics, Inc., 1991.
14. 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.
15. H. Inoue, S. Tachi, K. Tanie, K.Yokoi, S. Hirai, H. Hirukawa, K. Hirai, S. Nakayama, K. Sawada, T. Nishiyama, O. Mikiand T. Itoko, H. Inaba, and M. Sudo. Hrp: Humanoid robotics project of miti. In *Proc. of First IEEE-RAS International Conference on Humanoid Robots*, Cambridge, MA, USA, 2000.
16. O. C. Jenkins, M. J. Matarić, and S. Weber. Primitive-based movement classification for humanoid imitation. In *Proceedings, First IEEE-RAS International Conference on Humanoid Robotics (Humanoids-2000)*, Cambridge, MA, USA, September 2000.
17. E. Kreighbaum and K. M. Barthels. *Biomechanics: A Qualitative Approach for Studying Human Movement*. Burgess Publishing Company, Minneapolis, Minnesota, 1985.
18. 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.

19. M. J. Matarić. Sensory-motor primitives as a basis for imitation: Linking perception to action and biology to robotics (in press). In C. Nehaniv & K. Dautenhahn, editor, *Imitation in Animals and Artifacts*. The MIT Press, 2001.
20. M. J. Matarić and M. J. Marjanović. Synthesizing complex behaviors by composing simple primitives, self organization and life: From simple rules to global complexity. In *European Conference on Artificial Life (ECAL-93)*, pages 698–707, Brussels, Belgium, May 1993.
21. 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.
22. 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):23–43, 1999.
23. Maja J Matarić. Getting humanoids to move and imitate. *IEEE Intelligent Systems*, pages 18–24, July/August 2000.
24. 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.
25. F. A. Mussa-Ivaldi, S. F. Giszter, and E. Bizzi. Convergent force fields organized in the frog's spinal cord. *Journal of Neuroscience.*, 12(2):467–491, 1993.
26. H. E. Pashler. *The Psychology of Attention*. The MIT Press, 1999.
27. D. Pierce and B. Kuipers. Map learning with uninterpreted sensors and effectors. *Artificial Intelligence Journal*, 92:169–229, 1997.
28. M. Pomplun and M. J. Matarić. Evaluation metrics and results of human arm movement imitation. In *Proceedings, First IEEE-RAS International Conference on Humanoid Robotics (Humanoids-2000)*, MIT, Cambridge, MA, September 2000. Also IRIS Technical Report IRIS-00-384.
29. 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.
30. S. Schaal. Is imitation learning the route to humanoid robots. *Trends in Cognitive Sciences*, 3(6):233–242, 1999.
31. S. Schaal, S. Kotosaka, and D. Sternad. Programmable pattern generators. In *Proceedings, 3rd International Conference on Computational Intelligence in Neuroscience*, pages 48–51, Research Triangle Park, NC, 1998.
32. P. S. G. Stein. *Neurons, Networks, and Motor Behavior*. The MIT Press, Cambridge, Massachusetts, 1997.
33. K. A. Thoroughman and R. Shadmehr. Learning of action through combination of motor primitives. *Nature*, 407:742–747, 2000.
34. M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.
35. Y. Uno, M. Kawato, and R. Suzuki. Formation and control of optimal trajectory in human multijoint arm movement. *Biological Cybernetics*, 61(2):89–101, 1989.
36. P. Viviani and C. Terzuolo. Trajectory determines movement dynamics. *Neuroscience*, 7:431–437, 1982.
37. Y. Wada, Y. Koike, E. Vatikiotis-Bateson, and M. Kawato. A computational theory for movement pattern recognition based on optimal movement pattern generation. *Biological Cybernetics*, 73:15–25, 1995.

Ajo Fod received his B.S. in Mechanical Engineering from the Indian Institute of Technology - Madras (IIT) in 1998. He received two M.S. degrees, in Operations Research and Computer Science, from the University of Southern California, Los Angeles (USC) in 1999 and 2001, respectively. His current research interests include activity recognition using planar lasers and frameworks for representing such activity.

Maja Matarić is an associate professor in the Computer Science Department and the Neuroscience Program at the University of Southern California, the Director of the USC Robotics Research Labs and an Associate Director of IRIS (Institute for Robotics and Intelligent Systems). She joined USC in September 1997, after two and a half years as an assistant professor in the Computer Science Department and the Volen Center for Complex Systems at Brandeis University. She received her PhD in Computer Science and Artificial Intelligence from MIT in 1994, her MS in Computer Science from MIT in 1990, and her BS in Computer Science from the University of Kansas in 1987. She is a recipient of the IEEE Robotics and Automation Society Early Career Award, the NSF Career Award, and the MIT TR100 Innovation Award. She has worked at NASA's Jet Propulsion Lab, the Free University of Brussels AI Lab, LEGO Cambridge Research Labs, GTE Research Labs, the Swedish Institute of Computer Science, and ATR Human Information Processing Labs. Her research studies coordination and learning in robot teams and humanoids. More information about it is found at: http://robotics.usc.edu/∼maja

Chad Jenkins is a Ph.D. student in the Computer Science Department and the Robotics Research Labs at the University of Southern California. He received his BS in computer science and mathematics from Alma College and his MS in computer science from The Georgia Institute of Technology. He is currently working on his thesis pertaining to the generation of perceptual-motor primitives from motion capture data. His research interests include control for humanoid robots and characters, computer animation, activity modeling and sensing, and interactive environments. His email address is cjenkins@usc.edu and contact information is available at http://robotics.usc.edu/∼cjenkins