Int J Soc Robot (2012) 4:357–368
DOI 10.1007/s12369-012-0155-x

Learning and Recognition of Hybrid Manipulation Motions in Variable Environments Using Probabilistic Flow Tubes

Shuonan Dong · Brian Williams

Accepted: 31 May 2012 / Published online: 26 June 2012

Abstract For robots to work effectively with humans, they must learn and recognize activities that humans perform. We enable a robot to learn a library of activities from user demonstrations and use it to recognize an action performed by an operator in real time. Our contributions are threefold: (1) a novel probabilistic flow tube representation that can intuitively capture a wide range of motions and can be used to support compliant execution; (2) a method to identify the relevant features of a motion, and ensure that the learned representation preserves these features in new and unforeseen situations; (3) a fast incremental algorithm for recognizing user-performed motions using this representation. Our approach provides several capabilities beyond those of existing algorithms. First, we leverage temporal information to model motions that may exhibit non-Markovian characteristics. Second, our approach can identify parameters of a motion not explicitly specified by the user. Third, we model hybrid continuous and discrete motions in a unified representation that avoids abstracting out the continuous details of the data. Experimental results show a 49 % improvement over prior art in recognition rate for varying environments, and a 24 % improvement for a static environment, while maintaining average computing times for incremental recognition of less than half of human reaction time. We also demonstrate motion learning and recognition capabilities on real-world robot platforms.

Keywords Learning from demonstration · Motion learning · Real-time recognition · Flow tubes

1 Introduction

Continuous direct teleoperation of a complex robot can be tiring and difficult for an operator. Operator fatigue can lead to reduced precision during task execution [34]. We envision a robot that can recognize a teleoperator's intended motion and autonomously continue the execution of recognized routine tasks. To do this, the robot learns offline a library of generalized activities from a training set of user demonstrations. During online operations, the robot can recognize common teleoperated motions in real time, and if requested, autonomously execute the remainder of an activity. This involves determining the most likely motion in the learned activity library that the operator may be currently executing.

In many real-world motions, local state information is not sufficient to identify the motion. For example, correctly untying an anchor loop (Fig. 1) requires knowing whether the loop passes first around the left or right anchor. Locally, the motion looks the same in both cases: thus models with a Markovian assumption have difficulty distinguishing between the two possibilities. We instead choose a representation that reflects the temporal history of the entire motion.

Our approach provides three important features. First, since physical manipulation tasks are often non-Markovian, in that later parts of a motion may depend on past states, we use a model that can describe non-Markovian motions.

This research was supported by a National Defense Science and Engineering Graduate (NDSEG) Fellowship, 32 CFR 168a. Additional support was provided by a NASA JPL Strategic University Research Partnership.

S. Dong (✉)
362 Memorial Dr. #203, Cambridge, MA 02139, USA
e-mail: dongs@mit.edu

B. Williams
32 Vassar St. Room 32-227, Cambridge, MA 02139, USA
e-mail: williams@mit.edu

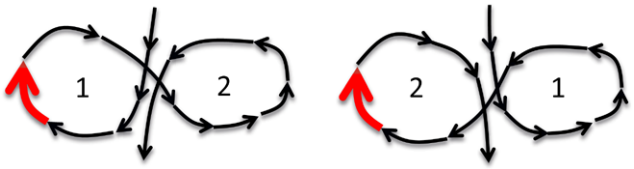

Fig. 1 Knowing only the previous state at the highlighted position will not distinguish the two motions

Second, our approach achieves real-time performance by efficiently sharing information across consecutive time steps. Third, our model of learned behaviors is robust to variations in initial environment states.

We model learned motions using probabilistic flow tubes (PFTs) [11, 12]. Constraint-based flow tubes have been used in planning and execution to represent sets of trajectories with common characteristics [16, 19]. In this context, a flow tube defines a state region where valid trajectories of a motion can be feasibly achieved given constraints on system dynamics. In motion learning, a *probabilistic* flow tube is computed by inferring the desired Gaussian state distribution at each time step from human demonstrations.

Geometrically, the width of a probabilistic flow tube represents flexibility in the robot's desired movement (Fig. 2, middle), enabling it to optimize additional performance criteria or recover from disturbances. The PFT representation produces humanlike trajectories because it directly models user demonstrations with minimal abstraction. In contrast, motions generated by planners may be unintuitive for human collaborators, even though they may be valid. Furthermore, the PFT representation is easily applied to situations with different initial conditions because it is parameterized by the relevant variables of the motion.

Using these learned motion models, we perform recognition of a new partial motion by computing its likelihood of being described by each model, after temporally aligning relevant time steps. Real-time performance is achieved by storing parts of the computation in memory and using an incremental version of dynamic time warping [24] for temporal matching.

In the remainder of this paper, we review related work, describe the input data used in our approach, formalize the problem, present our algorithms for offline learning and online recognition, and compare our results to prior art.

2 Related Work

Learning and recognizing human motions have long been an interest for human-robot interaction [4, 22, 30]. For example, learning human-taught policies has proven useful in the domains of underactuated pendulum control [5], autonomous helicopters [9], and vehicle navigation [1]. We are interested in manipulation tasks, where interaction with objects in the environment becomes important. Our approach is inspired by existing work in learning manipulation tasks from demonstration.

Peters and Campbell [28] taught the humanoid Robonaut to grasp a tool by time-normalizing and averaging demonstrated motions. Their work showed that learning from teleoperated demonstrations is an attractive approach to controlling complex robots. With a more robust motion representation and better adaptability to new situations, this type of approach will become compelling for a wider range of applications.

Researchers at USC [15, 27] modeled learned motions as a spring system described by differential equations with parameterized start and goal locations. In contrast, Mühlig et al. [21] chose to abstract the demonstrated time-series data in task space with Gaussian Mixture Models. Calinon et al. [7] modeled motions as a mixture of Gaussian Mixture Models and Bernoulli distributions that captures both spatial and temporal aspects of a motion. Inspired by these approaches, our work uses a representation that can faithfully capture important features of a human's demonstrated motion, without prior knowledge of the motion's distinguishing characteristics.

Several researchers have considered different models of motion characteristics while approaching different problems. Cederborg et al. [8] considers three reference frames in which a motion can be performed: relative to the starting position, relative to the robot frame, and relative to an object position. Alissandrakis et al. [3] considers a set of five motion characteristics: relative displacement, absolute position, relative position, rotation, and orientation. Our approach uses a similar set of motion characteristic candidates.

Motion recognition applications have ranged from visual gesture recognition [23, 35] to understanding domestic activities [14] to gait analysis [13], among others. We are mainly interested in learning and recognizing teleoperated manipulation tasks, though our approach is extensible to other applications.

Motion learning or recognition have commonly been explored using Hidden Markov Models (HMMs) [6, 20, 33, 36]. For example, recent work by Calinon et al. [6] learn HMM models of the motions using the Expectation Maximization algorithm, and employ Gaussian Mixture Regression to compute desired velocities for autonomous motion generation. Similarly, Martin et al. [20] model motions learned from training data as sequences of HMM states, where each state refers to a mixture of Gaussians, and recognition is performed using either the Viterbi algorithm or posterior probabilities during model learning. Their approach proved promising in recognizing grasping tasks and provides a good basis of comparison.

Lee and Ott [18] introduce a "refinement tube" generated from a sequence of GMMs drawn from a learned HMM, which they use to limit accidental disturbances when augmenting whole-body imitation learning with local kinesthetic teaching. By contrast, we use a distinctly different probabilistic flow tube representation to directly model user-demonstrated motions for learning and recognition.

Regression using Gaussian processes [29] differs from our approach in that it treats all demonstration data points independently in time. In contrast, we are interested in maintaining temporal ordering of the data points in the motion while using Gaussian distributions to describe spatial variability.

3 Problem Statement

Figure 2 illustrates an example learning and recognition problem. This 2D world contains three movable objects: "box", "ball", and "bin". The user gives several demonstrations each of three different motion types: "move box to bin", "move box left", and "move box home", where "home" refers to the center of the environment. Given a new environment with different object locations, the offline learning algorithm computes a probabilistic flow tube for each motion type. As the user begins a motion in this environment, the recognition algorithm determines in real time the likelihood that the user is executing each type of motion.

In later experiments, we collect human motion data directly using a teleoperated or kinesthetic teaching interface, where human-driven motion is recorded through robot

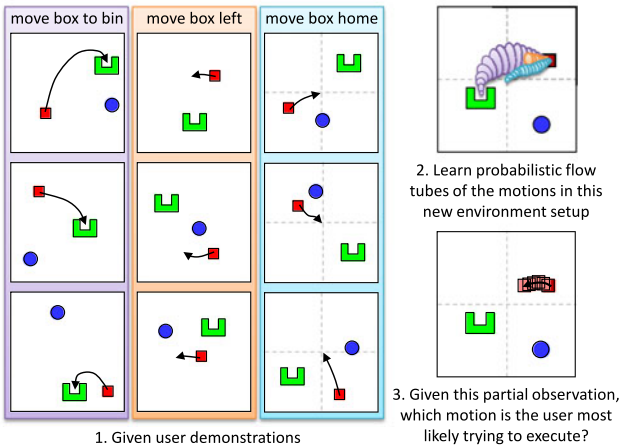

Fig. 2 Illustrated example of a learning and recognition problem. Initially, a user demonstrates a set of motions a few times. The learning problem consists of generalizing the demonstrations into a probabilistic flow tube representation. Given a new user motion, the recognition problem consists of determining which motion the user is performing

poses. The input data can have a hybrid mix of continuous and discrete variables. Each observed user demonstrated training sequence is recorded as $T = \langle \mathbf{C}, \mathbf{D}, \mathbf{P}, \mathbf{Q} \rangle$, where:

- $\mathbf{C}$ is a set of c single dimensional continuous variables at time steps $t = 0, \ldots, N$. Examples of such continuous variables include execution time, temperature, voltage, etc.
- $\mathbf{D}$ is a set of d discrete variables at time steps $t = 0, \ldots, N$. Examples of such discrete variables include gripper open/close, power on/off, etc.
- $\mathbf{P}$ is the set of Cartesian position variables x, y, z for each of $b = 1, \ldots, B$ points of interest at time steps $t = 0, \ldots, N$. $\mathbf{P}^{eff}$ denotes the position variables of the specific point of interest where $b = eff$, the robot end effector. The index eff is typically equal to 1.
- $\mathbf{Q}$ is the set of quaternion orientation variables q_1, q_2, q_3, q_4 for each of B points of interests at time steps $t = 0, \ldots, N$. Similarly, $\mathbf{Q}^{eff}$ refers to the orientation variables of the robot end effector.

Points of interest in the environment can be the robot end effector, objects or parts of objects that can be sensed, or other known markers in the environment. Future extensions of the learning algorithm may also utilize velocities $\mathcal{V}$ and accelerations $\mathcal{A}$ for each point of interest over time. The manipulation motions studied here are generally slow enough that position information alone is sufficient for good motion learning performance.

We use $\mathcal{S} = \{T_k\}_{k=1..K}$ to represent a set of K training sequences for a particular motion, and $\boldsymbol{\mathcal{T}} = \{\mathcal{S}_\ell\}_{\ell \in L}$ to represent the combined set of training sequences for all M motions with labels $L = \{\ell_1, \ldots, \ell_M\}$. The lengths of training sequences N_k may vary among different sequences.

The input to the real-time motion learning and recognition problem is a tuple $\langle \boldsymbol{\mathcal{T}}, L, T^{curr} \rangle$, where:

- $\boldsymbol{\mathcal{T}} = \{\mathcal{S}_\ell\}_{\ell \in L}$, where each $\mathcal{S}_\ell = \{T_k\}_{k=1..K}$ is a set of K training sequences capturing the environment states of B points of interest over time $t = 0, \ldots, N$ for motion $\ell \in L$. The initial states $T_k(0)$ and lengths N_k of the training sequences can be different across the different demonstration trials.
- $L = \{\ell_1, \ldots, \ell_M\}$ is the set of labels of all M learned motions.
- T^{curr} is a user's current unlabeled partial execution of a motion from $t = 0$ to $t = curr$.

To learn a generalization of a particular motion ℓ, the motion learning problem uses the inputs $\mathcal{S}_\ell$ and $T^{curr}(0)$, a set of training sequences for motion ℓ and the current state of the environment, respectively, and generates a probabilistic flow tube $pft = \langle T^{eff}, \Sigma^{eff} \rangle$, where $T^{eff} = \langle \mathbf{C}, \mathbf{D}, \mathbf{P}^{eff}, \mathbf{Q}^{eff} \rangle$ refers to a nominal desired robot end effector trajectory and $\Sigma^{eff} = \langle \boldsymbol{\sigma}_\mathbf{C}, \boldsymbol{\sigma}_\mathbf{D}, \Sigma^{eff}_\mathbf{P}, \Sigma^{eff}_\mathbf{Q} \rangle$ refers to the corresponding covariances throughout the trajectory.

After a probabilistic flow tube generalization is learned for each motion offline, the online motion recognition problem observes a user's current partial execution T^{curr}, and generates in real time a set of log probabilities $\mathbf{LL} = \{LL_1, \ldots, LL_M\}$ over the set of known labels $L = \{\ell_1, \ldots, \ell_M\}$ that reflect the likelihood that the label of the user's current motion is one of L.

We approach the recognition problem in two parts: learning models offline for each motion given a new environment, and recognizing the most likely motion an operator is performing in real time as the movement progresses.

4 Offline Motion Learning Approach

Algorithm 1 generates PFTs describing learned motions in a new environment. First, for each labeled motion, the algorithm determines the important features or relations in the demonstrations, which we call *motion variables* $\mathcal{F}$ (line 2). Then it uses the training sequences $\mathcal{S}$ to create a probabilistic flow tube defined as $\langle T^{eff}, \Sigma^{eff} \rangle$ obeying the same relations $\mathcal{F}$ in the new environment $T(0)$ (line 3). In lines 4–5, we augment the PFTs with additional values used during recognition (Sect. 5).

4.1 Motion Variable Identification

A key feature of our learning system is the ability to autonomously determine what features or relations, if any, are characteristic of a particular demonstrated motion. We use the general term *motion variables* to describe the class of potentially important features of a motion. Of these, the *relevant* motion variables are those preserved over different demonstrated trials of that motion, while other motion variables may vary due to changes in the environment or the user's movement.

For example, in the motion "move box to bin", the robot end effector starts at the box and ends at the bin. The system will learn that the displacement between the robot effector and the box is a relevant motion variable at the beginning of the motion, and that the displacement between the robot effector and the bin is a relevant motion variable at the end of the motion. The system will also learn that the positions of any other objects known in the environment are not relevant to this motion.

For each of the input variables $\mathbf{C}, \mathbf{D}, \mathbf{P}, \mathbf{Q}$, we consider the following modes for candidate motion variables (Fig. 3) that are preserved across all demonstrations: (1) *absStart*: value at start; (2) *absEnd*: value at end; (3) *relInit*: offset from start to end; (4) *relEffStart*: offset from robot end effector at start; (5) *relEffEnd*: offset from robot end effort at end. Since the continuous and discrete variables $\mathbf{C}$ and $\mathbf{D}$ (such as time and power on/off) are independent of the points of interest (such as objects) in an environment, they are not considered in the *relEffStart* and *relEffEnd* modes.

We chose this particular set of candidate motion variables based on commonalities observed among many practical robot manipulation tasks. A more thorough study of other possible modes may be warranted for other task domains. The learning approach described here remains applicable for additional types of motion variables.

In our implementation, motion variables are determined using endpoints of motions; however, our approach generalizes trivially to additional motion variables. For instance,

Algorithm 1 OFFLINEMODELLEARNING($\boldsymbol{\mathcal{T}}, L, T(0)$)

Input:
$\boldsymbol{\mathcal{T}} = \{\mathcal{S}_\ell\}_{\ell \in L}$; $\mathcal{S}_\ell$ is training set for motion $\ell \in L$
L, set of labels of all learned motions
$T(0)$, a new environment state

Output:
PFT, set of augmented probabilistic flow tubes

Notable local variables:
$\mathcal{F}$, set of relevant motion variables
$T^{eff} = \langle \mathbf{C}, \mathbf{D}, \mathbf{P}^{eff}, \mathbf{Q}^{eff} \rangle$, PFT end effector trajectory
$\Sigma^{eff} = \langle \boldsymbol{\sigma}_\mathbf{C}, \boldsymbol{\sigma}_\mathbf{D}, \Sigma^{eff}_\mathbf{P}, \Sigma^{eff}_\mathbf{Q} \rangle$, PFT covariance trajectory

1: **for** $\ell \in L$ **do**
2: $\quad \mathcal{F} \leftarrow$ IDENTIFYMOTIONVARIABLES($\mathcal{S}_\ell$)
3: $\quad \langle T^{eff}, \Sigma^{eff} \rangle \leftarrow$ MAKEPFT($\mathcal{S}_\ell, \mathcal{F}, T(0)$)
4: $\quad \mathbf{I} = \left\{ \Sigma^{eff}(n)^{-1} \right\}_{n=1..N_\ell}$
5: $\quad \mathbf{G} = \left\{ -\log\left((2\pi)^{\frac{\dim(\mathbf{T}^{eff})}{2}} \left| \Sigma^{eff}(n) \right|^{\frac{1}{2}} \right) \right\}_{n=1..N_\ell}$
6: $\quad \text{PFT}_\ell = \left\langle T^{eff}, \Sigma^{eff}, \mathbf{I}, \mathbf{G} \right\rangle$

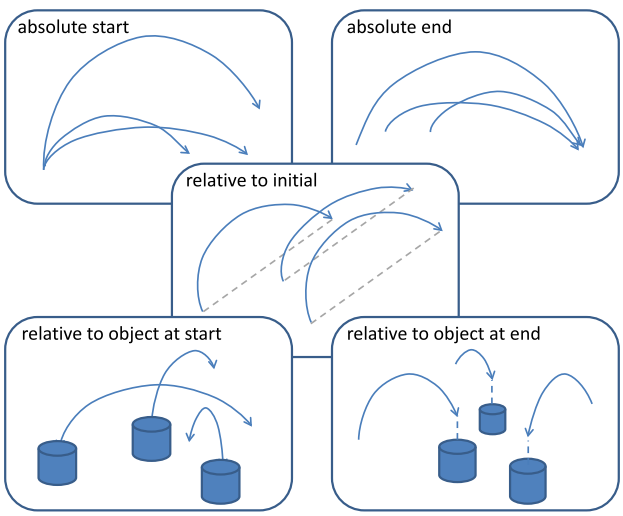

Fig. 3 Example illustrations of five possible ways that motion variables can be relevant. *Arrows* refer to the robot end effector trajectories

Algorithm 2 IDENTIFYMOTIONVARS($\mathcal{S}$)

Input:

$\mathcal{S}$, set of K demonstrated sequences $\{T_k\}_{k=1..K}$, where $T = \langle \mathbf{C}, \mathbf{D}, \mathbf{P}, \mathbf{Q} \rangle$

Output:

$\mathcal{F}$, tuple of features $\langle F_{\mathbf{C}}, F_{\mathbf{D}}, F_{\mathbf{P}}, F_{\mathbf{Q}} \rangle$

Notable local variables:

$\mathcal{M} = \{absStart, absEnd, relInit, relEffStart, relEffEnd\}$

$\mathbf{q}_k^{\overrightarrow{b1}} = \text{TOQUATERNION}(\mathbf{P}_k^b - \mathbf{P}_k^{eff})$, orientation of ray from object b to robot end effector for trial k

ϵ, small ratio to determine relevant motion variables

1: **for** $\mathbf{X} \in \{\mathbf{C}, \mathbf{D}, \mathbf{P}^{eff}, \mathbf{Q}^{eff}\}$ **do**

2: $\langle \boldsymbol{\mu}, \boldsymbol{\Sigma} \rangle_{\mathbf{X}} \left\{ \begin{array}{l} absStart \\ absEnd \\ relInit \end{array} \right\}$

$\leftarrow \text{FITGAUSS} \left\{ \begin{array}{l} \{\mathbf{X}_k(0)\}_{k=1..K} \\ \{\mathbf{X}_k(N_k)\}_{k=1..K} \\ \{\mathbf{X}_k(N_k) - \mathbf{X}_k(0)\}_{k=1..K} \end{array} \right\}$

3: $\langle \boldsymbol{\mu}, \boldsymbol{\Sigma} \rangle_{\mathbf{P}} \left\{ \begin{array}{l} relEffStart \\ relEffEnd \end{array} \right\}$

$\leftarrow \text{FITGAUSS} \left\{ \begin{array}{l} \{\mathbf{P}_k^b(0) - \mathbf{P}_k^{eff}(0)\}_{k=1..K}^{b=2..B} \\ \{\mathbf{P}_k^b(N_k) - \mathbf{P}_k^{eff}(N_k)\}_{k=1..K}^{b=2..B} \end{array} \right\}$

4: $\langle \boldsymbol{\mu}, \boldsymbol{\Sigma} \rangle_{\mathbf{Q}} \left\{ \begin{array}{l} relEffStart \\ relEffEnd \end{array} \right\}$

$\leftarrow \text{FITGAUSS} \left\{ \begin{array}{l} \{(\mathbf{Q}_k^b(0))^{-1} \cdot \mathbf{q}_k^{\overrightarrow{b1}}(0)\}_{k=1..K}^{b=2..B} \\ \{(\mathbf{Q}_k^b(N_k))^{-1} \cdot \mathbf{q}_k^{\overrightarrow{b1}}(N_k)\}_{k=1..K}^{b=2..B} \end{array} \right\}$

5: **for** $\mathbf{X} \in \{\mathbf{C}, \mathbf{D}, \mathbf{P}, \mathbf{Q}\}$ and $mode \in \mathcal{M}$ **do**

6: $\mathbf{relevant}_{\mathbf{X}}^{mode} = 0$

7: **if** $\max(\text{eig}(\boldsymbol{\Sigma}_{\mathbf{X}}^{mode})) < \epsilon \cdot \max(\text{range}(\mathbf{X}^{mode}))$ **then**

8: $\mathbf{relevant}_{\mathbf{X}}^{mode} = 1$

9: $F_{\mathbf{X}} = \langle \boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{relevant} \rangle_{\mathbf{X}}^{mode \in \mathcal{M}}$

users can indicate additional time points during compound motions using keyframes [2], which can be accommodated in our implementation by segmenting the motions at these points. In the future, we plan to automate such segmentation based on qualitative changes in the motion. In this paper, we consider demonstrated motions consisting of a single segment.

To determine relevant motion variables, we cluster the training samples for each candidate variable $\mathbf{X} \in \{\mathbf{C}, \mathbf{D}, \mathbf{P}, \mathbf{Q}\}$ for each mode (Algorithm 2). We fit a Gaussian $\langle \boldsymbol{\mu}, \boldsymbol{\Sigma} \rangle$ on each sample set (lines 1 to 4), and consider relevant those variables with a narrow spread compared to the range of the motion (lines 5 to 9).

Algorithm 3 MAKEPFT($\mathcal{S}, \mathcal{F}, T(0)$)

Input:

$\mathcal{S}$, set of K demonstrated sequences $\{T_k\}_{k=1..K}$, where $T = \langle \mathbf{C}, \mathbf{D}, \mathbf{P}, \mathbf{Q} \rangle$

$\mathcal{F}$, tuple of features $\langle F_{\mathbf{C}}, F_{\mathbf{D}}, F_{\mathbf{P}}, F_{\mathbf{Q}} \rangle$

$T(0)$, a new environment state

Output:

$T^{eff} = \langle \mathbf{C}, \mathbf{D}, \mathbf{P}^{eff}, \mathbf{Q}^{eff} \rangle$, robot trajectory

Σ^{eff}, covariances at each corresponding time step

Notable local variables:

$\mathbf{w}$, temporal matching indexes for two trajectories

1: $T'(0) \leftarrow \text{EXTRACTRELEVANTOBJECTS}(T(0), \mathcal{F})$

2: $\mathcal{S}' \subseteq \mathcal{S} \leftarrow \text{SIMILARINITCONDSEQS}(\mathcal{S}, T'(0))$

3: $\mathcal{S}^{norm} \leftarrow \text{NORMALIZESCALEROTATE}(\mathcal{S}', \mathcal{F}, T'(0))$

4: $T^{eff} = T_1^{norm}$, where $T_k^{norm} \in \mathcal{S}^{norm}$

5: **for** $k = 2$ to K **do**

6: $\mathbf{w} \leftarrow \text{FASTDTW}([\mathbf{P}, \mathbf{Q}]^{eff}, [\mathbf{P}, \mathbf{Q}]_k^{norm})$

7: $T^{eff} \leftarrow \frac{1}{k}[(k-1)T_{eff}(\mathbf{w}_{:,1}) + T_k^{norm}(\mathbf{w}_{:,2})]$

8: **for** $k = 1$ to K **do**

9: $\mathbf{w} \leftarrow \text{FASTDTW}([\mathbf{P}, \mathbf{Q}]^{eff}, [\mathbf{P}, \mathbf{Q}]_k^{norm})$

10: $T_k^{interp} \leftarrow \text{INTERPOLATE}(T_k^{norm}(\mathbf{w}_{:,2}), |T^{eff}|)$

11: **for** $n = 1$ to $|T^{eff}|$ **do**

12: $\Sigma^{eff}(n) \leftarrow \text{COVARIANCE}\{T_k^{interp}(n) : k = 1..K\}$

4.2 Flow Tube Generation

Algorithm 3 generates the PFT for a motion in a new environment based on training demonstrations, using the relevant motion variables determined in Sect. 2. The major steps are illustrated in Fig. 4.

We first determine the values in the new environment of initial states that are components of relevant motion variables (line 1), and a subset of demonstrations for which these relevant initial states are most similar to those of the new environment (line 2). The similarity measure will depend on the problem domain. These selected demonstration sequences are transformed by scaling, translation, and rotation so that the values of relevant motion variables involving the start and end poses of the robot end effector match those required in the new environment (line 3).

The spatially transformed sequences are temporally aligned by dynamic time warping (DTW) [24, 32]. We use a fast multiresolution DTW algorithm [31] to achieve linear time and space complexity. We compute a representative mean sequence using an iterative procedure (lines 4–7) since DTW only handles two trajectories at a time. This is the output trajectory sequence of the robot end effector T^{eff}.

The demonstrated sequences may have different numbers of data points, so we use DTW again to temporally match each of the normalized demonstrated sequences in $\mathcal{S}^{norm}$ to the mean sequence T^{eff}, and interpolate so that all

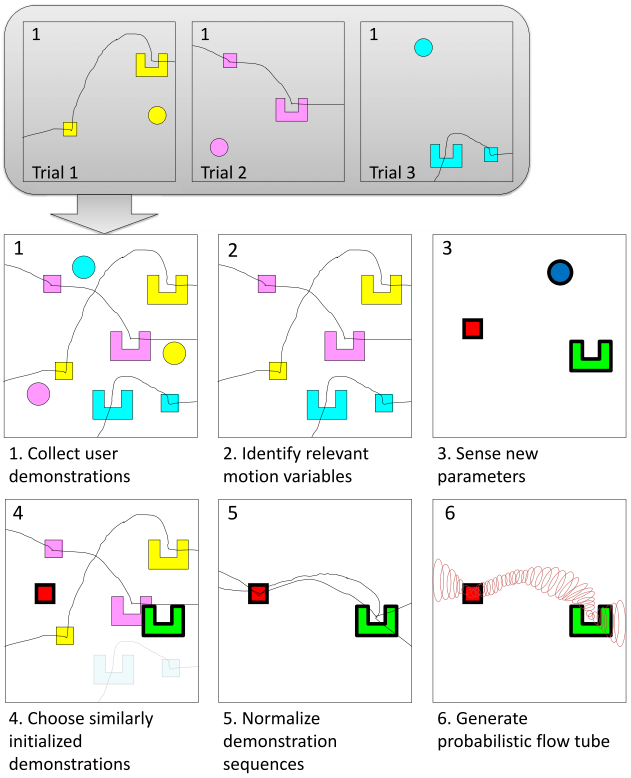

Fig. 4 Steps of flow tube generation from three demonstrations of "move box to bin" in our 2D simulation environment

have the same number of data entries (lines 8–10). Last, we compute covariances at each corresponding time step across the temporally matched normalized demonstrated sequences (lines 11–12).

4.3 Application to Autonomous Execution

We demonstrated our learning algorithm on the PR2 robot developed by Willow Garage, as part of the Learning from Demonstration (LfD) Challenge at the 2011 AAAI conference. Prior to the conference, we performed user teaching and preliminary testing through the Bosch remote lab facility [25]. We used the PR2's onboard sensing and the Robot Operating System (ROS)'s object recognition software to record the environment states throughout the demonstrations.

The PR2 faced a table holding certain objects. Motions were taught kinesthetically by manually moving the right arm of the robot. During execution, learned objects could be optionally replaced by user-specified objects. Users could also optionally instruct the robot to use its left arm to complete the task, which was accomplished by inverting the sign of the y position and q_w, q_y quaternion values during execution.

We tested several different motions, including: "move left", "move up and over", "go home", "pour into", "pour done", "reach", "put on table", "shake", and "stir". Five demonstrations of each motion were provided. Some additional helper commands for object detection and gripping were handled independently through ROS. Fig. 5 compares the learned motions with the original user demonstrations. During autonomous execution, the environment contained one object (an odwalla bottle). For each demonstration, the object and robot end effector (measured at the PR2 wrist roll link) were set to begin with the same arbitrary poses.

The motion learning algorithm successfully generated motions reflecting the character of the training motions, given just a few demonstrations. The computation time for offline PFT learning for each motion ranged between 0.9 seconds and 1.3 seconds for user inputs sampled at about 100 data points each.

A few points deserve particular note. The "move left" motion was intended to have $\mathbf{P}^{relInit}$ as a relevant motion variable, but the algorithm determined there was too much variation in the difference of beginning and end positions. Nonetheless, the resulting learned motion still closely resembled the user inputs. The motion "go home" moved the robot end effector to a raised arm tucked position. The algorithm correctly identified that the end position and orientation were the same across all demonstrations.

The "pour into", "pour done", and "reach" motions were demonstrated with an object in the environment. The algorithm recognized that the robot end effector was placed similarly over the object at the end of each "pour into" motion, and likewise at the beginning of each "pour done" motion. In the "pour done" example in Fig. 5, although the effector's initial pose is not over the object, it is immediately moved there before executing the motion.

The "shake" motion was a quick up-down-up-down activity inspired by the movement used to shake an orange juice bottle. The "stir" motion involved making five complete circular movements. Interestingly, the algorithm detected an unintended relation between the beginning and end positions of the "stir" motion as a relevant motion variable in the demonstrations; however, because the demonstrated motions lay in slighly different planes, this relation overconstrained the generated motion to a small vibration around a point. This example illustrates that more demonstrations may be necessary to fully capture the variability in a desired motion.

5 Online Motion Recognition Approach

Algorithm 4 performs real-time recognition based on a set of PFTs generated by Algorithm 1. The auxiliary parameters $\mathcal{W}$ are initially empty and continuously updated along with the current trajectory T^{curr} as execution proceeds.

The three main components of the recognition approach are: determine the point in each PFT to which the current partial motion corresponds, temporally align the identified

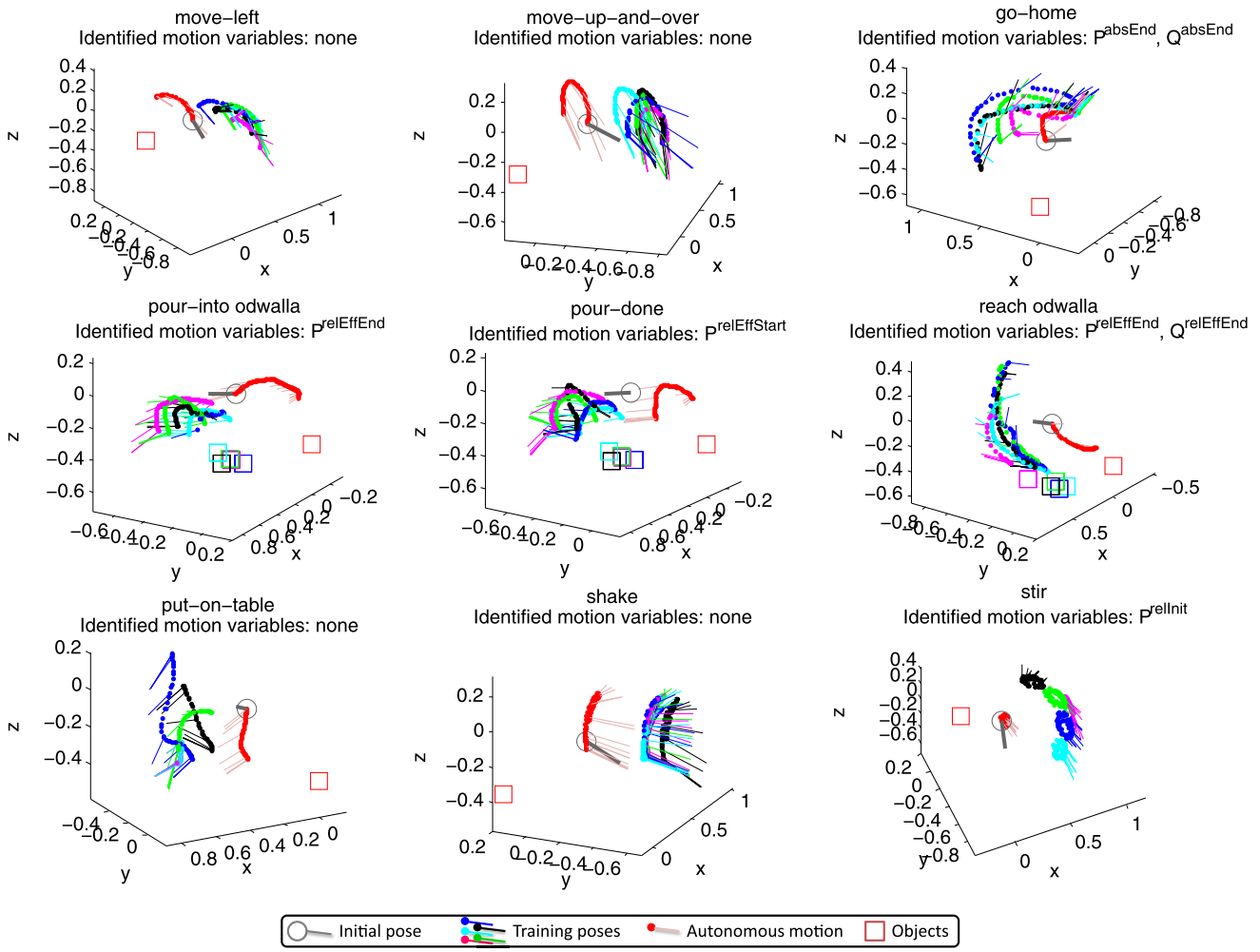

Fig. 5 Results of learning from 5 demonstrations of each motion (Color figure online)

portion of the PFT with the current partial motion, and compute the log likelihood that the current partial motion is recognized as each PFT.

To determine the location in a PFT that best corresponds to the current executed state (lines 2 to 6 in Algorithm 4), we consider both the distance between the current executed state and the PFT, and how much time has passed in the execution as compared with the trained models. Intuitively, the point in the PFT that best corresponds to the current executed state should be spatially close to it while having been executed at around the same time, as illustrated in Fig. 6.

In Algorithm 4, we first compute the distances $\mathbf{d}$ from the current position and orientation in $T^{curr}(N^{curr})$ to those in the nominal trajectory T^{eff} for motion ℓ through all the time steps in the PFT in line 2. Small values in $\mathbf{d}$ will help indicate which time steps in the PFT correspond to the current executed state.

Next, we represent how temporally different the current execution time t^{curr} is from the points in the temporal component of the PFT, or $\langle \mathbf{C}^{time}, \boldsymbol{\sigma}_{\mathbf{C}}^{time} \rangle$, by evaluating the probability density $\mathbf{p}$ of the current time at each point in the tube in line 3. We weight the distances by the temporal similarity measure to obtain $\mathbf{d}' = \{\frac{\mathbf{d}_n}{\mathbf{p}_n}\}_{n=1,\ldots,N_\ell}$. The time step in the PFT that corresponds to the current executed state occurs when the weighted distance is smallest.

We chose to use the actual distances between the current state and the points in the nominal PFT trajectory (i.e., $\|T^{curr}(N^{curr}) - T^{eff}(n)\|$) to represent spatial consistency instead of computing the spatial probability densities of the current state evaluated through all the Gaussians in the PFT (i.e., $\mathcal{N}(T^{eff}(n)\Sigma^{eff}(n))|_{T^{curr}(N^{curr})}$) because the distance is much faster to compute than probability density. While spatial probability densities give a more accurate estimate of a point's deviation from the flow tube and are more appropriate to combine with the temporal similarity measures, they take longer to compute due to the higher dimensionality of spatial states. We have found direct distances to be good estimates of spatial consistency for motions with flow tube widths that do not vary greatly with high frequency, as is true in most robotic tasks.

The next step in our recognition approach is to temporally align the identified portion of each PFT to the current execution trajectory in order to compute likelihoods. While we could use basic dynamic time warping to perform temporal matching in the recognition problem, we note that the algorithm would recompute similar cost matrices each time the test motion progresses. To leverage as much information as possible from one time step to the next, we perform incremental dynamic time warping [10] by keeping previously computed cost and back pointer matrices in memory and updating as necessary.

Algorithm 4 ONLINERECOGN(PFT, L, T^{curr}, $\mathcal{W}$)

Input:

PFT $= \langle T^{eff}, \Sigma^{eff}, \mathbf{I}, \mathbf{G} \rangle$, where
$T^{eff} = \langle \mathbf{C}, \mathbf{D}, \mathbf{P}^{eff}, \mathbf{Q}^{eff} \rangle$
$\Sigma^{eff} = \langle \boldsymbol{\sigma}_{\mathbf{C}}, \boldsymbol{\sigma}_{\mathbf{D}}, \Sigma^{eff}_{\mathbf{P}}, \Sigma^{eff}_{\mathbf{Q}} \rangle$
L, set of labels of all learned motions
T^{curr}, current observed trajectory
$\mathcal{W}$, cost matrices for dynamic time warping

Output:

LL, set of log likelihoods for each motion in L
$\mathcal{W}$, updated cost matrices for DTW

Notable local variables:

$\mathbf{C}^{time}$, $\boldsymbol{\sigma}^{time}_{\mathbf{C}}$, temporal component of $\mathbf{C}$, $\boldsymbol{\sigma}_{\mathbf{C}}$ in PFT
$\mathbf{d}$, spatial distance between a PFT and current state
$\mathbf{p}$, probability densities evaluated at time steps in PFT
N_ℓ, length of flow tube ℓ
N^{curr}, length of current trajectory T^{curr}
t^{curr}, time at current position $T^{curr}(N^{curr})$
$\mathbf{w}$, temporal matching indexes for two trajectories
π_ℓ, prior log likelihood of flow tube ℓ

1: **for** $\ell \in L$ **do**
2: $\quad \mathbf{d} = \{\|[\mathbf{P}, \mathbf{Q}]^{curr}(N^{curr}) - [\mathbf{P}, \mathbf{Q}]^{eff}_\ell(n)\|\}_{n=1,\ldots,N_\ell}$
3: $\quad \mathbf{p} = \{\mathcal{N}(\mathbf{C}^{time}_\ell(n), \boldsymbol{\sigma}^{time}_{\mathbf{C}_\ell}(n))|_{t^{curr}}\}_{n=1,\ldots,N_\ell}$
4: $\quad \mathbf{p} \leftarrow \frac{\mathbf{p}}{\max(\mathbf{p})}$
5: $\quad \mathbf{d}' = \{\frac{\mathbf{d}_n}{\mathbf{p}_n}\}_{n=1,\ldots,N_\ell}$
6: $\quad n^* = \arg\min_n(\mathbf{d}')$
7: $\quad \text{PFT}^*_\ell = \text{PFT}_\ell(1, \ldots, n^*)$
8: $\quad \langle \mathbf{w}, \mathcal{W}_\ell \rangle \leftarrow \text{INCREMENTDTW}(T^{eff*}_\ell, T^{curr}, \mathcal{W}_\ell)$
9: $\quad LL_\ell = \pi_\ell + \frac{1}{|\mathbf{w}|} \sum_{j=1}^{|\mathbf{w}|} (\mathbf{G}_\ell(\mathbf{w}_{j,1}) - \frac{1}{2}\boldsymbol{\delta}^T \mathbf{I}_\ell(\mathbf{w}_{j,1})\boldsymbol{\delta})$
$\quad$ where $\pi_\ell = \log(p_{\mathcal{L}}(\ell))$
$\quad$ and $\boldsymbol{\delta} = T^{curr}(\mathbf{w}_{j,2}) - T^{eff}_\ell(\mathbf{w}_{j,1})$

The INCREMENTALDTW algorithm used in line 8 of ONLINERECOGN accepts two input trajectories of lengths M and N, respectively, and an input parameter $\mathcal{W} = \langle \mathbf{c}, \mathbf{C}, \mathbf{b} \rangle$, where $\mathbf{c}$, $\mathbf{C}$, and $\mathbf{b}$ are the $m \times n$ cost matrix, cumulative cost matrix, and back pointer matrix, respectively, computed at the previous time step. Incremental DTW reuses these matrices when temporally aligning the two trajectories by computing new values only for rows $m+1$ to M, and columns $n+1$ to N. A new temporal matching is found by tracing the updated back pointer matrix from the end to the beginning. The temporal alignment is represented as a two column matrix $\mathbf{w}$ of corresponding indexes of the two input trajectories.

Finally, after temporally matching the current partial trajectory with the corresponding portions of each probabilistic flow tube, we can proceed to compute the likelihood that the current trajectory belongs to a particular modeled motion. Formally, we define random variables $\mathcal{L}$ and $\mathcal{O}$ to represent motion labels and observations, respectively. The probability that the motion is ℓ given the current observation sequence T^{curr} is $p_{\mathcal{L}|\mathcal{O}}(\ell \mid T^{curr})$, and applying Bayes Rule gives $p_{\mathcal{L}|\mathcal{O}}(\ell \mid T^{curr}) \propto p_{\mathcal{O}|\mathcal{L}}(T^{curr} \mid \ell) p_{\mathcal{L}}(\ell)$. We obtain the prior probability of each motion label $p_{\mathcal{L}}(\ell)$ by recording the number of times the motion was used during training weighted by how far along the current motion is in the flow tube, i.e. $p_{\mathcal{L}}(\ell) = \frac{\#_\ell}{\#_{all}} \frac{n^*_\ell}{N_\ell}$. The problem that remains is computing the likelihood of observing the current trajectory given a particular flow tube.

We model the observation likelihood $p_{\mathcal{O}|\mathcal{L}}(T^{curr} \mid \ell)$ as the product of the probability densities of each spatial distri-

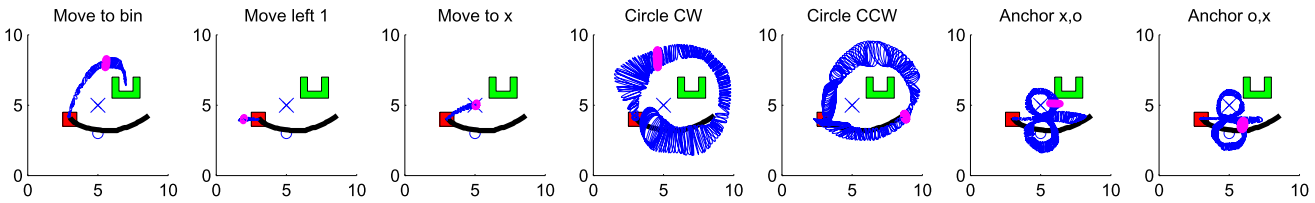

Fig. 6 An example partial test motion (*black*) is compared to each learned PFT (*blue*) in a 2D environment initially with a red box, green bin, and stationary locations x and o as shown. The magenta marking on each PFT indicates the spot on the PFT that best matches the current execution (rightmost end of black motion) as determined by lines 2 to 6 in Algorithm 4. The "anchor x, o" motion moves first around x, then around o, while "anchor o, x" does the opposite. Note the identified position in the "circle clockwise (CW)" PFT is not spatially near the current test position, but rather in a more reasonable position that is temporally consistent with the current execution (Color figure online)

bution in the flow tube evaluated at the temporally aligned points in the current trajectory, i.e.,

$$\prod_{j=1}^{|\mathbf{w}|}\left[\mathcal{N}\left(T_\ell^{eff}(\mathbf{w}_{j,1}),\, \Sigma_\ell^{eff}(\mathbf{w}_{j,1})\right)\Big|_{t^{curr}(\mathbf{w}_{j,2})}\right]^{\frac{1}{|\mathbf{w}|}}.$$

Since the length of the temporal matching matrix $|\mathbf{w}|$ can vary between $\max(M, N)$ and $M + N - 1$, we use the exponent $(\frac{1}{|\mathbf{w}|})$ to perform a multiplicative renormalization over the resampled points to ensure that the overall contribution of the probability values for the trajectory is not inflated by the resampling process.

Taking the log of the observation probability gives

$$\frac{1}{|\mathbf{w}|}\sum_{j=1}^{|\mathbf{w}|}\left[-\log\left((2\pi)^{\frac{d}{2}}\left|\Sigma_{\ell(\mathbf{w}_{j,1})}^{eff}\right|^{\frac{1}{2}}\right)-\frac{1}{2}\boldsymbol{\delta}^T\,{\Sigma_{\ell(\mathbf{w}_{j,1})}^{eff}}^{-1}\boldsymbol{\delta}\right]$$

where $d = \dim(T^{eff})$ and $\boldsymbol{\delta} = T^{curr}(\mathbf{w}_{j,2}) - T_\ell^{eff}(\mathbf{w}_{j,1})$. Finally, we use the pre-computed inverse covariances $\mathbf{I}$ and probability density coefficients $\mathbf{G}$ obtained from model learning to efficiently compute the posterior log likelihood as shown in line 9 of Algorithm 4.

6 Experimental Results

We first test our approach using a two-dimensional simulated environment, and later demonstrate the system working on a Barrett Technologies Whole Arm Manipulator (WAM) robot.

6.1 Two-dimensional Variable Environment

In our simulated environment, there are four entities: a red box, a green bin, and two stationary locations marked x and o. The box and bin are positioned at varying random locations in the environment, while the x and o always mark the fixed locations $(5, 5)$ and $(5, 3)$, respectively.

In our first experiment, we taught the system three different motions: "move the box to the bin", "move the box left one unit", and "move the box to x". During each trial, the box and bin locations were randomly generated. Two users demonstrated 150 trials across the three motions. Thirty randomly chosen trials of each motion were used for training, and the remaining 60 trials were used for testing.

Figure 7 shows some examples of what the learned PFT models look like in the environment states of three randomly chosen test cases. One can see that distinguishing among different motions is easier in some environments than others due to the relative positions of objects. For example, if the bin were located at x and the box straight above, then it is impossible to distinguish between "move to bin" and "move to x".

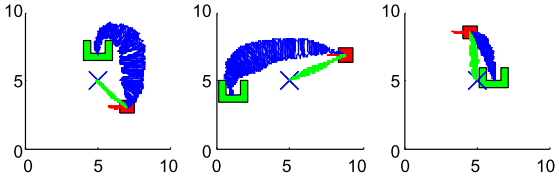

Fig. 7 Example learned flow tubes given different initial environment states. *Blue* PFTs represent "move box to bin", *red* PFTs represent "move box left 1 unit", and *green* PFTs represent "move box to x" (Color figure online)

Table 1 Comparison of recognition approaches using PFTs (P) and HMMs (H) in variable environments. N is the number tests out of 20 that was correctly recognized by the end of each motion. % is the mean percent of real-time execution that the correct motion was recognized. t is the average computation time at each real-time instance (in seconds)

	N_P	N_H	$\%_P$	$\%_H$	t_P	t_H
To bin	**20**	12	**81.1**	55.3	**0.008**	0.047
Left	**13**	6	**56.0**	28.0	**0.004**	**0.004**
To x	**16**	15	**74.4**	36.3	**0.005**	0.006

We compare our approach to that of Martin et al. [20], which represents learned motions as tied-mixture hidden Markov models, and uses a buffered Viterbi algorithm for fast recognition. For comparison, we implemented their approach using 3 states and 6 Gaussian mixtures, which we found to perform reasonably well.

Table 1 compares the results of the first experiment. We first compare the number of test cases correctly recognized by the end of each motion using the PFT approach (N_P) versus the HMM approach (N_H). Our algorithm recognized 49 of the 60 test motions (82 %), while the HMM-based algorithm recognized 33 (55 %).

The second comparison is how long throughout a test motion did the algorithm maintain the correct classification ($\%_P$ versus $\%_H$). On average, our algorithm recognized the motion correctly 71 % of the time spent during a test motion, while the HMM approach spent on average 40 %. This second comparison metric reflects how frequent the real-time estimates are correct throughout a test case, since as a test motion progresses, the estimated most likely motion label may change given new information at each time step.

An example of how estimation likelihoods may change for the two approaches throughout the same test cases is shown in Fig. 8. The PFT approach starts each motion with low likelihood, while the HMM approach starts each motion with equally high likelihood. Throughout the motion, the HMM approach updates the log likelihoods smoothly, but is often unable to distinguish among the motions. The PFT approach often produces higher frequency likelihood changes as a result of the spatio-temporal relationships among the motions, but it is often able to appropriately distinguish

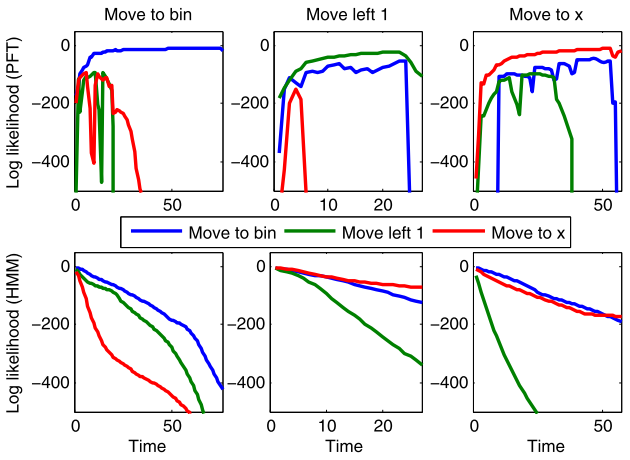

Fig. 8 Example log likelihoods over time for the same test cases using the PFT approach (*top row*) and the HMM approach (*bottom row*) (Color figure online)

among them through time. For example, comparing the "move to x" test case in Fig. 8, the HMM approach has difficulty distinguishing it from the "move to bin" motion. The PFT approach, however, quickly concludes that the label "move to x" is most likely, but "move to bin" also has a small log likelihood throughout. During the first part of execution, "move left 1" is also a low probability contender, but after the user moves beyond 1 unit, the likelihood of the motion being "move left 1" quickly drops to zero, which is a behavior that makes sense intuitively.

Finally, the third metric comparison in Table 1 is how long, on average, the algorithm takes to compute new likelihoods at each time step in a test motion (t_P versus t_H). As expected, the HMM approach gives very fast online computation times, considering human reaction time is generally longer than 0.1 second. Because our PFT approach stores all computation-intensive steps a priori offline and simply applies a few arithmetic operations during online recognition, it is able to achieve comparable online computation times.

We can see that our approach is able to perform real-time recognition even when the environment state varies among the different training and test trials. This is an important advantage in learning and recognizing manipulation tasks because the state of the environment often changes with manipulation, and we want to reduce the number of redundant training demonstrations an operator must perform in any given environment setup.

6.2 Two-dimensional Static Environment

Since the HMM-based technique is not designed to handle variations in the environment state, we chose the particular environment state shown in Fig. 6 and generated 25 trials for each of the depicted 7 motions in order to make a more fair comparison. We used 5-fold cross-validation using 5 trials for training and 20 for testing on each motion. Only 5 trials are needed for training because the motions are fairly similar in the static environment.

Table 2 Comparison of recognition approaches using PFTs (P) and HMMs (H). There were 100 cross-validation test cases for each motion

	N_P	N_H	$\%_P$	$\%_H$	t_P	t_H
To bin	**95**	94	65.6	**67.9**	**0.020**	0.064
Left	94	**100**	81.6	**95.6**	**0.007**	0.009
To x	96	**98**	79.3	**85.1**	**0.009**	0.010
CW	**100**	72	**83.2**	65.2	**0.023**	0.038
CCW	**99**	72	**94.3**	80.7	**0.022**	0.031
x, o	**96**	56	**79.0**	52.8	**0.034**	0.370
o, x	**99**	57	**86.7**	45.4	**0.035**	0.386

Table 2 summarizes the comparative results. The HMM-based approach performed slightly more favorably for the goal directed motions "move to bin", "move left 1", and "move to x". It had more trouble with motions where directionality plays an important role, and it performed poorly on motions that are not Markov in nature, such as the anchor loops. To the Markovian model, the loop, say around x, looks the same locally in the "anchor around x then o" motion as it does in the "anchor around o then x" motion. Overall, the HMM approach achieved a 78 % recognition rate. Our algorithm achieved an overall 97 % recognition rate while using less computation time on all of these motions.

Of the 35 total learning trials, the PFT reported an average offline learning time of 0.707 seconds with a standard deviation of 0.606, and the HMM reported an average learning time of 1.498 seconds with a standard deviation of 0.286. We suspect that learning a PFT on average takes less time than learning an HMM (though with more deviation) because a learning a PFT does not involve learning a mixture of Gaussians, but rather a single Gaussian at each time step, which is a fairly trivial computation compared with the EM algorithm required for GMMs. Furthermore, the HMM also requires autonomously determining the optimal number of states, using the BIC criterion, which can also be computationally intensive.

6.3 Hardware Demonstration

We also demonstrated our motion recognition capability on a Barrett Whole Arm Manipulator (WAM) robot, as shown in Fig. 9. We trained the robot by physically moving it through each of 5 motions in gravity compensation mode 10 times. Each trial for a motion started with the same environment state and was recentered in post-processing to all have the same starting location. Five of the 10 trials for each motion were used for testing for each of two cross-validations.

Table 3 shows that our algorithm successfully recognized all 50 cross-validation test trials correctly by the end of each

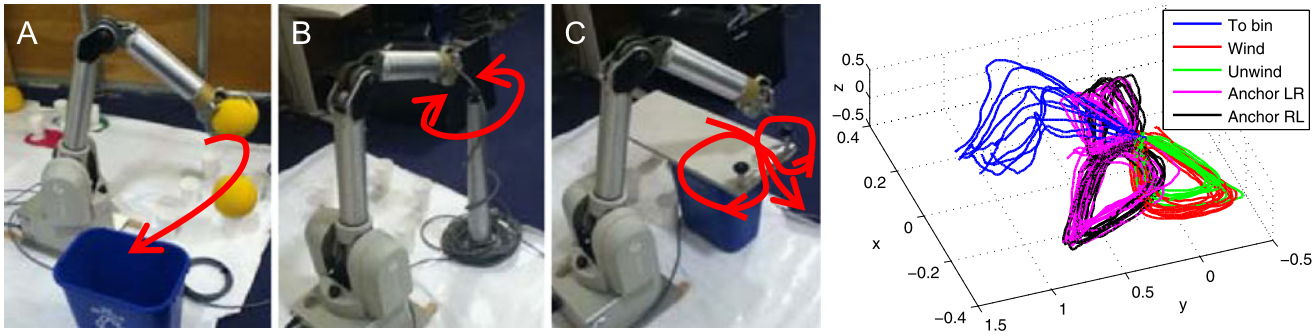

Fig. 9 *Left*: WAM robot setups for the five motions: "move ball to bin" (**A**), "wind cable" (**B**), "unwind cable" (**B**), "anchor rope left then right" (**C**), and "anchor rope right then left" (**C**). *Right*: WAM end effector trajectories. *Bottom*: Results on WAM robot using PFT approach. There were 10 cross-validation test cases for each motion (Color figure online)

Table 3 Results on WAM robot using PFT approach. There were 10 cross-validation test cases for each motion

	N_P	$\%_P$	t_P
To bin	10	93.5	0.024
Wind	10	92.7	0.013
Unwind	10	96.4	0.011
Anchor LR	10	84.1	0.032
Anchor RL	10	86.2	0.033

motion, and spent on average 91 % of each test trial classifying the motion correctly. We observe that the higher dimensionality of the state space is quite favorable for recognition as it allows motions to diverge more.

In this demonstration, the environment state did not change for the different demonstrations, so we only needed a few number of user demonstrations to be able to learn generalized PFTs of each motion. An interesting future study could determine the optimal number of user demonstrations needed for a certain degree of variation in the environment state.

7 Conclusion

We have presented an approach to learning complex physical motions from human demonstration that (1) provides flexibility during execution while robustly encoding a human's intended motions, and (2) automatically determines the relevant features of a motion so that they can be preserved during autonomous execution in new situations.

We have also introduced an approach to real-time motion recognition that (1) leverages temporal information to model motions that may be non-Markovian, (2) provides fast real-time recognition of motions in progress by using an incremental dynamic time warping approach, and (3) employs the probabilistic flow tube representation that enables our method to recognize learned motions despite varying environment states.

We envision several extensions of this approach: (1) the covariance sequence of a probabilistic flow tube can serve as a cost function for compliant execution; (2) overlaying probabilistic flow tubes on potential fields [17, 26] can provide a cost map for obstacle avoidance; (3) longer user demonstrations can be represented as PFT plans consisting of multiple motions, automatically segmented based on qualitative changes in discrete variables; (4) robots can indicate confidence levels for recognized motions based on absolute and relative likelihoods of the most likely learned motions, probing the user for more guidance when confidence is low.

Acknowledgements The authors thank David Mittman, Sarah Osentoski, and Shuo Wang for their help in operating the different robots used in our demonstrations and experiments.

References

1. Abbeel P, Dolgov D, Ng AY, Thrun S (2008) Apprenticeship learning for motion planning with application to parking lot navigation. In: IROS
2. Akgun B, Cakmak M, Yoo JW, Thomaz AL (2012) Trajectories and keyframes for kinesthetic teaching: a human-robot interaction perspective. In: HRI
3. Alissandrakis A, Nehaniv CL, Dautenhahn K, Saunders J (2005) An approach for programming robots by demonstration: generalization across different initial configurations of manipulated objects. In: IEEE international symposium on computational intelligence in robotics and automation
4. Argall B, Chernova S, Veloso M, Browning B (2009) A survey of robot learning from demonstration. Robot Auton Syst 57(3):469–483
5. Atkeson CG, Schaal S (1997) Robot learning from demonstration. In: ICML, pp 12–20
6. Calinon S, D'halluin F, Sauser E, Caldwell D, Billard A (2010) Learning and reproduction of gestures by imitation: an approach based on hidden Markov model and Gaussian mixture regression. IEEE Robot Autom Mag 17(2):44–54
7. Calinon S, Guenter F, Billard A (2007) On learning, representing and generalizing a task in a humanoid robot. IEEE Trans Syst Man Cybern, Part B, Cybern 37:286–298

8. Cederborg T, Li M, Baranes A, Oudeyer PY (2010) Incremental local online Gaussian mixture regression for imitation learning of multiple tasks. In: IROS
9. Coates A, Abbeel P, Ng A (2009) Apprenticeship learning for helicopter control. Commun ACM 52(7):97–105
10. Dixon S (2005) An on-line time warping algorithm for tracking musical performances. In: IJCAI
11. Dong S, Conrad PR, Shah JA, Williams BC, Mittman DS, Ingham MD, Verma V (2011) Compliant task execution and learning for safe mixed-initiative human-robot operations. In: AIAA Infotech
12. Dong S, Williams B (2011) Motion learning in variable environments using probabilistic flow tubes. In: ICRA
13. Frank J, Mannor S, Precup D (2010) Activity and gait recognition with time-delay embeddings. In: AAAI
14. Hamid R, Maddi S, Johnson A, Bobick A, Essa I, Isbell C (2009) A novel sequence representation for unsupervised analysis of human activities. Artif Intell 173(14):1221–1244
15. Hoffmann H, Pastor P, Park DH, Schaal S (2009) Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance. In: ICRA
16. Hofmann A, Williams B (2006) Exploiting spatial and temporal flexibility for plan execution of hybrid, under-actuated systems. In: AAAI
17. Krogh BH (1984) A generalized potential field approach to obstacle avoidance control. In: International robotics research conference, Bethlehem, PA
18. Lee D, Ott C (2011) Incremental kinesthetic teaching of motion primitives using the motion refinement tube. Auton Robots 31(2–3):115–131
19. Li H, Williams B (2008) Generative planning for hybrid systems based on flow tubes. In: ICAPS
20. Martin RA, Wheeler KR, Allan MB, SunSpiral V (2010) Optimized algorithms for prediction within robotic tele-operative interfaces. Technical report, NASA/TM-2010-216417
21. Mühlig M, Giengerand M, Hellbachand S, Steil J, Goerick C (2009) Task-level imitation learning using variance-based movement optimization. In: ICRA
22. Mitra S, Acharya T (2007) Gesture recognition: a survey. IEEE Trans Syst Man Cybern 37(3):311–324
23. Moni MA, Ali ABMS (2009) HMM based hand gesture recognition: a review on techniques and approaches. In: IEEE ICCSIT
24. Myers CS, Rabiner LR, Rosenberg AE (1979) Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. J Acoust Soc Am 66(S1):S34–S35
25. Osentoski S, Manfredi V, Mahadevan S (2004) Learning hierarchical models of activity. In: IROS, Sendai, Japan
26. Park DH, Hoffmann H, Pastor P, Schaal S (2008) Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In: IEEE-RAS international conference on humanoid robots
27. Pastor P, Hoffmann H, Asfour T, Schaal S (2009) Learning and generalization of motor skills by learning from demonstration. In: ICRA
28. Peters RA, Campbell CL (2003) Robonaut task learning through teleoperation. In: ICRA
29. Rasmussen CE, Williams CKI (2006) Gaussian processes for machine learning. MIT Press, Cambridge
30. Riley M, Cheng G (2011) Extracting and generalizing primitive actions from sparse demonstration. In: IEEE-RAS international conference on humanoid robots
31. Salvador S, Chan P (2007) FastDTW: Toward accurate dynamic time warping in linear time and space. Intell Data Anal 11(5):561–580
32. Senin P (2008) Dynamic time warping algorithm review. Technical report, University of Hawaii at Manoa
33. SunSpiral V, Wheeler KR, Allan MB, Martin R (2006) Modeling and classifying Six-dimensional trajectories for teleoperation under a time delay. In AAAI spring symposium
34. Wakabayashi S, Margruder DF, Bluethmann W (2003) Test of operator endurance in the teleoperation of an anthropomorphic hand. In: SAIRAS
35. Wang Z, Li B (2009) Human activity encoding and recognition using low-level visual features. In: IJCAI
36. Yang J, Xu Y, Chen CS (1997) Human action learning via hidden Markov model. IEEE Trans Syst Man Cybern, Part A, Syst Hum 27(1):34–44