International Journal of Humanoid Robotics

Improvements on Action Parsing and Action Interpolation for Learning through Demonstration

Jeff Lieberman*and Cynthia Breazeal
Robotic Life Group, Media Arts and Sciences, Massachusetts Institute of Technology, 77 Massachusetts Avenue
Cambridge, Massachusetts 02139, USA
xercyn@media.mit.edu, cynthiab@media.mit.edu

Received (31 May 2004)
Revised (Day Month Year)
Accepted (Day Month Year)

Programming humanoid robots with new motor skills through human demonstration is a promising approach to endowing humanoids with new capabilities in a relatively quick and intuitive manner. This paper presents an automated software system to enable our humanoid robot to learn a generalized dexterous motor skill from relatively few demonstrations provided by a human operator wearing a telemetry suit. Movement, end effector, stereo vision, and tactile information are analyzed to automatically segment movement streams along goal-directed boundaries. Further combinatorial selection of subsets of markers allows final episodic boundary selection and time alignment of tasks. The task trials are then analyzed spatially using radial basis functions [RBFs] to interpolate between demonstrations using the position of the target object as the motion blending parameter. A secondary RBF solution, using end effector paths in the object coordinate frame, provides precise end-effector positioning and orienting relative to the object. Blending of these two solutions is shown to both preserve quality of motion while increasing accuracy and robustness of object manipulation.

Keywords: Learning by Demonstration; Action Parsing.

1. Introduction

The ability to quickly teach humanoid robots new motor skills by human demonstration continues to be an important area of research (see [1] for a review). Some of the earliest work in this area is called *learning by demonstration*. In this approach, the robot (often a robotic manipulator) learns how to perform a new task by visually observing a human perform the same task. In *task-level imitation*, the robot learns how to perform the physical task of the demonstrator, such as stacking blocks[2] or peg insertion[3], by acquiring a high-level task model (such as a hierarchy of goal states and the actions to achieve them) from observing the effects of human move-

*50 Mass Ave., Cambridge MA 02139

ments on objects in the environment. Learning by demonstration generally breaks down into two tasks - motion temporal segmentation, and spatial interpolation.

In other work with highly articulated humanoid robots, learning by demonstration has been explored as a way to achieve efficient learning of dexterous motor skills[4,5]. The state-action space for such robots is prohibitively large to search for a solution in reasonable time. To address this issue, the robot observes the human's performance, often using both object and human movement information provided visually to estimate a control policy for the desired task. The human's demonstration helps to guide the robot's search through the space, providing it with a good region to initiate its own search. If given knowledge of the task goal (in the form of an evaluation function), robots have learned to perform a variety of physical tasks—e.g., learning the game of "ball in cup" or a tennis forehand[6,7] by utilizing both the demonstrator's movement and that of the object.

1.0.1. *Demonstrations via Telemetry*

Guidance of an action with a telemetry suit is an often used alternative approach[11,8]. Fagg[9,10] has used this, along with tactile feedback, to study grasping actions, but has not generalized to tasks in general.

Similarly, Peters et. al.[11] have explored teaching NASA JSC's Robonaut tool use skills through teleoperation, segmenting the action into known primitives, and interpolating these to form full actions. This method required manual parameter addition and tweaking for adequate results. It is difficult in the examples shown to test the limitations of the segmentation techniques, because the input examples were very similar, so it is possible that many techniques would segment these correctly. More difficult animations and input data were not available.

In our technique, no manual parameters or motion features are added, and all parameters are generated for the specific trial data, so that all data is processed autonomously, and more difficult scenarios and data inputs with more noise are still segmented accurately.

1.1. *Segmenting Movement and Motion Interpolation*

Baldwin and Baird[12] pinpoint the need to discern intention in human actions. As stated, the fundamental goal of this research is to discover "What kind of information about intentions and intentionality is actually available in the surface flow of agents' motions?" Identifying the relevant statistical patterns of motion, and how predictive they are of the human's underlying intentions is an important topic of investigation.

Similar work is necessary for robot learning through demonstration. A robot might possess the mechanical ability to complete a task [eg. grasping a ball, moving it to a cup, releasing it in the cup] but how can a robot recognize this task when demonstrated? If it uses actions it already knows, how might it blend these together to form a newly observed action?

Many have explored these issues. Matarić[13,8,14] and Jenkins[15] have explored the mean-squared velocity of joints to segment motion along intentional boundaries (see section 3.0.1). Principal Components Analysis is used to represent a movement trajectory with known primitives. Others such as Tennenbaum et. al[16] combine PCA with multidimensional scaling. Error tolerance in these methods remains a difficult issue. Examples used were typically very similar and of insufficient difficulty to test a segmentation algorithm's generalizability. Upon initial use of the Matarić technique, segmentation was not performed properly without manual manipulations.

Similarly, for Motion interpolation, Radial Basis Functions are most commonly used to interpolate between actions at known positions, to generate motions at arbitrary positions. The most accurate robotic spatial interpolation demonstrations[11] still possesses large errors for even theoretical data inputs, which cause repeated task failure.

1.2. *Approach*

To address the shortcomings of previous works, we have developed an automated software system (see Figure 1) to allow our humanoid robot to learn generalized motor skills from demonstrations given by a human operator via teleoperation. Data is captured as the human guides the robot through a task in different parts of its workspace. Joint angle and end effector motions are measured through time, and the motion is segmented into possible goal-directed streams, which are then weeded to find final episodic boundary selection, and thus time align tasks.

We then analyze the trials with radial basis functions, to interpolate tasks to other parts of the workspace. An analysis of the tasks from the object's reference frame and in the robot's reference frame, along with motion variances in each frame, leads to an *objectivity* measure of how much any part of an action is absolutely oriented and how much is object-based. A secondary RBF solution using end effector paths from the object reference frame provides precise end-effector path planning relative to the object. The objectivity measure is used as a blending weight between solutions, to preserve quality of motion [initial RBF solution] and provide accuracy and robustness in object interactions [the secondary RBF solution]. Our improvements in spatial interpolation are currently only employed on the animated robot model due to non-linearities in the mechanical system; however, we have devised a new system that provides zero theoretical motion error, contrary to previous work.

2. Data Capture and Task Data Integration

This section provides an overview of the hardware and software systems used to capture the human task demonstration data. Figure 2 shows the hardware layout involved in this recording system, which then goes on for software processing.

We use a Teleoperation suit (a 'Gypsy Suit' made by Animazoo) to measure the operator's joint angles through time, with 42 joints recorded. Table [1] lists the joints used in this analysis. The Gypsy Suit Display Recording Software allows suit

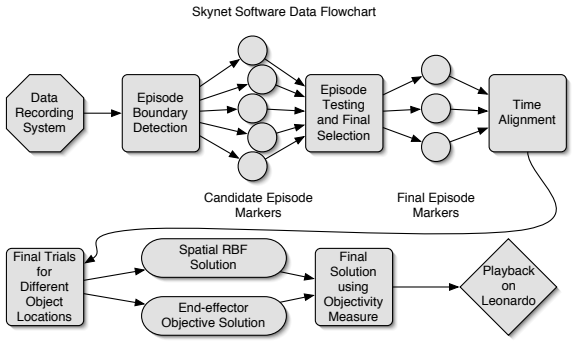

Fig. 1. The layout of the skynet learning by demonstration software. Data is captured by a teleoperation suit, stereo vision cameras, and a tactile sensor, and travels into the software, as a task is demonstrated repeatedly. From these task trials, several stages analyze the motion primitives and episode boundaries that make up these larger sequences, so they can be properly time-aligned and compared. Once this time alignment is complete, the trials are analyzed spatially, to determine how the action changes based on the movement of the object the robot is interacting with, and from this, a generalized motion sequence is generated for any object position, providing a general solution to that action for all objects in the robot's workspace.

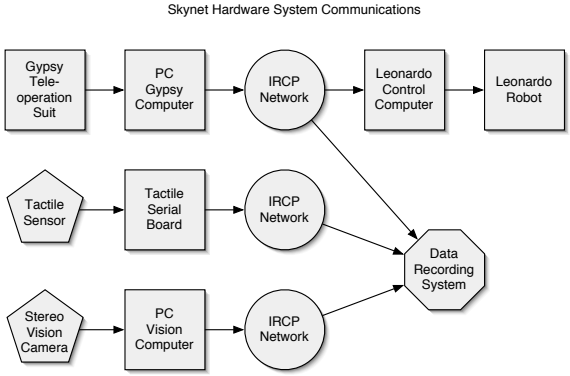

Fig. 2. The hardware layout for Leonardo's task recording and playback system. Motion is captured from the teleoperation suit, while tactile information and stereo vision data are grabbed concurrently, each traveling through its own hardware processor, and onto the network, where it is all saved by the recording computer. Meanwhile, the teleoperation suit data is passed onto the Leonardo control computer and is used to control Leonardo in real time.

calibration. Tactile feedback information is measured using Force Sensitive Resistors [FSRs] (made by Interlink Electronics) mounted in the robot's hands, responding to forces from under 1N to over 100N, also polled at 60 Hz. All data is transferred by our Intra-Robot Communications Protocol [IRCP] network and stored. A stereo vision camera (made by Videre Design) is used to locate Leonardo's objects. These

Fig. 3. Leonardo the humanoid robot, possessing 65 degrees of freedom that allow it to interact expressively, as well as perform simple object manipulations. Here Leonardo is surrounded by some of the buttons it uses in object manipulation tasks.

cameras consist of two CMOS sensors. The stereo visual data is analyzed at 10 Hz yielding approximately 240 levels of depth perception to locate objects with an accuracy of 0.5" in position. All data is captured at 60 Hz.

Table 1. A list of all the teleoperation suit joints employed in the action recording process.

Joint Number	Joint Name
1	Torso Rotation
2	Hip Front Back
3	Hip Side Side
4	Right Shoulder In/Out
5	Right Shoulder Forward/Backward
6	Right Upper Arm Rotation
7	Right Elbow In/Out
8	Right Forearm rotation
9	Right Wrist In/Out

As the operator guides the robot through a task, the telemetry, tactile feedback, and visual object identification data are recorded simultaneously. We low-pass the data to allow for smoother derivative calculation. The Robotics Toolbox[17] is used to create a kinematic model of Leonardo's torso-right arm subsystem to analyze joint and end-effector motions. Motions involving both hands are not currently dealt with by this system.

3. Improving Episode Analysis

In order to analyze the separate trials of an action spatially, we first time-align the trials so that each part of a complex action happens simultaneously to other trials. This is typically done using episode analysis, where motions are analyzed in order to segment them into constituent sub-actions. These sub-actions, known as *episodes*, are then aligned. Simple time averaging of complex actions is often insufficient - key features, such as absolute end effector minima, become sufficiently misaligned so that an averaging of motions loses salient features present in every trial. For instance, a typical trial might consist of a reach, grasp, release, and recoil; only with full alignment of these motions will we see the qualities present in each. Figure 4 shows that even if we normalize the times of trials and average the motion to produce somewhat clean results, important aspects of the motion are left unresolved. Notably, the maxima and minima are not aligned. As a result, the joint will never reach the extremes that it would if the maximal and minimal points were better aligned. Therefore, we argue that a better technique is needed to automatically determine the natural boundaries between episodes.

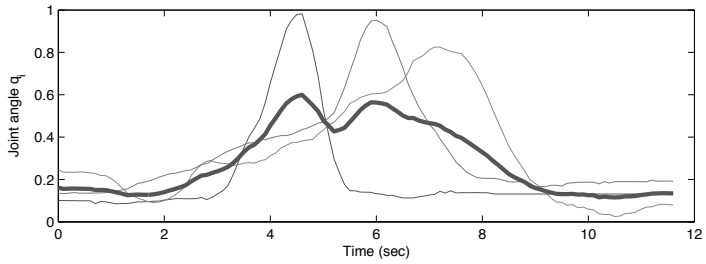

Fig. 4. A view of three example joint actions, averaged in length, and then the resultant average motion. Note that the maxima and minima of the original motions are lost, due to misalignment.

3.0.1. *The Matarić Segmenting Technique*

The most commonly employed technique of auto-segmentation action streams[13,15] is based on the observation that when humans engage in a complex action, they typically change direction and speed between each segment of that action, with an associated acceleration. Matarić[13] exploits this fact, by looking for changes in the overall motion of the joints involved in the robot.

To find episode boundaries, the technique works as follows. Generate the Mean Squared Velocity (MSV) of the joints, as shown in Eq. 1, where N represents the

number of joints, and $q(i)$ is the position of joint i.

$$MSV(t) = \sum_{i=1}^{N} \left(\frac{dq(i)}{dt} \right)^2 . \quad (1)$$

For each time t, if $MSV(t-1) \leq c$ and $MSV(t) \geq c$, search through the remaining times until finding u such that $MSV(u) > kc$, for some c and k, chosen in advance. If a time u is found, then this value of t is an episode beginning. Switching inequalities yields the method to find episode endings. A more detailed description of the Matarić segmenting technique is given in[13,15]. This technique looks for sufficiently low joint motion that reach sufficient highs, and vice versa.

Figure 5 shows the MSV during a button push, with marked episode beginnings and endings, for three trials. The episodes correspond to the reach, push down, release, and retract phases. The last two phases are analyzed as one episode, as the retraction from the button and the retraction back to the starting phase do not involve much slowdown. The Matarić technique, mentioned or employed in[13,15,8,14,11],

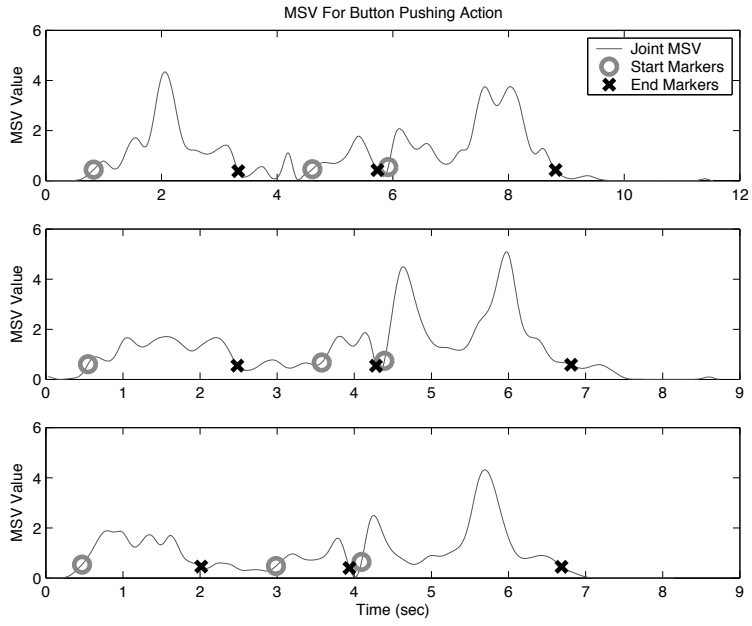

Fig. 5. MSV analysis of a button push, for three different trials. Episode beginnings and endings are marked as noted in the legend. The lower graph shows the tactile feedback data. As seen in the episode graphs, this separation corresponds with the reaching to grasp for the button, the press of the button, and the retract from the button action, which involves retraction from the button, as well as back to the starting position.

often has special adjustments made to deal with certain implementations. For ex-

ample [11] combined the general MSV analysis with an analysis of the maxima and minima of the elbow joint [on Robonaut]. This requires manual tweaks, and is a feature that changes for every action. Below are improvements that automatize this system.

3.0.2. *Modifications to the Mataric Segmenting Technique*

c and k are manually tuned to create proper episodes, and certain boundaries are missed here, due to the lack of tactile feedback. The goal is to find all possible boundaries, and later discern the boundaries common to all trials.

Sudden contact between the end effector and object usually signifies a new episode, but this is often missed, as motion does not necessarily cease. We add to the MSV a tactile factor [scaled by $c_{tactile}$] to accomodate these changes:

$$MSV_{\text{improved, tactile}}(t) = \sum_{i=1}^{N} \left(\frac{dq(i)}{dt} \right)^2 + c_{tactile} \frac{dw(t)}{dt}, \tag{2}$$

where $w(t)$ represents the tactile sensor value at time t.

We also automatically generate the Matarić parameters to compensate for overall high or low-motion actions. c represents the low-motion cutoff value, and k the ratio of high-motion to low-motion necessary for an episode. We choose c and k as such,

$$c = < MSV > - \frac{1}{2}\sigma_{MSV}, \tag{3}$$

$$k = \frac{< MSV > + \frac{1}{2}\sigma_{MSV}}{< MSV > - \frac{1}{2}\sigma_{MSV}}, \tag{4}$$

such that $ck =< MSV > + \frac{1}{2}\sigma_{MSV}$, creating a symmetric distribution about the mean of the MSV. This automatically creates constants for the generation of several episode boundary markers.

3.1. *Future Improvements*

In the future, several features will increase the episoding system's robustness. First, both c and k could change through a trial, allowing finer tuning of episode choices. Also, the addition of the end effector orientation vector and of joint maxima/minima would allow more episode boundaries to be detected, as was done manually in[11]. Furthermore, using direct joint angle measurements on the robot would provide a truer MSV than angles taken from a teleoperation suit.

4. Combinatorial Episode Selection and Canonical Motion Creation

The goal of the previous section is to find all the possible markers for episode separation for each trial. In real practice, different trials are disparate enough that

the episode markers will not possess an isomorphism between trials; so, we build a technique to search through possible subsets of markers, to determine which are truly reliable. We choose all possible episode boundaries from all trials, and test alignment between them to determine which markers are dominant through all trials. Once this marker subset space is searched, we mark true episode boundaries, and dynamically time warp the trials to align them.

4.1. *Combinatorial Selection and Alignment of Episodes*

In order to find the proper number of true episode boundaries, we find the minimal size boundary set through trials, $N_{min} = \min(|A_{T_I}|)$ where A_{T_i} is the set of episode beginning markers for trial i. We search for the best matches to this set of markers. Trials are compared in a pair-wise sense. First, we find all valid marker subsets of size N_{min}. Then, for each subset, we time align the trials to test those marker's validity. To align these trials, we first find the average times for each respective episode boundary, $M_{average}(j)$. We dynamically time warp the time series so that the individual markers from each trial align with these average times (this is done using Hermite polynomial splines, which allow high distortion while maintaining monotonicity in the time series). An example of this time warping for two trials is shown in Figure 6. A statistical correlation of the warped trials provides a measure of alignment between the two. For each trial, this ranking is added to a subset-specific list, to be combined with all of the iterations over all trial and subset combinations.

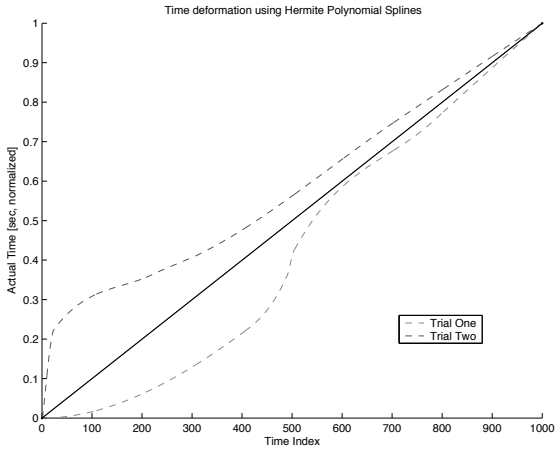

Fig. 6. A view of the splining of two trials' time series, showing alignment to the average episode boundary time markers, $M_{average}(i)$. The straight line represents the standard linear time development, and the two curved paths skew the time so as to align the markers in the trials. A steeper curve indicates faster time evolution locally.

In order to rank how well a subset selection works overall, we total the rankings

that involve that subset (ie. every ranking where it is one of the two trial sets being compared). Whichever subset possesses the highest total is chosen as the true marker set. Figure [7] shows three trials involved in a button pressing action, the initial set of possible episode boundary markers for each, and then the final selection of chosen markers.

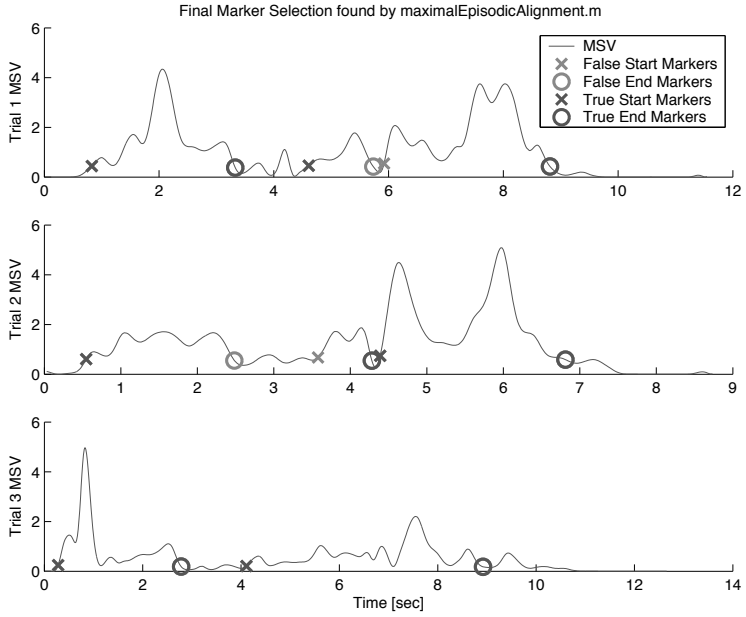

Fig. 7. A view of three trials involved in a button pressing action. The initial possible episode boundaries are marked in each example, as well as the final selection of markers, for use in combination for a canonical motion.

4.2. *Canonical Motion Creation and Results*

The episode analysis aligns the time sequences of the multiple actions, so they can be compared to generate a canonical representation of the action, to be performed anywhere in Leonardo's workspace. Now that the true subsets have been found, we dynamically time warp each of the trials, so that the episode markers align with the average time positions of the markers from all trials.

Figure [8] shows, for a button pressing trial, the final alignment between three separate trials, as well as the averaged motion of those three trials. This is not joint data, but the end effector MSV comparison. This is compared in the same Figure to the MSV data purely averaged, without the use of the alignment methods.

Figure [9] shows the importance of this analysis: the resultant three dimensions

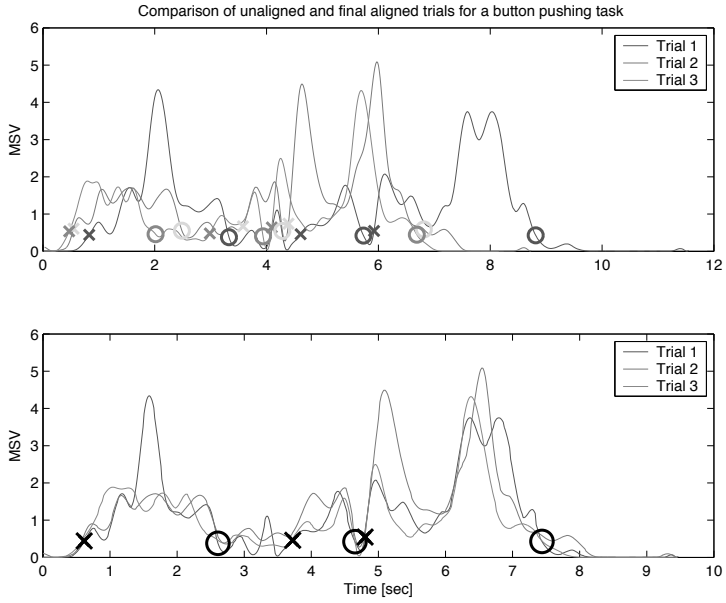

Fig. 8. A button pressing trial, with final MSV analyzed alignment on three trials. This view of the aligned MSV is compared to the original time averaged MSV, which has lost almost all of its distinctive features.

of end effector travel, after episodic alignment, for the same button pressing action. When the systems are aligned, we generate a motion that is cleaner than the original input systems, and yet reaches the important areas that the original motions reached. The default averaged motion did not resemble the original inputs in terms of their spatial progression, due to the problems from time misalignment.

5. Interpolation of Motions

So far we have performed an analysis that time-aligns an action performed in different areas of the robot's workspace. What remains is an analysis of the spatial variances in the motions, to generalize the performance of these actions to the entire workspace. Precision is important, in order to properly interact with objects.

We rely on as few input trials as possible, in order to minimize fatigue of the teleoperator. The fewer data points causes spatial interpolation to lose precision, and as of yet has not produced useful results. Typically several trials at each location need to be completed to reduce noise[11,18]. Here we repeat not points. The algorithm we develop captures both the quality of motion and the necessary accuracy of the end-effector to create a useful task generalization.

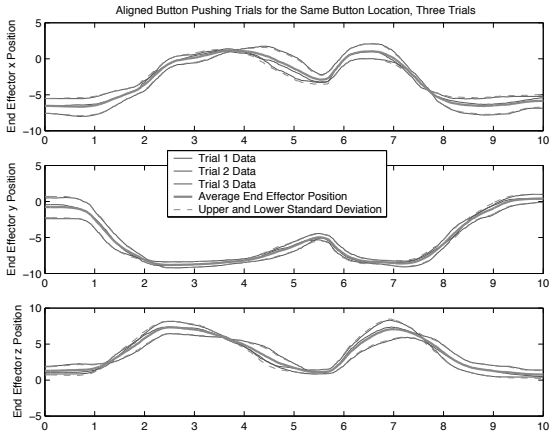

Fig. 9. Another comparison of unaligned and MSV-aligned data, this time for Leonardo's joint system. The images show the three dimensions of end effector motion of Leonardo's end effector path. Compared to the regular averaged motion, described earlier, this motion retains all maxima and minima of the original motions, as well as important velocity/derivative data.

5.1. *Switching to the Animation Model*

For the remaining research we use data from motion animations generated in Maya - our teleoperation data was imprecisely calibrated from gypsy suit to robot due to large joint nonlinearities, and teleoperation angles are very different from the robot's true joint angles, due to the lack of haptic feedback. In current research we are building a more accurate model to map these nonlinearities which restrict us to the animation models, so that we will soon be able to use direct human input for learning. However, the model suffices to show marked theoretical improvements over previous methods.

5.2. *Interpolation Techniques using Known Motions*

We are given several time series of motions $\vec{x_i}(t)$, all oriented involving an object at a known position, $(O_{x,i}, O_{y,i}, O_{z,i})$, where the point $\vec{O}$ indicates the most salient point of the object [i.e. for button pushing, this is the point of contact between hand and button]. Given this information, we generate a new trajectory $\vec{x}(t)$ that will describe the motion to achieve the task when the object is located at point (O_x, O_y, O_z). The most common interpolation method for 3d time series is Verb-Adverb theory, instantiated with Radial Basis Functions[18].

5.2.1. *Verb Adverb Theory and Radial Basis Functions*

Verb-Adverb theory has been applied for many years, primarily for computer graphics animation[19]. A detailed explanation of this theory is given elsewhere[18] and in

the general literature. The overview here will be brief and functional.

A Verb is a motion of some sort, i.e. 'grasping.' The Adverb is the parameter of the motion that we desire to change - in this case, the parameter is the object position, although it could also refer to an emotional parameter such as 'happiness' for different models of walking, etc.. Typical verb-adverb applications do not interact with objects; therefore, gauging their success is difficult. However, in applications with objects verb-adverb theory has been shown to be at least moderately successful[18]. This work attempts to apply verb-adverb theory to a more complicated (higher DOF) system, with fewer input trials, and with more robust output ability.

5.2.2. *Radial Basis Functions*

Radial Basis Functions produce for any point $\vec{p}$ [the adverb] a motion animation $m(\vec{p}, t)$ from some interpolation of input trials, or *exemplars*. We assume that every specific motion is some parametrization of a general motion function, and we do not know the function. Here, $\vec{p} = (O_x, O_y, O_z)$. We will derive a function for each joint in the robot model.

Radial basis functions have the form$R_i(d_i(\vec{p}))$, where d_i is some distance function in the parameter space. Since it is a function of only distances, there cannot be an affine translation of any kind - a linear or low-order polynomial representation is often added to the radial basis function to give the general baseline quality of motion. RBFs are compact, computationally efficient, able to interpolate sparse data, and can be evaluated on a continuous space. The initial RBF that we use here is $R(\vec{x}, \vec{x_i}) = r = |\vec{x} - \vec{x_i}|$, the standard distance function, known as the biharmonic spline.

Now we can generate the matrix of all Radial Basis Function influences over the points in parameter space,

$$D = [D_{ij}], \text{ where } D_{ij} = R_i(\vec{p}_j),$$

for all $i, j \in \{1, 2, \cdots N_e\}$, where N_e is the number of exemplars. The general solution to the error minimization problem is

$$\begin{pmatrix} D & T \\ T^T & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ c \end{pmatrix} = \begin{pmatrix} \vec{p} \\ 0 \end{pmatrix},$$

where D is as written above, T is the trend solution (linear approximation), and λ defines the coefficients for the purely radial solution.

5.3. *Application of Verb-Adverb Theory to Interpolation of Motions*

We apply the Radial Basis Function technique, to interpolate and extrapolate between known trials. To insure robustness, we apply few data points, and in unequal spacing. For every time frame, we take the parameters and known centers, and use

the radial basis function weighting to create the final output joint angle:

$$q_i(t) = \sum_{j=1}^{n} \alpha_{i,j} q_{i,n}(t) + \text{ an affine term}, \tag{5}$$

where $\alpha_{i,j}$ is the weight of the j^{th} trial for the i^{th} joint.

Figure [10] shows the error variation as we move the object location. Near any of the original trials, as expected, the error dissipates, but as we distance the object from the trial space [especially when we leave the convex hull of trials] the error increases dramatically, which causes task failure. Below a method is described that theoretically eliminates this error.

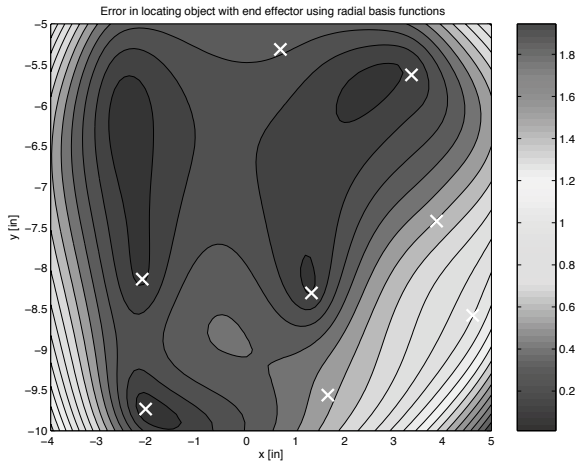

Fig. 10. A plot of the end effector error as it tries to locate a specific point in space, with the motion generated by radial basis function interpolation. White x's mark the spot in the x-y plane of a known trial [although with different z]. A value of z was chosen that was roughly the average of trial points. Near trial points, the error tends toward zero, and when we leave the convex hull of demo trials, the error rapidly increases.

5.4. *Improvements on Motion Interpolation and Extrapolation*

It is important to remind the reader that these trial motions were completed with no error. Since they were derived from animation data, they represent a precise task fulfillment. The error in the end effector comes from the non-linear averaging of joint angles - this averaging affects the end effector position non-linearly.

We are looking to position an end effector relative to an object; so, viewing motion in absolute space is not necessarily the most natural space in which to view the positioning problem. By viewing the trials in a new coordinate frame, we will find two improvements to the radial basis function solution: a method by which to

gauge what parts of an action are 'object oriented', and a method by which to reduce interpolation error *when we need to*, while the action involves object interation.

5.4.1. *Object Space*

There are two natural ways of looking at a system involving a robot and an object - from the perspective of the robot, and from the perspective of the object. Typically the reference frame of the object is ignored. However, it has several advantages.

In absolute space, all trials begin with the end effector in the same position, which slightly diverges on its path to the object. From the perspective of the object, the opposite is true. If we view a coordinate system created from an affine translation such that the object is at the origin, $(x', y', z') = (x - O_x, y - O_y, z - O_z)$, where (x', y', z') describes our new coordinate system, then all the trials start with the end effector in a different space, but they *converge* to the same space when nearing the object, and eventually come into contact with it at the origin. Figure [11] shows the comparison of end effectors traveling to a button in Leonardo's workspace, showing divergence in absolute space, and convergence in object space.

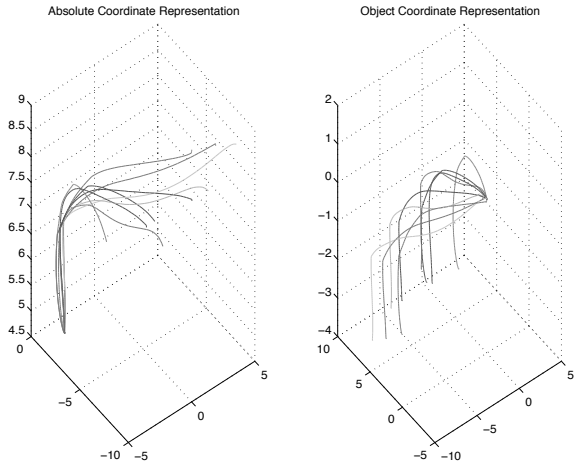

Fig. 11. A comparison of end effector trajectories in absolute coordinate space and the object workspace [the transformation of which changes from trial to trial]. Note the divergence of motion in absolute space, and convergence in object space, during a reach towards an object.

5.4.2. *Variances to show 'objectiveness'*

It is easy to note by a human, that when the motion is absolute (not related to the object), the variance in the absolute frame (σ_{abs}) is minimal, and maximal in the object frame (σ_{ob}). Similarly, when the motion is oriented to the object, the

variance in the absolute frame is maximal, but is minimal in the object frame.

This allows us to make a measure of how 'object oriented' any section of the motion is. A measure of the object-orientedness of the motion is found by a normalized fraction of absolute variance to object variance - if the absolute variance is lower then the motion is more absolute than object-oriented [and vice versa]. We let

$$\Upsilon(t) \equiv \frac{\sigma_{absolute}(t)}{\sigma_{absolute}(t) + \sigma_{object}(t)}$$

represent the *objectivity* of the action at any time t. Figure 12 shows a graph of a button reach, with the first plot displaying the object and absolute coordinate system variances, and the second plot showing Υ, a measure of the *objectivity* of that time in the action. Notice that, as expected, it provides a measure of an absolute action [because all presses begin in the same rest location] phasing into a object oriented action [because they all end up at the button].

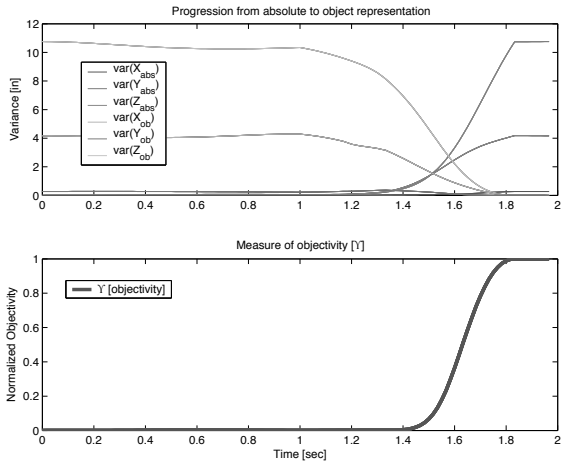

Fig. 12. The first plot shows a measure of variance in the absolute and object coordinate frames of a task. The second shows Υ, the *objectivity* of the action at time t, displaying a smooth transition from absolute representation to object representation.

5.4.3. *Objective Radial Basis Functions and Objective Motion Interpolation and Extrapolation*

Now we generate a new radial basis function, focused not on joint angles, but end effector motion; and not located in the absolute coordinate system, but the object representation space. This does not yield an explicit kinematic solution, but we will generate one afterwards.

In the object frame, all button grasps go toward the origin - thus, when interpolating between those motions, any interpolated motion will always go to the origin no matter where the object is located. With a RBF solution, not only the position, but movement of the object and angles of approach and reproach from objects will be accurately determined through the workspace. This does not occur in a typical RBF solution; it completely eliminates location errors in animation.

5.4.4. *Consequences of Objectivity and the* Υ *measure*

Figure [13] shows the end effector path generated by the use of objective radial basis functions, compared to the original radial basis function solution. Note that it proceeds from roughly the correct starting area [although not precisely] to exactly the object location area.

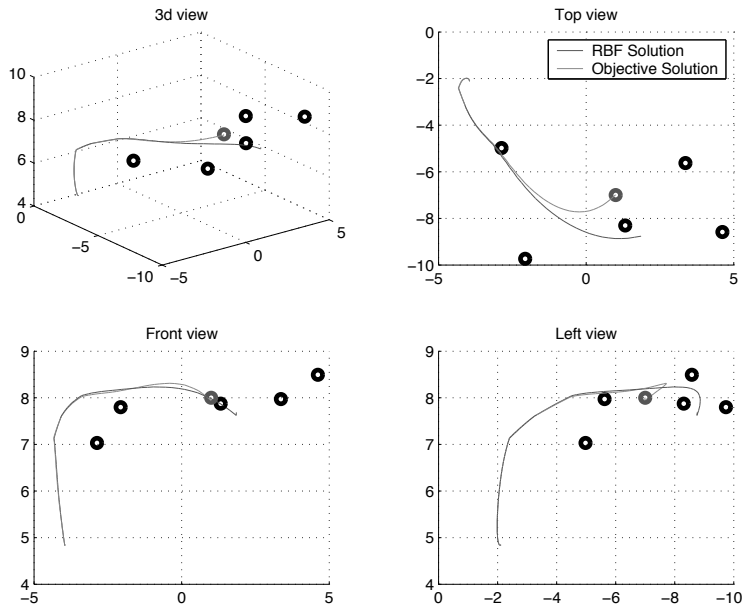

Fig. 13. A 3d plot of the end effector path from the radial basis function solution, alongside the plot of the end effector path from the objective radial basis function solution. Note that the latter path exactly reaches the desired object.

Clearly neither of these solutions are ideal. The first solution behaves more accurately in terms absolute behavior [in this case in the beginning of the motion], whereas the objective radial basis function solution is more precise when interacting with objects. Ideally we would like to blend between these solutions to form the overall optimal solution.

This is how we employ the objectivity factor Υ. This measure functions as a blending weight between the two solutions. Therefore, calling $\vec{x}_{absolute}(t)$ the original radial basis function solution, and $\vec{x}_{object}(t)$ the objective radial basis function solution, we obtain the full solution for the end effector,

$$\vec{x}_{final}(t) = \vec{x}_{absolute}(t) + \Upsilon(t)\left(\vec{x}_{object}(t) - \vec{x}_{absolute}(t)\right).$$

Once again, this solution does not yield joint angles, but merely an ideal overall path. The first half of the solution yields the desired quality of motion, and the second term, a correction to refine the end effector path, to remove all systematic error. Figure [14] shows this blended path in 3d for the button reaching task.

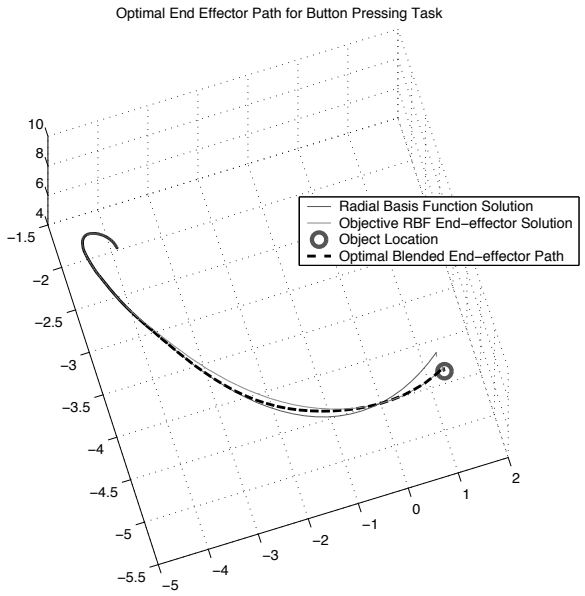

Fig. 14. A 3d plot of the final generated weighted path for the button reaching task. Note that this path initially follows the absolute coordinate system solution, and fades to the objective solution at the point in which the typical motions diverge.

5.4.5. *Inverse Kinematic Solution*

The aforementioned method provides the ideal path in task accomplishment [and is easily generalizable to provide precise path along with orientation, thus providing the entire 4x4 transformation matrix for the end effector at all times], in the sense of keeping the same quality and style of motion, while placing the end effector accurately when need be. Determining joint angles for this solution is still being researched - we take the initial joint angle solution as the starting point and use

an inverse Jacobian method [ie. $d\vec{q} = J^{-1}d\vec{x}$] to move the end effector differentially to the desired point $x_{final}(t)$. Employing this blindly, however, causes spurious motions, and energy minimization and path minimization IK techniques are being compared to improve quality of joint motion.

6. Conclusion

Overall, this technique, once combined with a better IK solution, will be able to deal with extremely complex interactions with objects, always retaining precision during times when precision with the object is needed, and maintaining quality of motion. Multiple interactions with an object during a trial, as well as the movement of an object during a trial, can all be accounted for in this scheme. Patterns of approach toward and object or subtle changes on the manner of interaction with an object [i.e. point of contact, etc] can be detected in this system, that are otherwise completely ignored in a pure radial basis function solution. By extending the use of this technique to allow rotations and translations for object frame of reference, for example in actions that involve grasping non-rotationally symmetric objects, this technique would allow modification of behaviors for successful completion. Furthermore, for cases such as a button press, where the initial tracking motion is dependent on object position [placement] but the pressing stage is univeral [downward, independent of object position], the system described above yields a solution that adequately gauges both stages of the motion, enabling robust accomplishment of these actions. In contrast to other currently employed techniques, this method is completely autonomous, and creates theoretically zero error in its ability to successfully interact with objects, in situations where high precision is necessary.

Acknowledgements

This work would not be possible for the contributions of many others. Stan Winston Studio provided the physical Leonardo robot, and Geoff Beatty and Ryan Kavanaugh provided the virtual model and facial animations. Special thanks to Matt Hancher for providing us with advice while developing the motor system and to Andrew Brooks for his assistance with the facial tracking software. This work is funded in part by a DARPA MARS grant and in part by the Digital Life and Things that Think consortia.

References

1. S. Schaal, Is Imitation Learning the Route to Humanoid Robots?, *Trends in Cognitive Science* **3**, 233-242 (1999).
2. Y. Kuniyoshi, M. Inaba, and H. Inoue, Learning by watching: Extracting reuseable task knowledge from visual observation of human performance. *IEEE Trans. Robotics Automation*, **10**, 799-822 (1994).
3. G. Hovland, P. Sikka, and B. Mc Carragher, Skill acquisition from human demonstration using a hidden Markov Model. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '96)*, 2706 - 2711. USA: IEEE (1996).

4. C. Atkeson, S. Schaal, Learning tasks from single demonstration. *IEEE International Conference on Robotics and Automation (ICRA 97)*, 1706-1712 IEEE (1997).
5. C. Atkeson, S. Schaal, Robot learning from demonstration. *International Conference on Machine Learning*, 12-20. 1997.
6. H. Miyamoto, S. Schaal, F. Gandolfo, H. Gomi, Y. Koike, R. Osu, E. Nakano Y. Wada, and M. Kawato, A Kendama learning robot based on bi-directional theory. *Neural Networks*, **9**, 1181-1302 1996.
7. H. Miyamoto, M. Kawato, A tennis serve and upswing learning robot based on bi-directional theory. *Neural Networks*, 11, 1131-1344 1998.
8. Ajo Fod, Maja Matarić, Odest Chadwicke Jenkins, Automated Derivation of Primitives for Movement Classification. *Autonomous Robots*, **12** (1) , 39-54 (2002).
9. Robert Platt Jr., Andrew Fagg, Roderic Grupen, Nullspace Composition of Control Laws for Grasping, *Proceedings of the International Conference on Intelligent Robots and Systems (IROS 02)*, 2002.
10. R. Platt Jr., A.H. Fagg, R.A. Grupen, Extending Fingertip Grasping to Whole Body Grasping. *Proceedings of International Conference on Robotics and Automation (ICRA 03)*, 2677-2682, 2003.
11. Richard Alan Peters II and Christina Campbell, Robonaut Task Learning through Teleoperation, *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, 2003.
12. Dare Baldwin, Jodie Baird, Discerning intentions in dynamic human action, *TRENDS in Cognitive Science*, **5**(4), 2001.
13. Maja Matarić, Sensory-Motor Primitives as a Basis for Imitation: Linking Perception to Action and Biology to Robotics. *Imitation in animals and artifacts*, 391-422, 2002.
14. Odest Chadwicke Jenkins, Maja Matarić, Automated Derivation of Behavior Vocabularies for Autonomous Humanoid Motion, *Autonomous Agents and Multi Agent Systems [in review]*, 2003.
15. Odest Chadwicke Jenkins and Maja Matarić, Deriving Action and Behavior Primitives from Human Motion Data, *In Proceedings of 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.
16. Joshua Tenenbaum, Vin de Silva, John Langford, A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science* **290**, 2000.
17. Peter I. Corke, Robotics Toolbox for Matlab (http://www.cat.csiro.au/cmst/staff/pic/robot).
18. R.A. Peters II and C. Campbell and R. Bodenheimer, Superposition of Behaviors Learned from Teleoperation. 2003.
19. C. Rose, B. Bodenheimer, M. Cohen, Verbs and Adverbs: Multidimensional Motion Interpolation Using Radial Basis Functions, *"Microsoft Research"*.