Automated Derivation of Primitives for Movement Classification

Ajo Fod, Maja J Matarić, and Odest Chadwicke Jenkins

University of Southern California, Los Angeles CA, USA,
afod|mataric|cjenkins@pollux.edu,
WWW home page: http://www-robotics.usc.edu/~agents/imitation.html

Abstract. We present 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, so it automatically classifies 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 data from a psychophysical experiment on human imitation to derive a set of primitives, and then used those primitives as a basis for superposition and sequencing to reconstruct the original movements. We performed principal component analysis on segments from these data, resulting in a set of basis vectors. Next we clustered in the space of projections of segments onto the eigenvectors, to obtain a set of frequently used movements. To validate the approach experimentally, we used the movement obtained by expanding the cluster points in terms of the eigenvectors as a sequence of via points to control a humanoid dynamic simulation. We also developed an error metric to measure the effectiveness of the process.

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 [19, 14, 17], inspired by neuroscience evidence for motor *primitives* [5] and *mirror neurons* [13], structures that directly link the visual and motor systems in the context of complete generalized movements. In our model, the ability to mimic and imitate is based on this 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].

Biological primitives are structures that organize the underlying mechanisms of movement, including spinal fields [5] and central pattern generators [31]. 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 [11, 9, 34], 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.

Determining an effective basis set of primitives is thus a difficult problem. Our previous work has explored hand-coded primitives [22, 14]. 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 "eigen-movements" or primitives. The eigenvectors corresponding to a few of the highest eigenvalues provide us with a basis set for a subspace. The projection of the segment vector onto this subspace contains most of the information about the original segment. By clustering in this subspace we obtain a set of points that correspond to a set of frequently used movements which can be used to calibrate controllers. To evaluate the method of movement encoding in terms of eigenmovement primitives, and the subsequent reconstruction of the original movements, we calculated the mean square deviation of the reconstructed movement. Finally, we demonstrate the movements on a dynamic humanoid simulation.

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 introduces the evaluation metric and the performance of the movement reconstruction. Section 7 describes the validation of the derived primitives on a humanoid simulation. In Section 8, the results are discussed, and Section 9 summarizes the paper.

2 Motivation and Related Work

Movement primitives are behaviors that accomplish complete goal-directed actions [20, 18, 19]. In our model, a relatively small set of such behaviors is used to structure movement as well as bias and classify movement perception. This results in a set of perceptuo-motor primitives, which unify visual perception and motor output. Primitives are a way of compactly describing information about a movement, and in our model, the visual and the motor descriptions are directly mapped. While our model is inspired by neuroscience evidence, the specific methods we describe for deriving the primitives are not intended to model specific biological processes.

The inspiration for primitives comes from neuroscience evidence. 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] perform 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 call Convergent Force Fields (CFFs). 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 [2, 1, 3] and our own work on basis behaviors [20, 18]. Recently, Sanger[28] demonstrated such motor synergies by applying principle component analysis (PCA) to analyze the trajectory 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 robotics, Pierce and Kuipers [26] proposed a method of abstracting control of a system using PCA to determine motor features (primitives) that are mapped onto sensory features.

Other studies and theories about motor primitives [23, 24, 13] suggest that they are viable means for encoding humanoid movement. [32] provides neuroscience evidence in support of motor primitives. He argues for a localized sensorimotor map from position, velocity, and acceleration to the torques at the joints. In contrast, our method gives us a fixed set of trajectories for which controllers can be generated. We also suggest a method to determine torques for a linear superposition of such these trajectories. [30, 29] 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 very 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.

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. Perceptuo-motor primitives 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. 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 to "gist" a video of action sequences by reasoning about changes in motions and contact relations between participants in an action. Usually only the agent (typically a part of an arm) and any objects under its control are segmented. Our other work [14] 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" [33]. There, PCA was applied to a set of static images of faces, while in this work we address classification and encoding of time-extended multi-joint movement data.

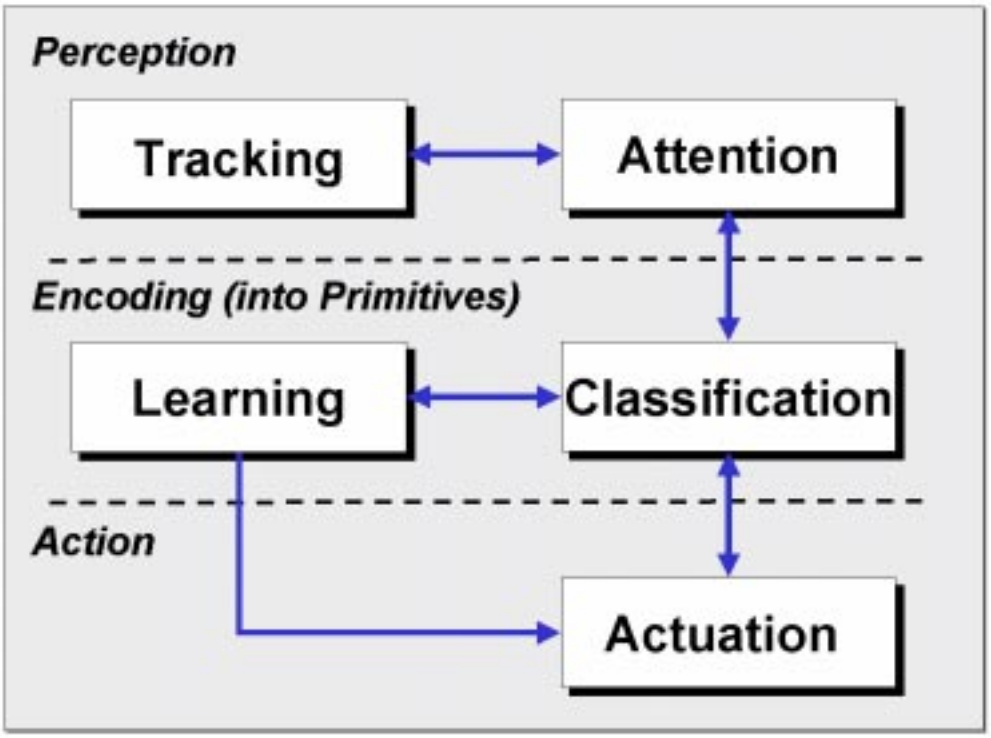

Fig. 1. *Our imitation model.*

3 The Imitation Model

Next, we describe in more detail our model for imitation using perceptuo-motor primitives. 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 first layer, Perception, 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.

In this work, 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) [15]. 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 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 [25].

3.2 Classification and Learning

The Encoding layer of the model 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. Thus, the classifier itself is a means for segmentating time based on the presence of primitives in the observed motion.

We consider several issues regarding motion representations at the Encoding layer in [14]. The first is how to represent motion segments such that they are amenable to classification and learning. The second is that each motion segment requires conversion into the chosen representation and may need to provide a means of segmentation. Additionally, the movement data can have invariances applied to it to account for variations in position and scale. The final issue is that the chosen representation and conversion must be consistent for all components used in this layer.

The Encoding layer provides two outputs to the Action layer: 1) the list of time 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 certain primitive is active.

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. Our previous work has also addressed the issue of dissimilar kinematics between performers and imitators [4].

We address three components in the model: Attention, Learning, and Classification. Attention is implemented in the form of a method for segmenting motion in time. Learning is addressed only at the level of acquiring 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 reconstruction of the original observed movement.

4 Human Movement Data

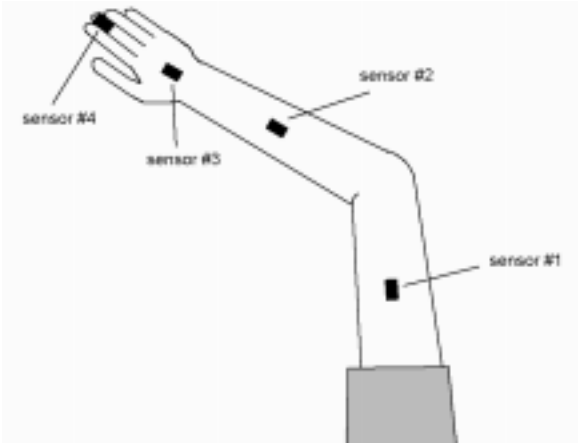

Fig. 2. Location of FastTrack sensors on the subject's arm.

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] [27]. Each stimulus was a 3 to 5-second long sequence of arm movements, presented against a black

[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.

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 FastTrak motion tracking system. Position sensors, about $2 \times 1 \times 1$ 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. on the phalanx of the middle finger.

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 experiment and the results are described in [27].

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 set of which represent almost all the variance in the data. We then applied K-means clustering to group the projections of the segments in order to obtain clusters that represent often used movements. The details of each of these methods are given in the following sections. Figure 3 summarizes the steps of the approach.

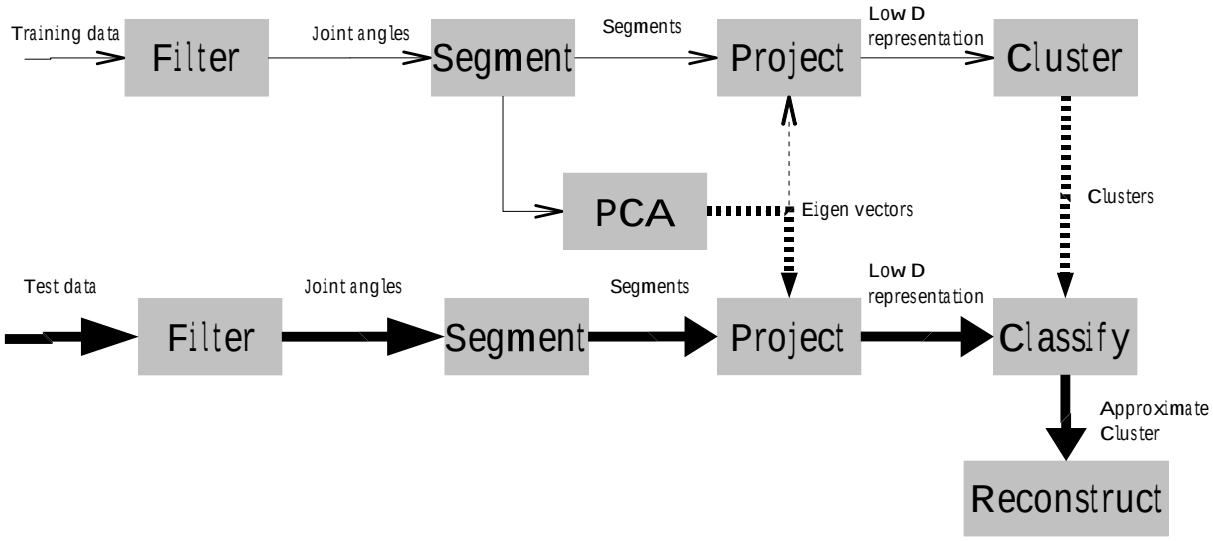

Fig. 3. Components of our system. Thin lines represent the flow of data in the training phase. Dark lines represent the flow of data in the testing phase. Dotted lines represent the use of previously computed data.

5.1 Inverse Kinematics

The raw movement data we used was 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 the 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 lie only 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 FastTrack system reads the 3D coordinates of all sensors every 34ms. The resulting data contains two distinct types of noise that had to be filtered out. 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. 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 that common features across segments can be extracted. Furthermore, in the context of movement representation and reconstruction, the segmentation algorithm needs to have the following properties:

1. *Consistency*: Segmentation needs to be consistent, producing identical segments in different repetitions of the action. For this, we need to define a set of features which could be used to detect the transition from one segment to another.
2. *Completeness*: Ideally, segments need to partition the input space completely. In other words, there would be no overlap between segments and, the set of segments would contain enough information to reconstruct the original movement perfectly.

A great deal of previous work has dealt with segmentation of time-series data. [6, 7] present an event-based method for continuous video segmentation; segments are introduced when a certain feature or contact configuration changes. [37] introduced a via-point method of segmenting handwriting, speech, and other temporal functions. [27] use a segmentation scheme based on the RMS value of velocity in different joints. To reduce the distance between two movements that could potentially be similar, they propose a modified scheme that minimizes the distance between two movement to chose the best segmentation. We consider a comparatively simpler approach to segmentation that has proven sufficient for our problem domain.

The change in joint angles over time can be plotted in a 4D space as a parametric curve. 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.

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

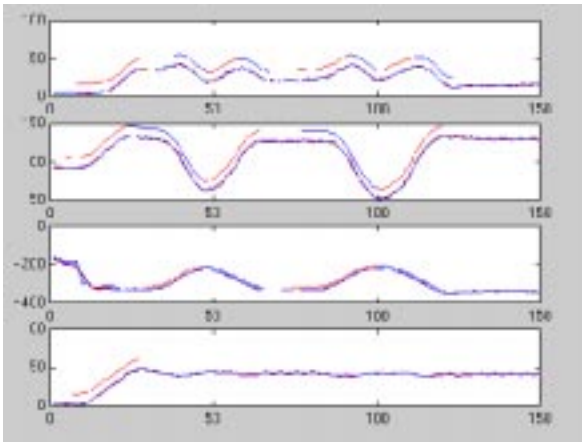

Fig. 4. Segmenting using zero-velocity crossings. Variables are plotted along the y-axis against time on the x-axis. The four sub-plots are of the four angles that we analyzed. The curved lines above the movement plots represent individual segments.

DOF occurs between ZVCs. If the movement is significant, 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 shown above the continuous curves indicated the instances when such segments are recorded.

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 continjous 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. For example, 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.

In order to produce a 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. Our 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 6 we compare the performance of the two segmentation algorithms.

5.4 Principal Component Analysis

The segmentation algorithms above convert the input data into a list of segments of varying lengths. In order to perform PCA on the segments, we first convert the movement vector form. For each DOF, the movement within a

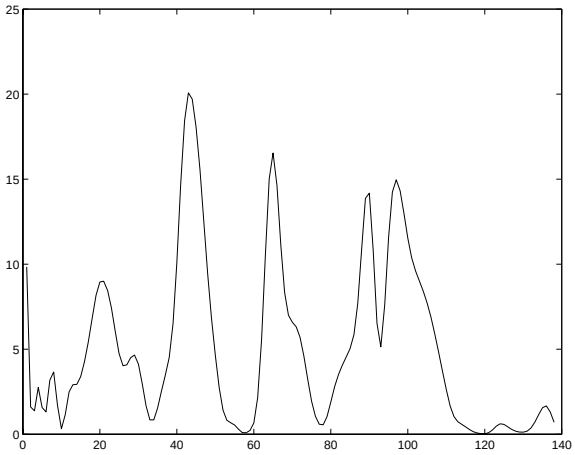

Fig. 5. A plot of the variation of the sum of squares of velocities (y-axis) as a function of the time (x-axis).

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 **s** of 400 elements. This is effectively a 400D representation of the movement within each segment.

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.

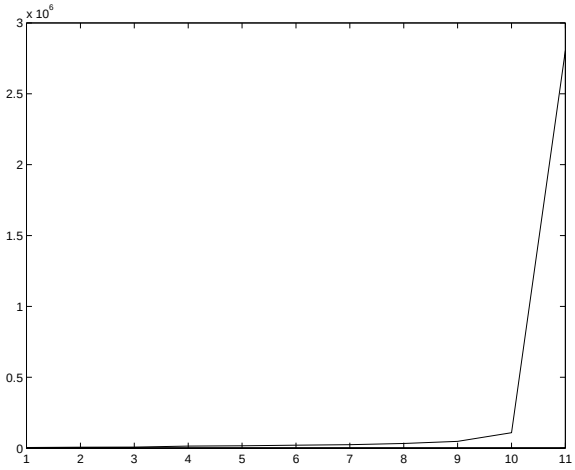

Fig. 6. The the magnitude of the last 11 eigenvalues.

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 eignvectors obtained by

sorting the 400 principal components of $\mathbf{K}$ by increasing magnitude. Thus, we can describe the vector $\mathbf{s}$ in terms of the projection of the vector along these f eigenvectors $\mathbf{E_f}$, in other words, an f-dimensional vector $\mathbf{p}$:

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

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

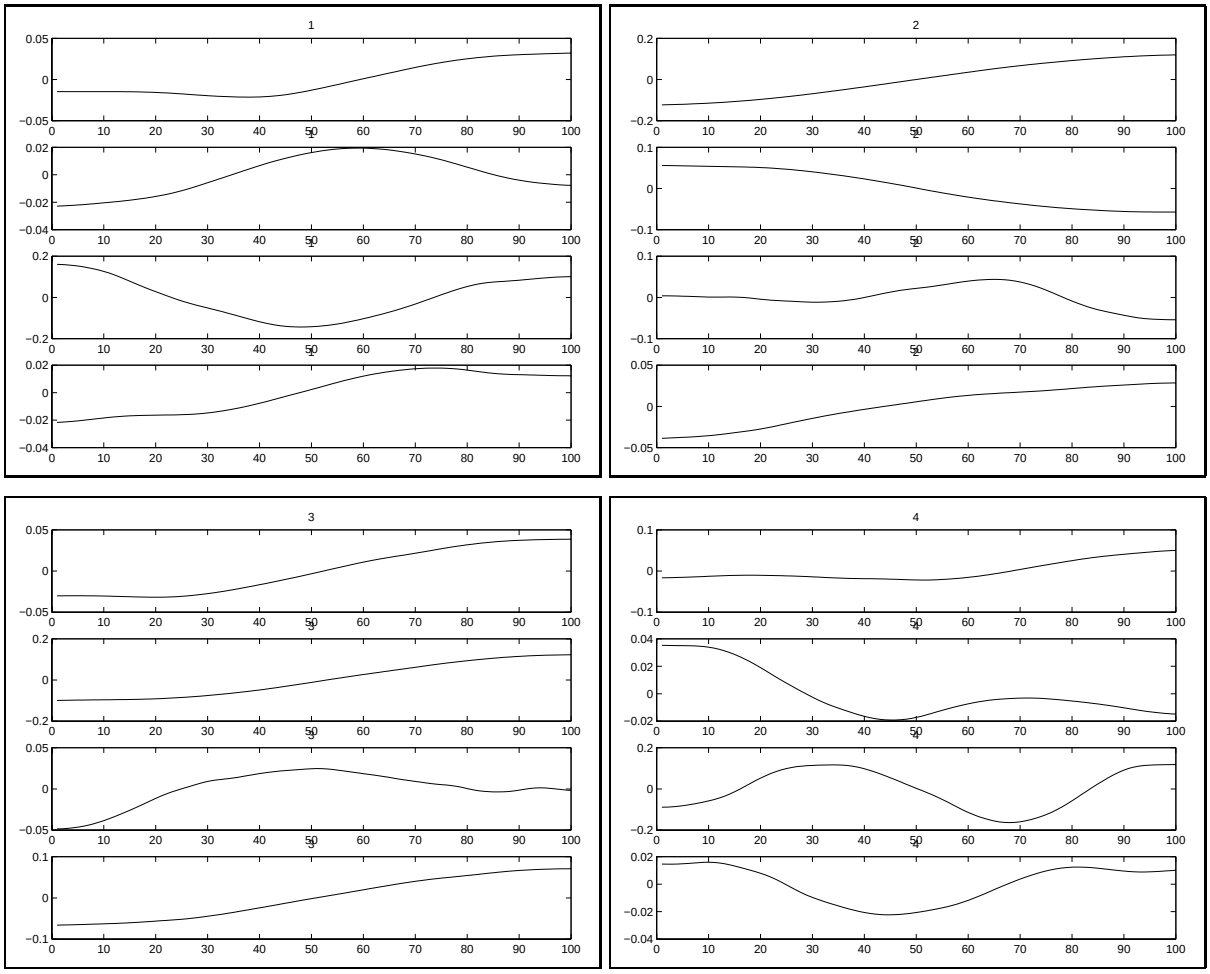

Fig. 7. The first 4 eigenvectors. Each figure shows one eigenvector. Each sub-plot shows the variation of an angle (y-axis) over time (x-axis) in each of the eigenvectors.

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 Clustering

Linearization of the pose-torque transformation [35] is desirable in the design of controllers for complex dynamical systems. In the following we describe one approach in determining a choice of linearization points based on frequency of use.

Each segment is projected on to 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 and the local linearity of the pose-torque map. 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, i.e., in the process of deriving the primitives, and not on-line, while the imitation system is executed in real-time.

5.6 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 help 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.

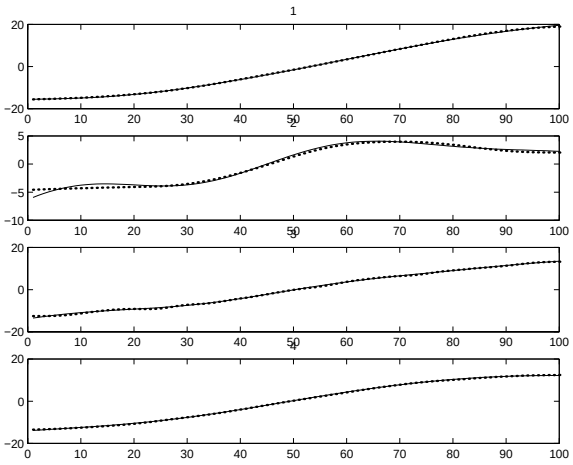

Fig. 8. Reconstruction of a segment. Each plot shows the variation of one angle over time. The solid line shows the original movement, the dotted line the reconstruction.

6 Results and Evaluation

We chose Mean Squred 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 **s** is given by:

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

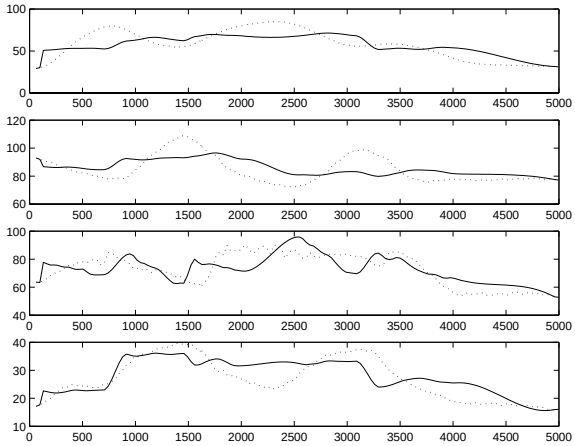

Fig. 9. Reconstruction of an entire movement composed of multiple segments. Each subplot shows one angle as a function of time. The solid line shows the original movement, the dotted line the reconstruction.

If we tolerate an RMS error of θ degrees at each sample point in $\mathbf{s}$, the required bound on the expectation of ϵ, i.e., the Mean Squared Error (MSE), is given by:

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

The Kohonen-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 formlated 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.

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

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

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. MSE in projection using SEG-1 and SEG-2 segmentation routines.

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. 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.

The majority of the provided human 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 subplot 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 data into two sets. The training set consisting of 75% of movement sequences and the test data consisting of 25% of sequences. We see that the reconstruction is marginally better with the training set figure 10. The median RMS reconstruction error is 19.4 degrees for the training data, and 22.27 degress for the test data. Both reconstructions, observed by the naked eye, qualitatively appear to have very high fidelity.

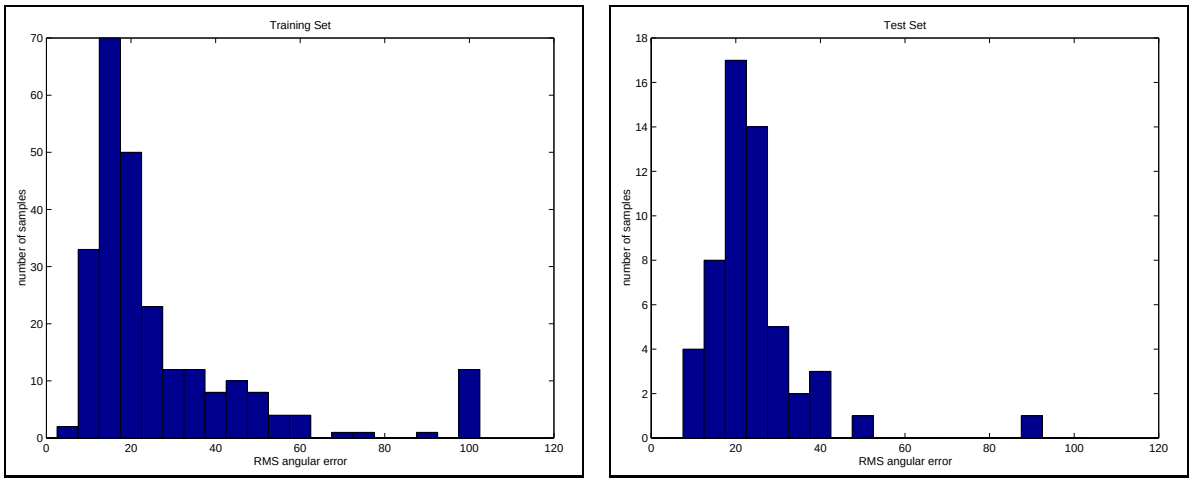

Fig. 10. A comparison of the histograms of RMS angular errors in the reconstruction of movements in the training and test data sets, demonstrating their similarity.

7 Simulation and Validation

To demonstrate the use of the derived primitives, we implemented them on a humanoid simulation with full dynamics. We first describe the simulator and then the controller implementation and results of the validation.

7.1 The Humanoids 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. Adonis consists of eight rigid links connected with revolute joints of one and three degrees of freedom (DOF). 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.

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 [12]. The simulation acts under gravity and accepts other external forces from the environment. Although collision detection, with itself or with the environment, are implemented, they were not used in the experiments presented here, as they were not relevant.

7.2 Motor Control in the Simulation Test-Bed

In Adonis, the static ground alleviates the need for explicit balance control. We used joint space PD servos to actuate the humanoid in execution of the derived primitives, after converting the movement segments into a quaternion representation. Specifically, the movement segments were given to Adonis 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 tracked. This approach has the potential of over-constraining the trajectory. However, this is merely to validate the

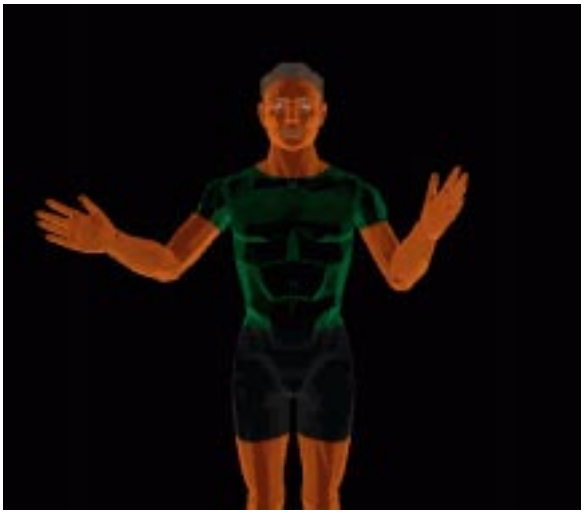

Fig. 11. The graphical display of Adonis, the dynamic humanoid simulation we used for validation.

reconstruction of the primitives and to visualize them. Our model mandates the use of motor controllers associated with each primitive [19, 14], an area of research we are currently pursuing.

The derived primitives, when executed in isolation, involve strokes, movements of the arm from one pose to another [29]. High frequency components contribute increasingly in lower eigen-valued primitives. When combined, they generate a reconstruction of the original movement executed by the human subject.

8 Discussion

The approach we describe is suitable for encoding movements into "eigenmovements" or primitives. Parts of any visible movement can thus be segmented and classified as belonging to a cluster associated with a controller. Then, control algorithms can be developed for these clusters. Simple modifications like temporal or spatial scaling applied to these control algorithms are expected to lead to control strategies for other movements that belong to the same clusters.

The choice of our 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.

The segmentation routine SEG-1 we described is not complete, i.e., it does not guarantee that any input movement will be partitioned. There is considerable overlap when two joint angles cross their zeros simultaneously at one point 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 [30].

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 presented method for deriving the primitives is the basis for our work on primitives-based motor control and imitation [19, 14]. It is also relevant to other work. For example, [29] 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.

9 Continuing Research

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.

A simple schematic representation of a primitives-based controller is shown in Figure 12. 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 controler 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.

Our eigen vectors are trajectories with specified velocities. As mentioned earlier we need to stretch and compress the normalized eigenvectors. This can be affected by multiplying the velocity along the trajectory by a scalar. [16] present Passive Velocity Field Control (PVFC) - a scheme to track a given trajectory. An interesting property of the controller is that it stabilizes any multiple of the specified velocity field and approaches it exponentially. We believe that this method can be put to good use in our system to effect a superposition of eigenvectors at the controller.

We can modify the torque strategy in a small neighborhood of the trajectory the controller was designed for by using a linear approximation of the movement-strategy map. A candidate set of points where linearization can be done are given by the cluster points mentioned earlier in 5.5.

10 Summary

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. 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. Raw movement data in the form of 3D locations of the arm joints is 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, 14]. 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.

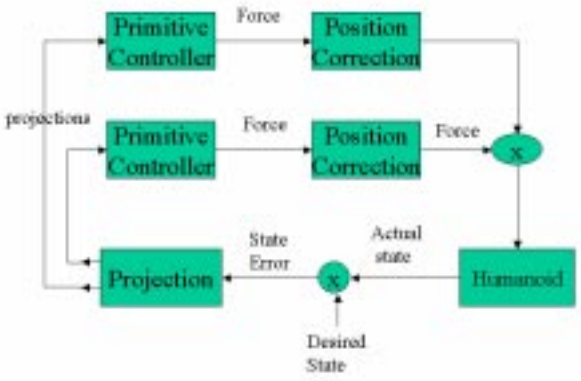

Fig. 12. The controllers model. The Primitive Controller is an individual controller for each primitive. The Position Correction module corrects for changes in the location of the center of the stroke. In this work, the Humanoid is modeled by Adonis. The Projection Module projects the errors back onto the eigenmovement space.

Acknowledgments

The research described here was supported in part by the NSF Career Grant IRI-9624237 to M. Matarić, in part by the National Science Foundation Infrastructure Grant CDA-9512448, and in part 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. The authors thank Aude Billard for her assistance in developing the segmenting algorithm.

References

1. M. Arbib. Schema theory. In S. Shapiro, editor, *The Encyclophedia of Aritificial Intelligence, 2nd Edition*, pages 1427–1443. Wiley-Interscience, 1992.
2. 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.
3. R. C. Arkin. *Behavior-Based Robotics*. MIT Press, CA, 1998.
4. A. Billard and M. Matarić. A biologically inspired robotic model for learning by imitation. In *Autonomous Agents*, pages 373–380, 2000.
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*, Killington, VT, 1996.
7. M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. In *Proceedings, CVPR*, pages 994–999. IEEE 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. *J. Neurosci*, pages 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. *J Neurophysiol*, pages 1760–1764, 1996.
11. N. Hogan. An organizing principle for a class of voluntary movements. *J. Neuroscience*, pages 2745–2754, 1984.
12. M. Hollars, D. Rosenthal, and M. Sherman. Sd/fast user's manual. Technical report, Symbolic Dynamics, Inc., 1991.
13. 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.

14. 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)*, 2000.
15. E. Kreighbaum and K. M. Barthels. *Biomechanics: A Qualitative Approach for Studying Human Movement*. Burgess Publishing Company, Minneapolis, Minnesota, 1985.
16. P.Y. Li and R. Horowitz. Passive velocity field control of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, 15(4), August 1999.
17. M. J. Matarić. Learning motor skills by imitation. In *Proceedings, AAAI Spring Symposium Toward Physical Interaction and Manipulation*, Stanford University, California, 1994.
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 C. Nehaniv & K. Dautenhahn, editor, *Imitation in Animals and Artifacts*. The MIT Press, 2000.
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. Mataric, 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. F. A. Mussa-Ivaldi. Nonlinear foce fields: A distributed system of control primitives for representation and learning movements. *Trends Neurosci.*, 1995.
24. 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.
25. H. E. Pashler. *The Psychology of Attention*. The MIT Press, 1999.
26. D. Pierce and B. Kuipers. Map learning with uninterpreted sensors and effectors. *Artificial Intelligence Journal*, 92:169–229, 1997.
27. 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)*, pages 7–8, MIT, Cambridge, MA, Sep 2000. Also IRIS Technical Report IRIS-00-384.
28. 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.
29. S. Schaal. Is imitation learning the route to humanoid robots. *TICS*, 3(6):233–242, 1999.
30. 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.
31. P. S. G. Stein. *Neurons, Networks, and Motor Behavior*. The MIT Press, Cambridge, Massachusetts, 1997.
32. K. A. Thoroughman and R. Shadmehr. Learning of action through combination of motor primitives. *Nature*, 407:742–747, 2000.
33. M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.
34. Y. Uno, M. Kawato, and R. Suzuki. Formation and control of optimal trajectory in human multijoint arm movement. *Biol Cybern*, pages 89–101, 1989.
35. S. Vijayakumar and S. Schaal. An o(n) algorithm for incremental real time learning in high dimensional space. In *ICML 2000*, Stanford, California, 2000.
36. P. Viviani and C. Terzuolo. Trajectory determines movement dynamics. *Neuroscience*, pages 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. *Biol. Cybern*, 73:15–25, 1995.