In *2003 IEEE/RSJ Int. Conf. on Intelligent Robotics and Systems*
Las Vegas, NV

Generating and Recognizing Free-space Movements in Humanoid Robots

Evan Drumwright and Maja J Matarić
USC Robotics Research Labs, Center for Robotics and Embedded Systems
941 West 37th Place, Mailcode 0781
Los Angeles, CA 90089-0781
{drumwrig|maja}@robotics.usc.edu

***Abstract*— We introduce a computationally efficient methodology for generating and recognizing free-space movements for humanoid robots. This methodology operates on exemplar-based representations of behaviors. Our method for actuating humanoid robots allows us to perform variations on a given behavior, resulting in a very human-like movement appearance. Besides control, this method also facilitates classification of perceived human and humanoid robot movement. We demonstrate the method on a physically-simulated humanoid robot with 132 degrees-of-freedom and evaluate our movement classification methodology on two data sets: human motion-capture data and joint-angle data sampled from the simulated robot.**

I. Introduction

Present humanoid robots are powerful machines with primitive brains. Such robots can be made to walk, but are able to perform few tasks in dynamic, human-situated environments. We aim to mitigate this problem by providing a methodology for performing and recognizing free-space movements of humanoid robots. The methodology we propose also serves to recognize free-space movements of humans, thereby facilitating human-robot interaction and communication.

Our system operates on a behavior representation based on the notion of *perceptuo-motor primitives*. Proposed by Matarić [13], these primitives are meant as a potential solution to the problems of controlling high degree-of-freedom humanoid robots as well as perceiving human and humanoid robot motion. In this model, on which our work is based, *primitives* serve as a basis set for generating movements; sequences and combinations of primitives produce desired behavior. Additionally, primitives also serve as the vocabulary for classifying observed movements into executable ones. Prediction of movement is enabled by mapping sequences of observed motions onto the basis vocabulary.

II. Related work

The computer vision community has explored the idea of primitive movements from a different perspective. Bregler and Malik [2] have proposed the existence of *movemes*, primitive motions in images that may be sequenced to exhibit complex behavior. Their work operates at the level of image blobs, and does not appear to be amenable to synthesizing humanoid movement.

There has been much work to date in the area of humanoid control. Matarić et al. [14] use three different control strategies to make a dynamically simulated humanoid dance. Hodgins and Wooten [9] use state-machine based algorithms for animating dynamically simulated human running, bicycling, and vaulting. Faloutsos et al. [6] control dynamically simulated humanoid characters rolling, rising, and performing other complex activities. Riley et al. [18] represent joint-angle trajectories with B-spline wavelets; the trajectories, which code for dancing, are then executed on a humanoid robot. Ijspeert et al. [10] use non-linear dynamical systems and locally linear learning to store and execute trajectories for swinging a tennis racket on a humanoid robot. Rose et al. [19] describe a system for interpolating motion data relevant for both humanoid robots and non-humanoid animated agents. Jenkins and Matarić [11] demonstrate automatic derivation of primitive movements from motion data for use in control of humanoid robots and agents.

Human action recognition from video streams has been researched by Bobick and Davis [1] use temporal templates and Petkovic et al. [16] use hidden Markov models for human action recognition. In contrast to these methods, our work on human activity classification assumes the existence of human joint-angle data, which is obtainable with motion-capture systems. This assumption obviates the need for human activity detection and simplifies the classification process.

Little research has been conducted on human action recognition using joint-angle or joint-position data. Mori et al. [15] use a fuzzy rule-based system, with rules determined by human judges, in order to classify human joint-angle data. Campbell and Bobick [3] successfully recognize ballet movements by first converting joint-position trajectories into a high-dimensional phase space and then operating within that space.

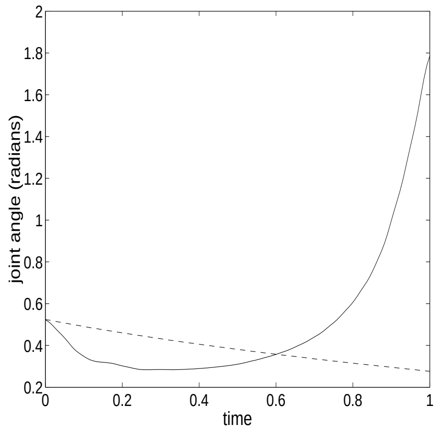

Fig. 1. Plot of the two exemplars of the *jab* behavior for a single degree of freedom in the shoulder.

III. Primitive representation

We define a *behavior* as a class of movements that perform a task in a manner that is semantically equivalent. Examples of behaviors include throwing a ball overhand, swinging a golf club, and hitting a forehand stroke in tennis. Each movement in a class varies according to environmental conditions and in order to satisfy the goal of the behavior, but remains semantically equivalent to the other movements in that behavior class. In the current work we limit the behaviors to free-space movements to target postures. A *perceptuo-motor primitive* is a parameterized model of a behavior, and is represented by a set of *exemplars*, key examples of that behavior. For example, if the behavior is a tennis forehand stroke, then exemplars might consist of swings to different targets. Each exemplar is represented by a set of joint angle trajectories relevant to that behavior (see Figure 1). In the tennis forehand example, the relevant joint angles are those of the shoulder, elbow, hand, and fingers; all of the other limbs of the robot can be oriented arbitrarily as long as intra-robot collisions do not result.

A set of exemplars defines a primitive. We choose exemplars so as to represent the extents of a behavior. Again, if the proposed behavior is a tennis forehand, then we might require only two exemplars which represent the trajectory of the arm when swinging at the low and high extremes of the movement range, thus providing the boundaries of the behavior space. Trajectories that lie inside of that space can be viewed as mixtures (linear or non-linear) of the trajectories that lie on the boundaries, and constitute the instances of that behavior. Variations are produced by interpolating between exemplars. Classification is performed by determining whether a given trajectory lies within this space. Prediction of future movement is done by minimizing the error between a reference trajectory and the output of the interpolation mechanism (with respect to the mixture parameters).

A. Representation of joint angle trajectories

We represent the individual joint angle trajectories using normalized radial basis functions (RBFs) [8] equally spaced over the interval $[0, 1]$. To encode a given degree of freedom of an exemplar, we first time-normalize the trajectory so that it lies in the interval $[0, 1]$. We are left with a vector of *n* time points *T* spanning $[0, 1]$ and a vector of corresponding joint angles *Y*. Each point $t \in T$ is then transformed into a *k*-dimensional vector, where K is the number of radial basis functions. A $n \times k$ matrix is produced using equations 1-4:

$$j = 1 \ldots k : center_j = \frac{j-1}{k-1} \tag{1}$$

$$\sigma = \frac{1}{2(k-1)} \tag{2}$$

$$\phi_i(x) = e^{\frac{-(x-center_i)^2}{\sigma}} \tag{3}$$

$$i = 1 \ldots n, j = 1 \ldots k : X_{ij} = \frac{\phi_j(T_i)}{\sum_{l=1}^{k} \phi_l(T_i)} \tag{4}$$

Above, the center vector stores the location of the kernel centers while σ is a constant that determines the uniform width of the kernels. Note that σ varies with the number of basis functions, so that the basis functions can “specialize” in smaller regions as *k* increases.

To perform RBF approximation, we now find multiplicative constants that minimize the error between the original and approximating trajectories. Formally, we need to find the vector of constants *A* that minimizes *J*:

$$J = \sum_{i=1}^{n} [Y_i - \sum_{j=1}^{k} A_j \phi(T_i)]^2 \tag{5}$$

This minimization can be performed by least-squares linear regression [8].

$$A = (X^T X)^{-1} X^T Y \tag{6}$$

We store the resulting vector, A, as an exemplar of a joint-angle trajectory.

B. Interpolation

We perform variations on a behavior by interpolating between exemplars. The interpolation mechanism was appropriated from the Verbs and Adverbs system of Rose et al. [19], and functions by combining both linear and non-linear mixtures of exemplars in proportion to arbitrarily chosen mixing parameters [19]. Additionally, inverse kinematics can be performed with the Verbs and Adverbs system [20]; satisfying postures are constrained to the exemplar space. In general, however, the actual interpolation algorithm is not important to our methodology.

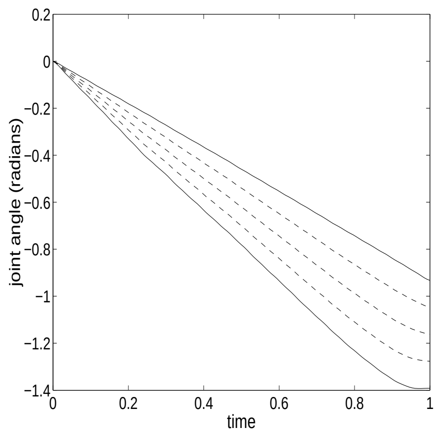

Fig. 2. Plot of interpolation between two exemplars of the *jab* behavior for a single degree of freedom in the shoulder. The exemplars are plotted with solid lines. The interpolated trajectories are plotted with dashed lines.

IV. Action via motor primitives

To illustrate the efficacy of our motor primitive representation for control of a real-time humanoid robot, we developed a behavior-based control architecture [12] for a physically simulated humanoid, *Ares*, which is similar in size, mass, and appearance to a male human. Its 44 Euler joints yield 132 degrees-of-freedom (DOF), including 72 DOF in the spine, 15 DOF in each arm, 12 DOF in each leg, and 3 DOF in the neck. Each arm is composed of five joints located at the clavicle, shoulder, elbow, wrist, and fingers; the fingers are treated as a single unit. The robot's legs include joints at the hip, knee, ankle, and toes. Such detailed articulation comes at the price of difficulty of control. For example, inverse-kinematics-based approaches are inherently underconstrained from the many redundant degrees-of-freedom. These approaches must constrain the solution in some way (usually via an optimization criterion), and often result in motion that appear unnatural.

We developed five motor primitives for *Ares*: *jab*, *hook*, *elbow*, *shield*, and *uppercut*, based on martial arts techniques. The represented behaviors are free-space movements that do not require interaction with the environment. For illustration, **http://robotics.usc.edu/∼agents/projects/martial-arts.html** contains animations of the five behaviors during execution.

A. Behavior-based control architecture

The control architecture we used, shown in Figure 3, is behavior-based, and consists of several concurrently executing *modules* responsible for controlling *Ares*. The modules determine which primitives can be executed, perform inverse kinematics, and send motor commands

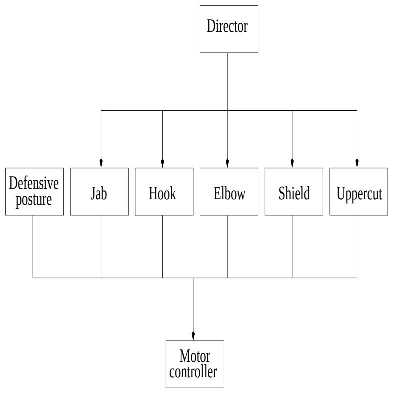

Fig. 3. Diagram of the robot's behavior-based control architecture

to the simulation. Additionally, each motor primitive (*jab*, *hook*, etc.), is encapsulated within a module.

The *director* module activates the *jab*, *hook*, *elbow*, *shield*, and *uppercut* behaviors, and acts as a reactive planner [7]; it is similar to Faloutsos et al.'s supervisor controller [6]. The *defensive posture* module continuously attempts to achieve the posture that serves as a precondition for the five motor primitives. The *motor controller* module receives joint commands from the primitive modules, resolves conflicts via a priority-based arbitration scheme [17], and sends outputs to the simulated joint motors. In keeping with the behavior-based control design philosophy, all modules are capable of perceiving the environment, almost all are capable of actuation, and all are simple and fast.

The primitive modules execute movements in the following manner. First, each primitive determines whether its precondition, described below, is met. Next, the desired trajectory is generated via interpolation. This trajectory is selected either directly, by specifying the mixing parameters of the exemplar trajectories, or by giving a target location and using inverse kinematics (see below). According to its reactive plans, the *director* module selects a behavior to execute, sets its duration for execution, and activates it. The behavior then sends joint commands to the *motor controller*, using the simulator clock so that the movement is executed for the precise duration. The commands are currently given in an open-loop fashion; our continuing work addresses real-time interactions with the environment.

B. Preconditions

A single precondition must be satisfied for *jab*, *hook*, *elbow*, *sheild*, or *uppercut* to activate: the *defensive posture* module must have achieved its posture. This condition is necessary because all motor primitives' sets of exemplars originate from that posture. However, the motor primitive representation does not intrinsically require a single originating point; each jab exemplar could emanate from a different starting position.

C. Performance

The performance of motor primitives for control can be evaluated based on the fidelity of the approximated trajectories to those of the exemplars, the quality of interpolated trajectories, and the computational time required to generate a desired trajectory.

The first factor, fidelity of approximated trajectories, is influenced directly by the number of radial basis functions used in the approximation. We have found that using 25 RBFs results in an excellent approximation, producing a mean-squared error per datapoint on the order of 10^{-6} radians.

The second factor, quality of interpolated trajectories, is determined by both the exemplars and the interpolation mechanism. Adding more exemplars in the region where artifacts occur [19] or replacing the interpolator is sufficient for improving performance in this area; thus, this factor is not specific to the primitive representation.

The final factor, computational time required to generated a desired trajectory, is of key importance in a real-time system. This factor is also dependent upon the interolation mechanism. Our interpolator has a running time on the order of $\Theta(MN + N^2 + NS)$, where N is the number of exemplars, M is the number of degrees-of-freedom for a given primitive, and S is the number of mixing parameters. Empirically, the interpolator produces a trajectory in about 200 μs for a 9-exemplar primitive on a 500 Mhz computer. We expect that most interpolation algorithms will produce comparable performance. Trajectory formation from interpolation is only required to start a movement; the resulting reference trajectory is stored for the movement's duration.

V. Action recognition

So far we have shown that we can use motor primitives, parameterized models of behavior, to facilitate efficient action. We now show how to use them for action recognition, by constructing a Bayesian classifier from a set of exemplars.

The Bayesian classifier attempts to determine the probability that an observed movement is an instance of a behavior, *C*. Put another way, it calculates the probability of *C*, given a time series *X* of length n, where X_i is a vector of joint angle values sampled at time *i*:

$$P(C|X) = P(C|X_n, X_{n-1}, \ldots, X_1)$$

By applying Bayes rule we transform the left hand side:

$$P(C|X) = \frac{P(X|C)P(C)}{P(X|C)P(C) + P(X|\overline{C})P(\overline{C})}$$

In the absence of *a priori* knowledge, it must be assumed that a behavior is as likely to occur as not. Thus, the prior probabilities $P(C)$ and $P(\overline{C})$ can be eliminated from the above equation. What is left are the probability of seeing behavior *C* given the time series and the probability of not seeing *C* given the time series.

A. Determining the conditional distributions

The primitive representation intrinsically yields the distribution $P(X|C)$. Unfortunately, the primitive does not indicate how to model $P(X|\overline{C})$. In the absence of better information, we must make several assumptions. First, we assume that the time series given $\overline{C}$ is independent:

$$P(X_n, X_{n-1}, \ldots, X_1|\overline{C}) = \\ P(X_n|\overline{C})P(X_{n-1}|\overline{C})\ldots P(X_1|\overline{C})$$

Next, we assume all of the joints involved in $\overline{C}$ are independent. Formally, if *j* joints are involved in behavior *C*:

$$P(X_i|\overline{C}) = P(X_{i,1}, X_{i,2}, \ldots, X_{i,j}|\overline{C})$$

$$P(X_{i,1}, X_{i,2}, \ldots, X_{i,j}|\overline{C}) = \\ P(X_{i,1}|\overline{C})P(X_{i,2}|\overline{C})\ldots P(X_{i,j}|\overline{C})$$

Finally, we assume that each joint angle value in time $X_{y,z}$ is uniformly distributed over the joint limits for each degree of freedom. In our humanoid, this corresponds to a range of $[-2\pi, 2\pi]$. Thus the distribution for every random variable is $X_{y,z} \sim U(-2\pi, 2\pi)$. Finally, we evaluate $P(X|\overline{C})$:

$$P(X|\overline{C}) = \left(\frac{\epsilon}{2\pi}\right)^{n\times j}$$

The base above is a simplified version of the standard evaluation of continuous uniform distributions. ϵ is a constant used for error tolerance, explained below in more detail.

Calculating $P(X|C)$ is a more difficult endeavor than that of calculating $P(X|\overline{C})$. The joint random variables of this distribution are not independent; instead, the joint angles and time series data are tightly coupled. Therefore, we cannot factor the distribution as we did with $P(X|\overline{C})$. We also have no basis for choosing a parametric distribution, which could simply the evaluation considerably.

For the purposes of illustrating a part of our determination procedure, let $n = 1$ and $j = 2$. Thus the motor primitive is composed of two DOF, and we consider only the current joint angle data. We then need to calculate:

$$P(X_{t,1}, X_{t,2}|C)$$

We have the means to calculate this distribution. It is simple to generate samples from the distribution off line. To perform the classification on-line, numerical integration over this function must be performed. Unfortunately,

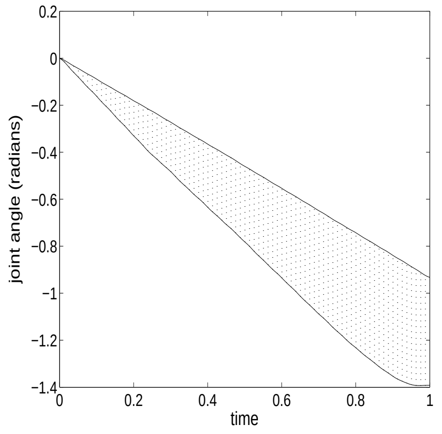

Fig. 4. Plot of the mixture space between the two exemplars of the *jab* behavior for a single DOF in the shoulder. The exemplars are plotted with solid lines.

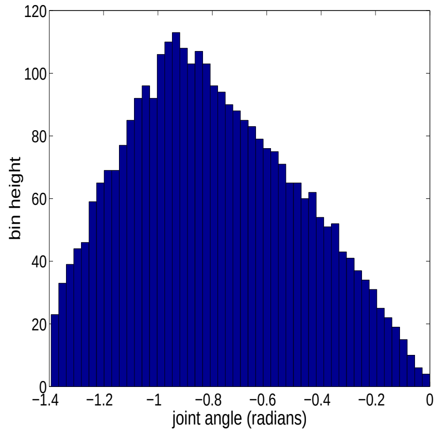

Fig. 5. Histogram produced by sampling the mixture space seen in Figure 4 over $[0, 1]$.

multidimensional integration is quite slow. Additionally, the number of dimensions of the general joint distribution is $n \times j$, making this method intractable except for toy primitives. In place of this approach, we factor the distribution into a set of a one-dimensional conditional distributions using the probability chain rule:

$$P(X_{t,1}, X_{t,2}|C) = P(X_{t,1}|X_{t,2}, C)P(X_{t,2}|C)$$

The second term, $P(X_{t,2}|C)$, is trivial to calculate. We sample the mixture space of the exemplars uniformly (see Figure 4). We then build a histogram from the samples (see Figure 5). Normalizing this histogram yields a probability distribution.

Determining $P(X_{t,1}|X_{t,2}, C)$ is only slightly more complicated. We are conditioning on new information, namely that $X_{t,2} = x$. For the sake of illustration, assume that $x = -0.9$. If we examine Figure 4, it is apparent that the mixture space can assume this value only during

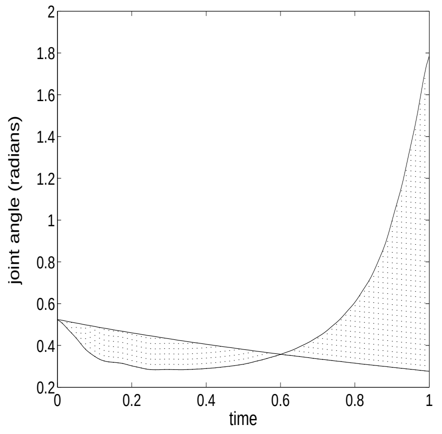

Fig. 6. Plot of the mixture space between the two exemplars of the *jab* primitive for a second DOF in the shoulder. The exemplars are plotted with solid lines.

the time interval $[0.57, 1]$. Samples can then be drawn from the mixture space in this limited domain, yielding the distribution $P(X_{t,1}|X_{t,2}, C)$.

We have just demonstrated how to evaluate a conditional distribution with the conditioning variable being another degree of freedom at the same t in the time series. We now show how to evaluate a conditional distribution with the conditioning variable being the same degree of freedom but at a different t. In this case, we evaluate:

$$P(X_{t,1}, X_{t-1,1}|C) = P(X_{t,1}|X_{t-1,1}, C)P(X_{t-1,1}|C)$$

$P(X_{t-1,1}|C)$ is conditioned only on the class and can therefore be calculated in the same manner as $P(X_{t,2}|C)$. To calculate $P(X_{t,1}|X_{t-1,1}, C)$, we again construct a hypothesis from our conditioning variable. However, the hypothesis is constructed in a different manner. Consider that we observe a joint value that can occur in the interval $[q, r]$ in a trajectory. Given that information, we know that the next joint value should occur in the interval $[q, 1]$ of that trajectory (if the joint value is an instance of C). The hypothesis construction is based on this idea.

To illustrate the determination of $P(X_{t,1}|X_{t-1,1}, C)$, assume that $X_{t-1,1} = 0.5$. If the mixture space is that pictured in Figure 6, then the value 0.5 can be found within the intervals $[0.2, 0.5]$ and [0.74,1]. Thus, given that $q \in [0.2, 0.5] \cup [0.74, 1]$, the next value of the trajectory will come from $[0.2, 1]$. The distribution for $X_{t,1}$ is sampled only in the interval $[0.2, 1]$. The resulting distribution is shown in Figure 8.

Finally, we must extend our discussion of evaluation to handle conditioning on multiple random variables. This extension is rather simple. We determine set(s) of intervals for each random variable being conditioned upon in the manners previously described. If the random variable comes from the same point in time t in the time series

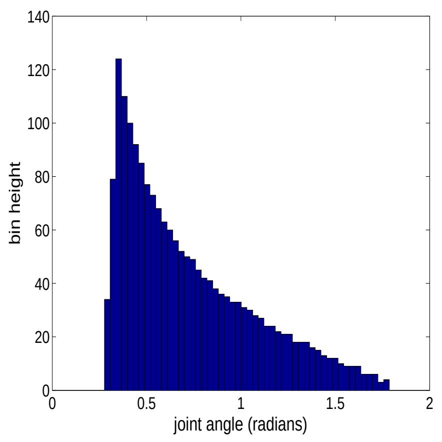

Fig. 7. Histogram produced by sampling the interval $[0.57, 1]$ of the mixture space seen in Figure 6.

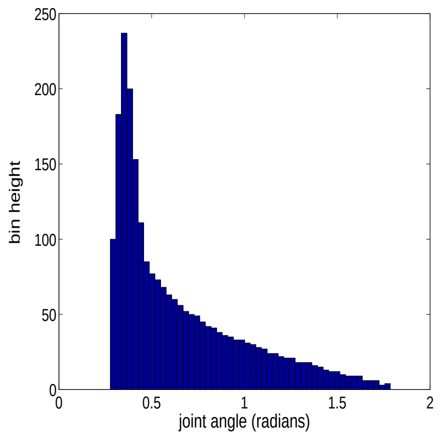

Fig. 8. Histogram produced by sampling the interval $[0.2, 1]$ of the mixture space seen in Figure 6.

as the variable of the distribution that we are trying to determine, we use the first method. If the random variable comes from a different point in time, we use the second method. We take the intersection of all of the sets that have been produced to determine the interval to be sampled. If the interval is the null set, then the resulting distribution is uniform over $[-2\pi, 2\pi]$.

B. *Accounting for joint error*

We previously discussed a constant ϵ used in computing the conditional distribution $P(X|\overline{C})$. This value is used to calculate $P(X|C)$ as well. Essentially, whether we are trying to evaluate $P(X = x|C)$ or $P(X = x|\overline{C})$, we want to compute the following:

$$\int_{x-\epsilon}^{x+\epsilon} P(Y = y)dy$$

Larger values of ϵ allow for some error in joint angle measurement or controller deviation from the intended trajectories. Where such errors are minimal, smaller values of ϵ should increase classification accuracy. We set ϵ to 0.75 degrees, a relatively small value.

TABLE I

CLASSIFICATION ACCURACY

Dataset	$n = 1$	$n = 2$
Ares movements	96.61%	96.61%
Choreographed animations	99.94%	99.94%
Motion-capture	99.97%	99.98%

C. *Computational complexity*

We require that the perceptual system be fast, so that it, like the actuation system, can be used in a situated robot. Our algorithm exhibits a complexity of $\Theta(knj)$, where k is the number of primitives in the robot's vocabulary, n is the number of points in the time series, and j is the number of DOF involved in a behavior. This translates into a running time of approximately 20ms per primitive on a 2 Ghz Pentium 4 processor when $n = 2$ and $j = 12$. This performance precludes real-time recognition of multiple primitives, unless multiple, faster processors are used. Alternatively, speed can be increased significantly at the expense of accuracy by reducing sampling and interval search granularities.

D. *Evaluation*

We evaluated the described classifier using three datasets consisting of 209,188 data points, where each data point is a vector of joint positions. The first dataset contained movements performed on our simulated humanoid robot, generated by the motor primitives (*jab*, *hook*, etc.); it also contained movements required to transition between primitives. The second dataset consisted of over 50 animations (e.g., *swim*, *reach*, *swat bees*, etc.) choreographed by Credo Interactive [4]. Each animation was composed of many movements, and all animations include substantial movements of the arm. The third dataset was composed of 500 motion-capture files that represent behaviors such as tennis, scrubbing, bowling, and walking; this dataset, as well as a descriptions of the included motions, is also available from Credo Interactive [5].

As can be seen in Table I, the classifier performed very well on all datasets. It is evident that few of the movements included in the second and third datasets enter the mixture space of either behavior. This is unfortunate because we are almost entirely prevented from observing the additional classification power gained by increasing n; considering previous joint data is useful only to reduce false-positive classifications. The additional power of greater n might prove to be unnecessary. We synthesized just five behaviors, which are highly similar semantically and structurally. Yet there is good separability between these five behaviors, as evidenced by the high accuracy in classifying the first dataset. Furthermore, there is almost complete separability between these behaviors and those encoded in the second and third datasets, even though they span much of the same Cartesian space.

VI. Conclusions

We have introduced a methodology that enables both real-time action and recognition of free-space movements for humanoid robots, which requires only a few exemplars of a behavior. We demonstrated how these exemplars, with the aid of an interpolation mechanism, can be used to control a robot. We also demonstrated how to classify the actions of humanoid robots using only these exemplars. Our future work will validate the action recognition methodology further on motion-capture data exclusively representing martial arts movements; the apparent similarity between movements in this domain should more rigorously test the discrimination capabilities of our classifier. We will also investigate sequencing and combining behaviors to produce more complex movements.

VII. Acknowledgments

This work was funded by the Defense Advance Research Projects Agency (DARPA) grant MARS DABT63-99-1-0015.

VIII. REFERENCES

[1] A. Bobick and J. Davis. The recognition of human movement using temporal templates. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(3):257–267, March 2001.

[2] C. Bregler and J. Malik. Learning and recognizing human dynamics in video sequences. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 568–574, San Juan, Puerto Rico, 1997.

[3] L. Campbell and A. Bobick. Recognition of human body motion using phase space constraints. In *Proc. of 5th Intl. Conf. on Computer Vision*, pages 624–630, 1995.

[4] Credo interactive, lifeforms studio 4.0. http://www.credo-interactive.com, 2001.

[5] Credo interactive, mega mocap v2. http://www.credo-interactive.com, 2002.

[6] P. Faloutsos, M. van de Panne, and D. Terzopoulos. The virtual stuntman: dynamic characters with a repertoire of motor skills. *Computers and Graphics*, 25(6):933–953, December 2001.

[7] J. Firby. *Adaptive Execution in Complex Dynamic Worlds*. PhD thesis, Yale University, January 1989.

[8] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan/IEEE Press, 1994.

[9] J. Hodgins and W. Wooten. Animating human athletes. In Y. Shirai and S. Hirose, editors, *Robotics Research: The Eighth Int. Symposium*, pages 356–367, Berlin, 1998. Springer-Verlag.

[10] A. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, 2002.

[11] C. Jenkins and M. Matarić. Automated derivation of behavior vocabularies for autonomous humanoid motion. In *Proc. of Autonomous Agents and Multi-agent Systems*, Melbourne, July 2003.

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

[13] M. Matarić. Sensory-motor primitives as a basis for imitation: Linking perception to action and biology to robotics. In K. Dautenhaun and C. Nehaniv, editors, *Imitation in Animals and Artifiacts*, pages 391–422. MIT Press, 2000.

[14] M. Matarić, V. Zordan, and M. Williamson. Making complex articulated agents dance: an analysis of control methods drawn from robotics, animation, and biology. *Autonomous Agents and Multiagent Systems*, 2(1):23–44, March 1999.

[15] T. Mori, K. Tsujioka, M. Shimosaka, and T. Sato. Human-like action recognition system using features extracted by human. In *Proc. of the 2002 IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, pages 1214–1220, Lausanne, Switzerland, October 2002.

[16] M. Petkovic, W. Jonker, and Z. Zivkovic. Recognizing strokes in tennis videos using hidden markov models. In *Proc. of Int. Conference on Visualization, Imaging, and Image Processing*, Marbella, Spain, 2001.

[17] P. Pirjanian. Behavior coordination mechanisms - state-of-the-art. Technical Report IRIS-99-375, IRIS, School of Engineering, University of Southern California, October 1999.

[18] M. Riley, A. Ude, and C. Atkeson. Methods for motion generation and interaction with a humanoid robot: Case studies of dancing and catching. In *Proc. of AAAI and CMU Workshop on Interactive Robotics and Entertainment*, Pittsburgh, PA, April 2000.

[19] C. Rose, B. Bodenheimer, and M. Cohen. Verbs and adverbs: Multidimensional motion interpolation using radial basis functions. *IEEE Computer Graphics and Applications*, 18(5):32–40, 1998.

[20] C. Rose, P. Sloan, and M. Cohen. Artist-directed inverse-kinematics using radial basis function interpolation. *Eurographics*, 20(3):239–250, 2001.