Noname manuscript No.
(will be inserted by the editor)

Unsupervised Early Prediction of Human Reaching for Human-robot Collaboration in Shared Workspaces

Ruikun Luo · Rafi Hayne · Dmitry Berenson

Received: date / Accepted: date

Abstract This paper focuses on human-robot collaboration in industrial manipulation tasks that take place in a shared workspace. In this setting we wish to predict, as quickly as possible, the human's reaching motion so that the robot can avoid interference while performing a complimentary task. Given an observed part of a human's reaching motion, we thus wish to predict the remainder of the trajectory, and demonstrate that this is effective as a real-time input to the robot for human-robot collaboration tasks. We propose a two-layer framework of Gaussian Mixture Models and an unsupervised online learning algorithm that updates these models with newly-observed trajectories. Unlike previous work in this area which relies on supervised learning methods to build models of human motion, out approach requires no offline training or manual labeling. The main advantage of this unsupervised approach is that it can build models on-the-fly and adapt to new people and new motion styles as they emerge. We test our method on motion capture data from a human-human collaboration experiment to show the early prediction performance. We also present two human-robot workspace sharing experiments of varying difficulty where the robot predicts the human's motion every 0.1s. The experimental results suggest that our framework can use human motion predictions to decide on robot motions that avoid the human in real-time applications with high reliability.

Keywords Human motion prediction · Human-robot collaboration · Human-robot manipulation · Learning

Ruikun Luo
Robotics Program,
University of Michigan, Ann Arbor, 48109, MI, USA
E-mail: ruikunl@umich.edu

Rafi Hayne
Computer Science Department,
Worcester Polytechnic Institute, Worcester, 01609, MA, USA
E-mail: rhhayne@wpi.edu

Dmitry Berenson
Electrical Engineering and Computer Science Department,
University of Michigan, Ann Arbor, 48109, MI, USA
E-mail: berenson@eecs.umich.edu

1 Introduction

Prediction of human motion is useful for human-robot interaction, especially for a human and robot collaborating in a shared workspace. Previous work (e.g. Mainprice and Berenson (2013)) have shown that early prediction of human reaching motion can help the robot plan its trajectory while avoiding the workspace that the human is going to occupy, which results in a more fluid collaboration in the shared workspace. In this paper, we work on a similar problem as in Mainprice and Berenson (2013), however, we focus more on the algorithm for early prediction of human reaching motion as well as experiments which demonstrate how real-time early prediction can help human-robot collaboration in a shared workspace.

The human reaching motions we consider in this paper are reaching motions in industrial manipulation tasks, e.g. those that would be used for a human and a robot assembling components in a shared workspace. Supervised learning methods are widely used in previous work for recognition of human motions. However these supervised learning methods require an offline training process and manually-labeled training data, i.e. a human gives different sets of motions different labels before the offline training process. The performance of these supervised learning methods depends highly on the accuracy and consistency of the manual

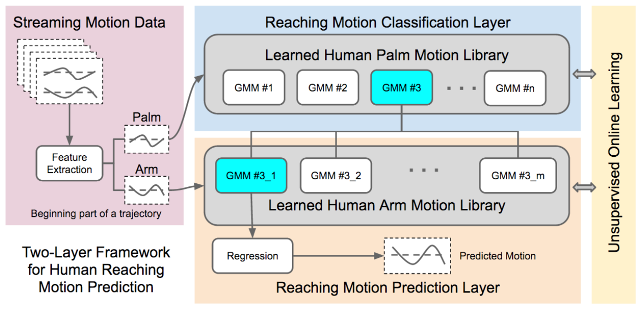

Streaming Motion Data
Feature Extraction
Palm
Arm
Beginning part of a trajectory
Two-Layer Framework for Human Reaching Motion Prediction
Reaching Motion Classification Layer
Learned Human Palm Motion Library
GMM #1
GMM #2
GMM #3
GMM #n
GMM #3_1
GMM #3_2
GMM #3_m
Learned Human Arm Motion Library
Regression
Predicted Motion
Reaching Motion Prediction Layer
Unsupervised Online Learning

Fig. 1 A two-layer framework for human reaching motion prediction. The framework consists of a reaching motion classification layer and a reaching motion prediction layer. The first layer consists of a human *palm* motion library. The second layer consists of a set of human *arm* motion libraries, where each human *arm* motion library links to a class of palm motion in the human *palm* motion library. The two libraries are learned by the proposed unsupervised online learning algorithm.

labeling. As it is hard to keep consistent labeling for geometrically similar motions, e.g. distinguishing motions for the same purpose but with different styles, most of the human labeling describes the purpose of the motion but not the geometric similarity of the motions. However, these geometric features are important if we wish to predict human motions for human-robot collaboration because we need to know what parts of the workspace the human is going to occupy. Thus unsupervised methods, which cluster motions only based on the geometric similarity and then build motion models for each cluster, tend to be useful in this context. Also, if the human changes the way they perform a given task or a new human, with a different method of doing the task, is observed, pre-trained models will not be able to predict the new style of motion. Thus we seek an unsupervised online learning method, which can build models on-the-fly and adapt to new people and new motion styles when they are observed. A key question is whether such a prediction can be fast to compute and is reliable enough for human-robot collaboration in a shared workspace.

Our framework (see Fig. 1) consists of a two-layer library of Gaussian Mixture Models (GMMs) which model the human reaching motions. The reason to use GMMs as elements of the library is that they can be used as generative models (via Gaussian Mixture Regression) and they can describe arbitrary trajectories. The first layer of our framework is a library of human palm motion, which contains a set of GMMs for human's palm position. The second layer is composed of a set of GMMs for the human's arm joint center positions. By using our proposed unsupervised online learning algorithm, our framework can iteratively cluster trajectories based on a geometric similarity measure, and each cluster of trajectories corresponds to a GMM that models those trajectories.

To maintain the motion libraries in our framework, our unsupervised online learning algorithm uses each observed trajectory to determine whether to update the parameters for an existing GMM (using the incremental EM algorithm introduced in Calinon and Billard (2007)) or to initialize a new GMM if none of the existing GMMs can "explain" the new trajectory. Thus we can build models on-the-fly (initialize a new GMM) or adapt (update an existing GMM's parameters) to new people and new motion styles. As our approach can build new GMM models, the framework can handle noise (i.e. atypical reaching motions) by building a new GMM model for the atypical trajectory and adding a membership-proportional prior for each GMM in the library.

Our framework can be used to recognize trajectories (i.e. to determine which cluster of previous trajectories is similar to the given one), but its main purpose is to do early prediction: Given an observed part of a trajectory, the framework can recognize the trajectory first to determine which GMM it belongs to, and then predict the remainder of the trajectory by using Gaussian Mixture Regression (GMR). While GMMs and GMR have been used to represent and predict trajectories, two central contributions allow us to apply these methods to early prediction of human reaching motion without supervision:

1. An unsupervised learning algorithm for human motion recognition allows us to building the library of reaching motions online.
2. A two-layer framework that considers different representations of the human arm allows us to recognize and predict reaching motion.

We first used a dataset from human-human collaboration experiment to compare the recognition and early prediction performance of our method and other supervised/semi-supervised methods. The experimental results showed that our method outperformed the baseline methods. We then performed a simple human-robot collaboration experiment and a more realistic human-robot collaboration experiment. We achieved 99.0% success rate in the first experiment and 93.0% in the second experiment. The experimental results from these two robot experiments showed that our method can be applied in a real-time human-robot collaboration system and even in a more realistic and complicated scenario. The source code and example data is released on GitHub[1].

[1] `github.com/WPI-ARC/unsupervised_online_reaching_prediction`

A previous version of this work appears in Luo and Berenson (2015) and Luo et al (2016), where we tested our method on motion-capture data recorded during a human-human collaboration experiment and a simple real-time human-robot collaboration experiment.The expanded version presented here presents more details about the proposed framework, expands on related work, and presents a more realistic human-robot collaboration experiment to show that our framework can be used in a more complicated environment.

The remainder of this paper is organized as follows: Section 2 will introduce related work. Section 3 will present the proposed unsupervised online learning algorithm. Section 4 will show how to use the two-layer framework for human reaching motion early prediction and how to maintain motion libraries in the framework. Section 5 presents the results for human workspace sharing data. Section 6 shows the experimental results for two real-time human-robot collaboration experiments. Finally we conclude in Section 7.

2 Related Work

Our work contributes to the field of human motion prediction for human-robot collaborations. Most previous work in this area uses supervised learning for human motion recognition and prediction. For example, in Xia et al (2012); Zhao et al (2012); Zhang and Parker (2011), the authors propose different types of feature representations of human motions for use inside a supervised learning framework. While these works aim to find underlying features of human motion, another class of work aims to design the underlying models of human motions. For example, Sung et al (2012b) used a two-layered maximum-entropy Markov model (MEMM) for human activity detection from RGBD images. Koppula et al (2013) used a Markov random field (MRF) to model both human activities and object affordances. Koppula and Saxena (2016, 2013) used Conditional Random Fields (CRFs) for similar applications. Unlike our work, these works recognize or predict action labels rather than human trajectories. Recently, the work of Koppula and Saxena (2016, 2013) has been extended in Jiang and Saxena (2014) to predict high-dimensional trajectories rather than action labels. However, this still differs from the work presented in this paper because they aim to predict future human actions rather than predict the remainder of a trajectory. In our work, we recognize the observed part of a human's motion and then predict the remainder of this trajectory. The early prediction of human motion is useful for a robot to react quickly to human motion in a human-robot collaboration task.

The problem of early motion prediction has been studied in previous work. Mainprice and Berenson (2013) used GMMs to model human reaching motions and GMR to predict the remainder of an observed trajectory. Recently Mainprice et al (2015) explored using Inverse Optimal Control (IOC) to learn a cost function under which demonstrated trajectories are optimal and used that cost function to do iterative re-planning to predict human reaching motions. In related work, Perez-D'Arpino and Shah (2015) used additional task-level information as a prior for early human reaching motion prediction. However, the above methods are all supervised learning algorithms, which require an offline training process and a batch of labeled training data. Unlike these previous works, we consider unsupervised online learning, which requires no manually-labeled data and no offline training process.

Unsupervised human motion prediction has been explored in Weinrich et al (2013), for the purpose of predicting the trajectories of pedestrians. We are interested in prediction of human reaching motions, which are more challenging to predict because they are more high-dimensional and execute much faster than pedestrian motion. Kulić et al (2011) proposed an online method for incremental learning of full-body motion primitives. They segmented the human motion into several motion primitives and then used a Hidden Markov Model (HMM) to model both the structure of the primitives and each motion primitive. Unlike their method, we model sets of trajectories using a library of GMMs because we are interested in modeling human reaching motion, which is not clearly separable into primitives. Calinon and Billard (2007) proposed a GMM-based incremental learning of gestures for humanoid robot imitation. The incremental training of a GMM is done by the human manually moving the robot. We use the same incremental EM method proposed in their work as part of our algorithm. However, unlike their work, our framework is given motions corresponding to different tasks and can cluster the motions into different classes. Unsupervised online learning GMMs have been studied in speech recognition Barras et al (2004); Zhang and Scordilis (2008). Unlike these works, which rely on a well-trained background GMM, our proposed unsupervised online learning algorithm requires no offline training.

The learning part of our framework can be treated as an unsupervised online clustering algorithm. Clustering trajectories has been studied in previous work. Cederborg et al (2010) proposed an incremental local online Gaussian Mixture Regression algorithm for imitation learning. Their work focused on local trajectories by building local GMMs around each current state using the data points which are close to the current state. Sung et al (2012a) proposed a trajectory clustering algorithm which clustered the trajectory into several local motion patterns. These local motion primitives can then be used to build a Hidden Markov Model for predicting trajectories. Bennewitz et al (2002) proposed an EM-algorithm-based method that clustered human trajectories into different types of motions while simultaneously learning multiple motion patterns. Bruce and Gordon (2004) incorporated this method with a path planner to predict human trajectories in order to improve a human tracking system. Unlike works focusing on learning local patterns, we focus on building new motion models for entire trajectories. Nyga et al (2011) focused on clustering human trajectories and learning the relations between these trajectories as well as the contexts in which they are used. However, this work operated offline on a batch of human trajectories, while our online framework can handle streaming input data.

We intend to use our framework for human-robot collaboration. Ravichandar and Dani (2015) proposed an algorithm to infer human intention by modeling human motion for human-robot collaboration tasks. Unlike their work which focused on predicting a goal location, we focused on predicting the remaining part of the human trajectory including the positions of arm joint centers which can then be used to extract a goal location. Maeda et al (2016) proposed a method to learn probabilistic movement primitives for human-robot collaborative tasks. In their work, they learned motion primitives for both a human and robot together and predicted robot trajectories after observing the human motions. Unlike their work which required offline training and focused on the motion primitives of collaborations of the human and robot, our approach is an online algorithm and focuses on the prediction of the entire human trajectory in order to infer the workspace which the human is going to occupy.

3 Unsupervised Online Learning Algorithm

In this section we introduce the core component of our framework: the unsupervised online learning algorithm. Fig. 2 shows the pipeline of our proposed algorithm. This algorithm is designed to learn GMMs that model trajectories.

As shown in Fig. 2, the algorithm builds and maintains a trajectory library that consists of multiple GMMs where each GMM G_i represents a class of trajectories. Given a trajectory X_j, the algorithm will first calculate the probabilities of this trajectory given each GMM - $p(X_j|G_i)$ for $i = 1, 2, ..$ and then calculate the posterior probability $p(G_i|X_j)$ (explained in Section 3.2). If

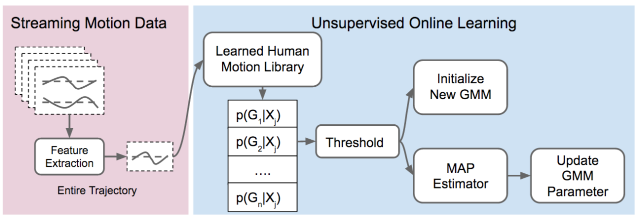


Fig. 2 Data flow for the unsupervised online learning algorithm.

all the posterior probabilities are smaller than a specified threshold, the algorithm will use this trajectory X_j to initialize a new GMM and store it in the trajectory library. If some posterior probabilities are larger than that threshold, the algorithm will classify (maximum a posteriori estimation) this trajectory into a GMM class G_k with the highest probability $p(G_k|X_j)$. Then the algorithm will update the parameters of the GMM G_k.

Algorithm 1 shows the pseudocode of the proposed unsupervised online learning algorithm. Note that we define a structure of trajectory library - Lib as shown in Algorithm 1. This trajectory library structure contains the feature used in this trajectory library, the threshold used in this trajectory library as well as the learned GMMs for different trajectory classes. **ExtractFeature** is the function to extract features from the observed data. **LogPrior** and **LogProb** are the functions to compute log-likehood of $p(G_i|X_j)$ (see Section 3.1 and Section 3.2). **OneTrajBuildGMM** is the approach to initialize a new GMM using only one input trajectory (see Section 3.3). **UpdateGMM** is the function to update the parameters of the selected GMM in the trajectory library using the observed trajectory (see Section 3.4). The rest of this section will discuss all these functions in detail. This approach is used at both levels of our framework (as in Fig. 1).

3.1 Gaussian Mixture Models for Trajectories

In this section, we discuss how to use GMM to model trajectory data. As shown in Algorithm 1, each GMM in the trajectory library represents a class of trajectories. G_i for $i = 1, 2, 3, ...$ represents each GMM in the library. X_j for $j = 1, 2, 3, ...$ represents a given trajectory. X_j is an $L \times D$ matrix where L is the number of points in a trajectory and D is the number of feature dimensions of the trajectory.

Each GMM G_i is a combination of K multivariate Gaussians gc_k for $k = 1, 2, 3, ..., K$. Let ξ_j^l be a vector that concatenates the time index l and the extracted feature from the lth point in the trajectory. The probability of a point ξ_j^l in GMM G_i represented by K mul-

Algorithm 1: Unsupervised Online Learning

```
input  : Lib: trajectory library
         Lib.GMMs = $\{G_1, G_2, ..., G_n\}$: a set of GMMs
         Lib.threshold: threshold parameter for Lib
         Lib.feature: feature used in Lib
         X: observed trajectory
output: Lib: updated trajectory library
1  X ← ExtractFeature(X, Lib.feature);
2  if Lib.GMMs == $\phi$ then
3      G ← OneTrajBuildGMM(X);
4      Lib.GMMs ← Lib.GMMs $\bigcup$ G;
5  else
6      p ← $\max_{G \in Lib.GMMs}$ LogPrior(G,Lib,X)+LogProb(G,X);
7      G ← $\operatorname{argmax}_{G \in Lib.GMMs}$ LogPrior(G,Lib,X)+LogProb(G,X);
8      if p < Lib.threshold then
9          G ← OneTrajBuildGMM(X);
10         Lib.GMMs ← Lib.GMMs $\bigcup$ G;
11     else
12         UpdateGMM(G,X)
```

tivariate Gaussians is given by:

$$p(\xi_j^l|G_i) = \sum_{k=1}^{K} p(gc_k|G_i)p(\xi_j^l|gc_k, G_i) \tag{1}$$

where ξ_j^l is the lth row vector of X_j, representing the lth point of trajectory X_j. $p(gc_k|G_i) = \pi_k$ (we will use π_k in the following section) is the prior probability of component gc_k in G_i. The probability of ξ_j^l given gc_k and G_i is defined as follows:

$$\begin{aligned} p(\xi_j^l|gc_k, G_i) &= \mathcal{N}(\mu_k, \Sigma_k) \\ &= \frac{1}{\sqrt{(2\pi)^D|\Sigma_k|}} e^{-\frac{1}{2}(\xi_j^l-\mu_k)^T \Sigma_k^{-1}(\xi_j^l-\mu_k)} \end{aligned} \tag{2}$$

where $\{\mu_k, \Sigma_k\}$ are the mean and covariance parameters of the Gaussian component gc_k. Thus the probability of trajectory X_j in G_i is defined as follows:

$$p(X_j|G_i) = \prod_{l=1}^{L} p(\xi_j^l|G_i) \tag{3}$$

Using GMMs to represent trajectory has an important advantage: we can use the parameters not only for classification, but also to do GMR, which can be used for prediction as shown in Calinon (2009).

3.2 Classifying trajectories

In this section, we discuss how to determine which (if any) GMM a trajectory belongs to while accounting for trajectories of varying length. Previous work (Mainprice and Berenson (2013)) used Maximum likelihood estimation (MLE) to estimate the class label. This assumption makes sense because the number of motion classes is fixed and they do not need to compare different trajectories. However, in our proposed unsupervised online learning algorithm, we need to decide a threshold to determine whether a trajectory is used for initializing a new GMM or for updating an existing GMM's parameters. This threshold is a constant regardless of trajectory length, so we need to define a likelihood of each trajectory that is not as sensitive to trajectory length as Eq. (3).

Unlike Eq. (3), we assume the geometric mean of the postures' probability densities given G_i is the probability density of trajectory X_j in G_i, shown as follows:

$$p(X_j|G_i) = \sqrt[L]{\prod_{l=1}^{L} p(\xi_j^l|G_i)} \tag{4}$$

The geometric mean accounts for the length of the trajectory in the probability density.

As an unsupervised online learning algorithm, the proposed algorithm can capture some noisy trajectories and build GMMs for these trajectories. The prior distribution of the GMMs thus should not be the uniform distribution as GMMs for a noisy trajectory would be given the same weight as those for a commonly-used trajectory. Instead we propose a prior distribution we call a "ratio prior", which can also be interpreted as a regularizer:

$$p(G_i) = \frac{f(r_i)}{\sum_{i=1}^{M} f(r_i)} \tag{5}$$

where M is the current number of GMMs, r_i is the ratio between the number of trajectories classified in G_i and the total number of the trajectories observed so far. We define $f(r_i) = \arctan(10r_i)/\arctan 10$ as the potential function shown in Fig. 3. The potential will drop quickly as the ratio decreases to 0 and will increase smoothly as the ratio increases to 1. Thus GMMs with small numbers of trajectories will be assigned small values of the

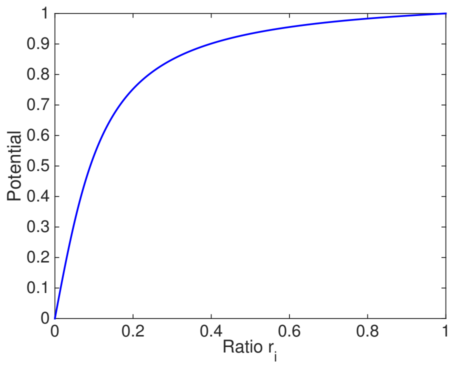


Fig. 3 Potential function of r_i

prior and GMMs with significant numbers of trajectories will be assigned similar large values of prior. The ratio prior can be treated as a "denoising" function in order to reduce the influence of the noisy motions. Note that, at the beginning of the experiment, there will be a small number of GMMs and each GMM will have a small number of trajectories. The normalization of the prior ensures that, in this case, each GMM receives a similar prior because there should be no prior information for each GMM at the beginning. In Section 5, we show that this ratio prior outperforms a uniform prior and MLE (no prior). The uniform prior is defined as $p(G_i) = 1/M$.

Combining Eqs. (4) and (5), the posterior probability distribution of X_j is as follows:

$$\begin{aligned} p(G_i|X_j) &\propto p(G_i)p(X_j|G_i) \\ &= p(G_i) \sqrt[L]{\prod_{l=1}^{L} p(\xi_j^l|G_i)} \end{aligned} \tag{6}$$

As the product of the probability is too small to represent accurately, we use the log of the probability. The log-likelihood of $p(G_i|X_j)$ is be computed as follows:

$$\log(p(G_i|X_j)) = \frac{1}{L}\sum_{l=1}^{L} \log p(\xi_j^l|G_i) + \log p(G_i) \tag{7}$$

The first term in the right-hand side of Eq. (7) is the function **LogProb** in Algorithm 1. Similarly, the second term is the function **LogPrior** in Algorithm 1.

3.3 Initializing a GMM from single trajectory

In general, initialization of a new GMM requires a set of training data and uses K-means and expectation-maximization (EM) algorithms to compute the prior, mean, and covariance matrix of each multivariate Gaussian component. The number of training trajectories and the variance of the trajectory data will influence the generality of the GMM. If the number of training trajectories is small and they are very similar, the GMM variance will be very low and all other trajectories will get near-zero probability given this GMM. This problem is especially acute when we try to generate a GMM from a single trajectory. For our framework, we need a way to generate a GMM from a single trajectory such that the variance is not too low. Thus we propose a Random Trajectory Generation (RTG) algorithm to generate random trajectories that are close to a given trajectory. The generated trajectories and the given trajectory can be used as training data to initialize a new GMM using the standard method in Calinon and Billard (2007). This method, which builds a new GMM from a single trajectory is the function **OneTrajBuildGMM** in Algorithm 1.

Algorithm 2: Random Trajectory Generation

```
  Input      : Trajectory $X \in \mathbb{R}^{T\times D}$
               $\Delta$: Maximum distance to $X$
  Precompute: $A$ = finite difference matrix (Eqn 8)
               $R^{-1} = (A^T A)^{-1}$
               $Q = R^{-1}$ with each column scaled
  such that the maximum elements is $1/L$
1 begin
2     Generate difference matrix $\delta X$ where each column
       vector $\theta_i \sim \mathcal{N}(0, R^{-1})$ for $i = 1, 2, 3, .., D$
3     while DTW(X, X + $\delta$X) > Δ do
4         $\delta X = Q\delta X$
5     end
6 end
```

Similar to the STOMP algorithm of Kalakrishnan et al (2011), which sampled trajectories to estimate a gradient for optimization, our RTG algorithm also uses a finite differencing matrix A, which is an $(L + 2) \times L$ matrix that, when multiplied by the position vector θ, produces accelerations $\ddot{\theta}$:

$$A = \begin{bmatrix} 1 & 0 & 0 & & 0 & 0 & 0 \\ -2 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 1 & -2 & 1 & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & & 1 & 0 & 0 \\ 0 & 0 & 0 & & -2 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & -2 & 1 \\ 0 & 0 & 0 & & 0 & 1 & -2 \\ 0 & 0 & 0 & & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

where L is the length of the trajectory and position vector θ is the column vector of the points of a given trajectory X. The idea of RTG is to generate a random difference trajectory δX and iteratively reduce the difference scale until the distance between the generated trajectory $X + \delta X$ and the given trajectory X is smaller than some given value. The RTG algorithm is shown in Algorithm 2. The covariance matrix $R^{-1} = (A^T A)^{-1}$ and normalization matrix Q can ensure that the generated trajectory in each iteration keeps the same goal and start position. $Q = R^{-1}$, and each column vector is scaled such that the maximum element is $1/L$. Q is used to reduce the difference of the generated trajectory iteratively. Note that the maximum distance Δ can be used to control how close the generated trajectory is to the observed one. This value can help control the covariance of the initialized new GMM. The difference between trajectories is calculated using the Dynamic Time Warping (DTW) metric described in

Müller (2007), using Euclidean distance as the underlying distance metric.

3.4 Update GMM parameters

When a given trajectory X has the log posterior probabilities $\log(p(G_i|X))$ (as in Eq. (7)) larger than the threshold, the algorithm uses MAP estimation to assign a GMM ID to this trajectory as follows:

$$\hat{i} = \operatorname*{argmax}_i \log(p(G_i|X)) \tag{9}$$

Then we use the directed update method for incremental EM from Calinon and Billard (2007) to update the parameters of GMM $\hat{i}$. Recall that $X = \{\xi^l | l = 1, 2, 3, ..., L\}$, where ξ^l is a column vector. The incremental EM algorithm assumes ξ^l as training data, thus we have L data points for the current update. Let $\tilde{L}$ represent the number of all previous data points to train this GMM. We set the current GMM $\hat{i}$'s parameters $\{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$ as the initial parameters $\{\tilde{\pi}_k^{(0)}, \tilde{\mu}_k^{(0)}, \tilde{\Sigma}_k^{(0)}\}_{k=1}^K$. Let $\tilde{p}_{k,l} = p(k|\xi^l)$ represent the posterior probability, where k is the kth Gaussian component. Let $\{\tilde{E}_k^{(0)} = \sum_{l=1}^L \tilde{p}_{k,l}^{(0)}\}_{k=1}^K$. The incremental EM procedure is then:

E-step:

$$\begin{aligned} \tilde{p}_{k,l}^{(t+1)} &= \frac{\tilde{\pi}_k^{(t)} \mathcal{N}(\xi^l; \tilde{\mu}_k^{(t)}, \tilde{\Sigma}_k^{(t)})}{\sum_{i=1}^K \tilde{\pi}_i^{(t)} \mathcal{N}(\xi^l; \tilde{\mu}_i^{(t)}, \tilde{\Sigma}_i^{(t)})} \\ \tilde{E}_k^{(t+1)} &= \sum_{l=1}^L \tilde{p}_{k,l}^{(t+1)} \end{aligned}$$

M-step:

$$\begin{aligned} \tilde{\pi}_k^{(t+1)} &= \frac{\tilde{E}_k^{(0)} + \tilde{E}_k^{(t+1)}}{L + \tilde{L}} \\ \tilde{\mu}_k^{(t+1)} &= \frac{\tilde{E}_k^{(0)} \tilde{\mu}_k^{(0)} + \sum_{l=1}^L \tilde{p}_{k,l}^{(t+1)} \xi^l}{\tilde{E}_k^{(0)} + \tilde{E}_k^{(t+1)}} \\ \tilde{\Sigma}_k^{(t+1)} &= \frac{\tilde{E}_k^{(0)} (\tilde{\Sigma}_k^{(0)} + (\tilde{\mu}_k^{(0)} - \tilde{\mu}_k^{(k+1)})(\tilde{\mu}_k^{(0)} - \tilde{\mu}_k^{(k+1)})^T)}{\tilde{E}_k^{(0)} + \tilde{E}_k^{(t+1)}} \\ &+ \frac{\sum_{l=1}^L \tilde{p}_{k,l}^{(t+1)} (\xi^l - \tilde{\mu}_k^{(t+1)})(\xi^l - \tilde{\mu}_k^{(t+1)})^T}{\tilde{E}_k^{(0)} + \tilde{E}_k^{(t+1)}} \end{aligned}$$

The two steps iterate until convergence. This method is the function **UpdateGMM** in Algorithm 1.

4 Two-Layer Framework for Early Prediction of Human Reaching Motion

In the previous section, we introduced our proposed unsupervised online learning algorithm which updates parameters of a GMM or builds a new GMM to update the trajectory library. In this paper, we use this algorithm for human reaching motions. We consider three types of features to represent human motions: 1) palm position (PP), 2) arm joint center positions (AJCP), 3) arm configurations (AC). The feature comparison experiment in Section 5.1 will show comparisons between these representations and the reason why we only use the first two feature representations in our framework. In this section, we will introduce how we use our proposed framework to predict human reaching motions and how to use the proposed unsupervised online learning algorithm to maintain the motion libraries in the framework.

Our proposed framework for early prediction of human reaching motion is shown in Fig. 1. It consists of a reaching motion classification layer and a reaching motion prediction layer. The first layer is the current learned human *palm* motion library, which contains a set of GMMs where each GMM represents a class of human reaching motion. The second layer contains a set of human reaching *arm* motion libraries, where each human *arm* motion library links to a GMM in the human *palm* motion library in the first layer. The human motion libraries are learned by the unsupervised online learning algorithm (Section 3).

4.1 Early Prediction of Human Reaching Motion

The purpose of early human motion prediction is to regress the remainder of a human's trajectory based on the observed part of the trajectory. We decompose the human motion early prediction problem into two steps: 1) human motion early recognition and 2) human motion trajectory regression. As we focus on the application of human motion prediction for human-robot collaboration tasks, we require regressing the whole arm trajectory (not only the palm trajectory) in order to compute the human's workspace occupancy. However, the results in Table 1 show that the proposed unsupervised online learning algorithm using PP features significantly outperforms the algorithm using AJCP features in the *recognition* task. As early recognition is vital for the early prediction problem, we propose a two layer framework for human reaching motion early prediction (Fig. 1). The first layer uses PP features and the second layer uses AJCP features. This two-layer framework can take the advantages of PP features (better recognition performance) and can still model the whole arm trajectory. Both layers use the proposed unsupervised online learning algorithm to build their motion libraries. The first layer builds a palm motion library and the second layer builds an arm motion library for each palm motion class as shown in Fig. 1. Note that as an online system, the framework will observe human motion trajectories one-by-one and human motion postures from each trajectory one-by-one. At the beginning of each trajectory, the framework will only do early prediction based on the current learned models. After observing this trajectory, the framework will then update the human motion libraries using the method in Fig. 2 for each layer.

Fig. 1 shows the data flow for the human reaching motion early prediction. The early prediction consists of the following steps:

1. *Feature extraction*: The framework observes the beginning part of the human motion and extracts two types of features: PP and AJCP.
2. *Human motion early recognition*: The first layer of the framework takes the PP features and uses MAP to estimate the palm motion class ID i (GMM ID in the library). As each GMM in the first layer links to a different human *arm* motion library in the second layer, we then use the palm motion class ID i found in the first layer to find the corresponding human *arm* motion library in the second layer. Then the second layer takes the AJCP features and uses MAP to estimate the human arm motion class ID j in this arm motion library.
3. *Human motion trajectory regression*: The second layer computes the regressed trajectory X' using the method of Calinon and Billard (2007) with the GMM parameters of the jth human arm motion class in the arm motion library for the ith palm motion class.
4. *Normalize regressed trajectory*: Move the regressed trajectory such that the beginning posture of the regressed trajectory overlaps with the end posture of the observed trajectory.

Though it may be possible to first predict the palm trajectory and then compute the Inverse Kinematics (IK) solutions on the predicted palm trajectory to generate the arm trajectory, we do not take this approach because the human arm has redundant DoFs. There are no unique IK solutions for a given palm pose, and it is difficult to predict which IK solution the human will choose.

4.2 Updating Motion Libraries in Two-Layer Framework

To update the libraries in the framework, we run Algorithm 1 at both layers as follows: In the first layer of the framework, there is only one palm motion library. Thus we can directly run Algorithm 1 to update this palm motion library. If the algorithm updates one GMM in the palm motion library, then we find the arm motion library which links to this GMM in the second layer and run Algorithm 1 to update this corresponding arm motion library. If the algorithm builds a new GMM in the palm motion library, then we use Algorithm 1 to generate a new arm motion library with a new GMM in the second layer and link it to the new GMM in the palm motion library.

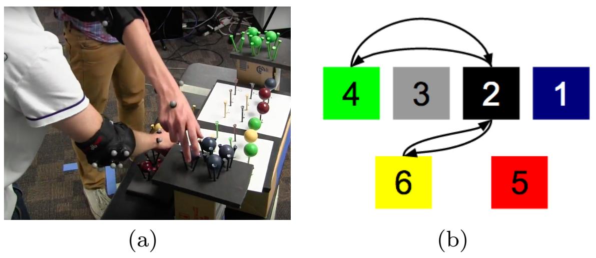


Fig. 4 (a) Experiment setup. Two human subjects are performing the assembly task side by side. We only considered the motions for the "active" human on the right. (b) Reaching motions we considered in this paper were moving balls between location 2 and 4, and location 2 and 6.

5 Results from Human Workspace Sharing Data

To test our framework, we require an experiment where human subjects perform a variety of reaching motions in an industrial context. In this context, we can expect that a human will spend most of their time performing the given task in the same way, but will occasionally change their reaching motion to avoid another worker who has temporarily entered their space (e.g. to retrieve a part). Thus we also wish to see how our framework performs in the presence of noise (i.e. reaching motions that are atypical for the task at hand). In the previous work of Mainprice and Berenson (2013) observed that a human in isolation would produce the same stereotypical motion with low variance when asked to perform a repetitive reaching task. To produce a more realistic set of trajectories that includes noise we devised an experiment where two humans are performing an assembly task side-by-side (see Fig. 4(a)). We devised the assembly task so that the humans occasionally need to reach into their partner's workspace, therefore forcing their partner to change their reaching strategy to avoid them. The "active" human, the person on the right in our experiments, is the one whose motion we wish to predict. The other human is there to generate disturbances that force the active human to produce atypical (i.e. "noisy") trajectories. We wish to include such trajectories in our testing because we want to evaluate our system's robustness to atypical motion, which humans commonly exhibit (e.g. responding to an interruption from a co-worker).

The assembly task required the "active" human to move balls between location 2 and location 4 and between location 2 and location 6 as shown in Fig. 4(b).

We used a VICON system to capture the human motions. Human subjects wore a suit consisting of nine markers and three rigid plates which were placed following the standards used in the field of biomechanics (Wu et al (2005)). Our suit consists of rigid marker plates attached to a belt, a headband and an elbow pad, a marker on the back of the hand, two on each side of the wrist, two on either side of the shoulder, and two markers straddling the sternum and xyphoid process. The VICON system runs at 100 fps. We used recordings from 3 pairs of human subjects doing the assembly task and each pair performed the assembly task 6 times. Thus we have 18 sets of experiment data. There were a total of 254 trajectories captured from the three "active" human subjects. The average number of frames in each trajectory is 107. The algorithm is implemented in MATLAB. The average runtime to process a trajectory (update the parameters or add a new GMM model) is 0.1s and the average runtime for one call of the prediction process is 0.0036s.

In the proposed two-layer framework, we setup the parameters as follows: To initialize a new GMM, we set the $\Delta = 10$ and generate 5 random trajectories for PP, however, we set $\Delta = 45$ and generate 10 random trajectories for AJCP. We set the threshold as -8 in the first layer and -108 in the second layer. The parameters were found by manual tuning.

Below we compare feature representations and methods in terms of their precision, recall, and accuracy in trajectory prediction. Precision and recall are computed as the average precision and recall over all 4 classes of motions. For the supervised or semi-supervised methods, the number of GMMs is fixed and each GMM has the same label as the training data that trains this GMM. For the unsupervised methods, the number of GMMs is not fixed and we are actually clustering the trajectories. When we compute the precision and recall, the trajectories in the unsupervised GMMs will take the label of the ground truth label of the majority of the trajectories in that GMM. The analysis is set up this way to show that we cluster trajectories for the same task together, even though we do not know what the tasks are.

5.1 Feature comparison experiment

In this section we evaluate which features allow GMMs to model human reaching motions most effectively. We ran leave-one-out experiments to compare different human motion feature representations using supervised GMMs. In each round of the leave-one-out experiment, we used 1 of 18 sets of the experiment data as the testing data and other 17 sets as training data. We considered three features: PP, AJCP and AC. The AJCP are positions for the right arm's palm, wrist, elbow and shoulder, which are recorded by our motion capture system. In this paper, we only considered the joint angles of the arm in the AC feature, which we obtain through IK on the set of markers. The dimensions of each feature representation are 3, 12, and 9, respectively.

Table 1 Feature Representation Comparison

	PP	AJCP	AC
precision(%)	98.6 ± 3.4	98.1 ± 4.1	90.8 ± 7.3
recall(%)	98.6 ± 3.4	97.9 ± 5.2	87.0 ± 13.1

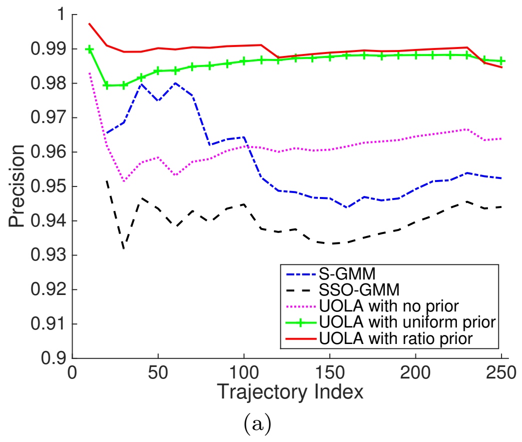
Table 1 shows the performance for each type of feature using supervised GMMs. Both the PP and AJCP outperform the AC and have no significant difference between them. Although the AC tries to reduce the influence of different body types, people with different body type may perform the same motion with different joint angles. Thus we only used PP features and AJCP features in the rest of the experiments.

5.2 Human reaching motion trajectory recognition

In this section, we compared our proposed unsupervised online learning algorithm (UOLA) with supervised GMMs (S-GMM) and semi-supervised online GMMs (SSO-GMM). We also tested using the PP feature vs. the AJCP feature with different methods in order to see which feature is more suitable for which method. For the S-GMM, we only used data from one pair of the human subjects' first run of the assembly task as training data and tested on the rest of the data as streaming input in order to simulate real world circumstance. For the SSO-GMM, we used the same training and testing data as supervised GMMs. The training data is used to initialize the 4 GMMs (4 classes of motions). Given a trajectory, the SSO-GMM classified this trajectory into one of the 4 GMMs (e.g. the ith GMM) and use this trajectory to update the ith GMM's parameters using the incremental EM algorithm of Calinon and Billard (2007). For our proposed algorithm (UOLA), there was no training data and we used the whole dataset as streaming test data. We also tested using different types of priors for UOLA: ratio prior, uniform prior, and no prior (i.e. MLE). Note that in this experiment, we only test our UOLA algorithm at the first layer in our framework because we wish to evaluate its performance for recognition. We ran the experiments 100 times for each model and each feature representation. Table 2 shows the performance of each model. It shows that all the models have better performance using the PP

Table 2 Human Reaching Motion Trajectory Recognition

	Precision(%)	Recall(%)	# GMM
S-GMM (PP)	95.3 ± 1.5	94.2 ± 2.5	4
S-GMM (AJCP)	78.1 ± 5.7	68.0 ± 10.6	4
SSO-GMM (PP)	94.4 ± 1.9	92.3 ± 4.1	4
SSO-GMM (AJCP)	72.7 ± 6.4	30.0 ± 4.5	4
UOLA (PP, MLE)	$\mathbf{97.6 \pm 3.4}$	$\mathbf{96.7 \pm 6.5}$	9.8 ± 2.8
UOLA (AJCP, MLE)	84.8 ± 5.2	68.6 ± 13.8	14.6 ± 4.5
UOLA (PP, uniform)	$\mathbf{99.3 \pm 2.2}$	$\mathbf{98.8 \pm 4.5}$	17.0 ± 3.3
UOLA (AJCP, uniform)	86.8 ± 4.8	74.1 ± 14.2	17.5 ± 5.1
UOLA (PP, ratio)	$\mathbf{98.8 \pm 2.1}$	$\mathbf{98.6 \pm 4.0}$	16.3 ± 2.9
UOLA (AJCP, ratio)	85.9 ± 4.2	68.5 ± 13.4	17.4 ± 5.6



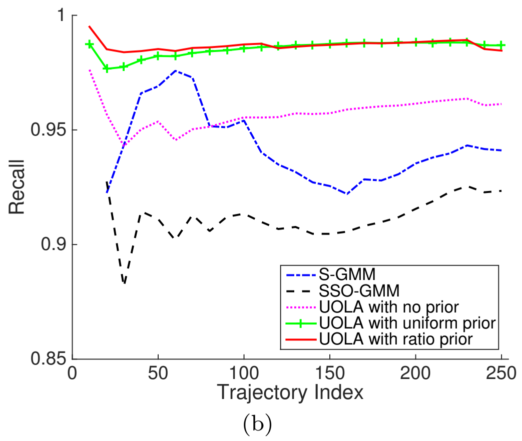


Fig. 5 Human reaching motion trajectory recognition experiment using only the PP feature. (a) Precision vs. trajectory indices, (b) Recall vs. trajectory indices. The proposed unsupervised online learning algorithm with ratio prior and uniform prior consistently outperform the baselines.

feature. Although the PP feature is contained in the AJCP feature, AJCP feature contains more information such as the positions of the elbow and shoulders of the human arm which do not appear to be useful for reaching motion classification. The GMM considers all features equally and does not have the ability to remove unhelpful features, so the inclusion of such features decreases performance. The number of GMMs shows that our algorithm is not over-clustering the dataset.

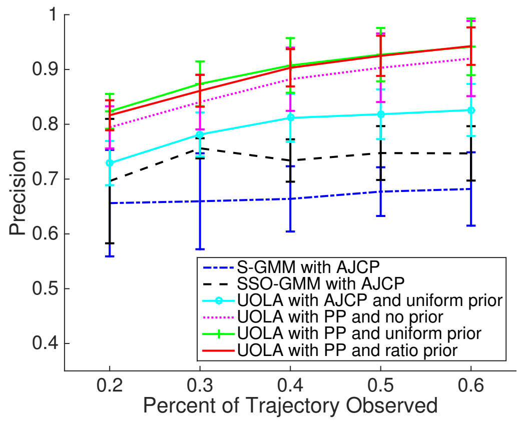
Fig. 5 shows the performance changes along with the trajectory indices streaming into the system. Here we only considered the PP feature as it always outperforms the AJCP feature. The figure shows that both proposed unsupervised online learning algorithm with ratio prior and uniform prior consistently outperform the baselines. The figure also shows that MAP estimation with ratio prior and uniform prior outperform the MLE estimation and have no significant difference between them.

5.3 Human reaching motion trajectory early recognition

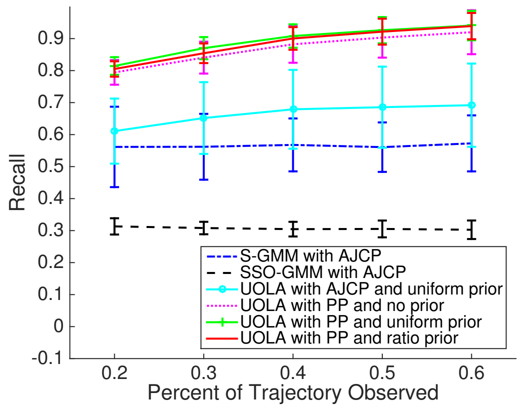
Early recognition of a human motion trajectory is the first step of human motion early prediction. The performance of early recognition is crucial for the human motion trajectory prediction, as using the correct generative model is necessary for trajectory prediction. Note that our final goal is to predict the remainder of a given arm motion trajectory, so we only considered the baseline methods that have generative models for AJCP features: S-GMM with AJCP features and SSO-GMM with AJCP features. We compared the baseline methods with our proposed UOLA (one layer) with AJCP features, uniform prior and PP features with different priors. We tested with different observed percentages $(20\%, 30\%, 40\%, 50\%, 60\%)$ of the trajectory for each method. We used the same testing data and training data for each method as mentioned in Section 5.2. We ran each method 100 times on the dataset. Fig. 6 shows the overall performance of each method. All of our proposed UOLA variants outperform the baselines. The proposed UOLA using PP features with ratio prior and uniform prior slightly outperform UOLA using PP features with no prior.

5.4 Human reaching motion trajectory prediction

In this section, we used the same setup as the previous section, however, we focused on the performance of motion trajectory prediction for each method. In the previous experiment we found that the UOLA variants with PP features outperform the UOLA with AJCP features for early recognition. This result shows that PP features are most useful for early recognition, however, since we wish to predict the entire arm's trajectory, we require a second layer to perform this prediction that uses AJCP features. Thus in this experiment we use both layers of our framework: the first with PP features, and the second with AJCP features. The evaluation method is the DTW distance between the predicted trajectory and



(a)



(b)

Fig. 6 Human reaching motion trajectory early recognition experiment. (a) Average precision vs. percent observed, (b) Average recall vs. percent observed. The proposed UOLA using PP features with uniform prior and ratio prior consistently outperform the baselines.

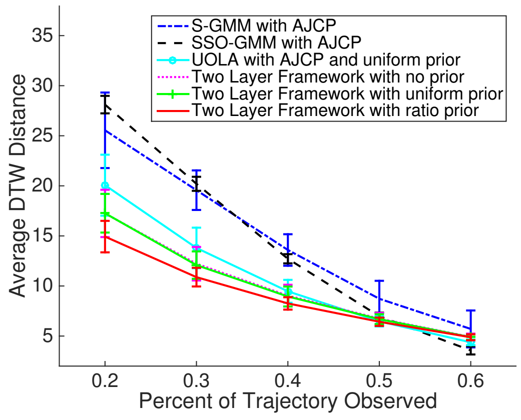


Fig. 7 Average DTW distance vs. percent observed of the trajectory for each model. Both layers of the framework are used in this experiment. We tested on all 254 trajectories in our data set. Both of our proposed two-layer framework variants outperform the baselines.

the remainder of the given testing trajectory. Fig. 7 shows the relationship between the average DTW distance and the percent observed of the trajectory. All of our proposed two-layer framework variants outperform the baselines. The framework using ratio prior outperforms the framework variants using uniform prior and using no prior. Note that our proposed framework with the ratio prior significantly outperforms the baselines when given a small percentage of the observed part of the trajectory (e.g. 20%, 30%). This result suggests that, using our proposed framework, the robot is likely to infer the human's true motion more quickly than the baseline methods.

Fig. 8 shows an example prediction comparing the proposed two-layer framework with ratio prior with S-GMM. This example illustrates that early prediction, when performed with a supervised method considering only the task label, can lead to very erroneous prediction. Because the early recognition of the S-GMM is not correct, the prediction is not close to the real trajectory and even goes in the wrong direction (see Fig. 8(c)). Our proposed framework gives a better prediction even when just observing the first 20% of the trajectory (see Fig. 8(e)).

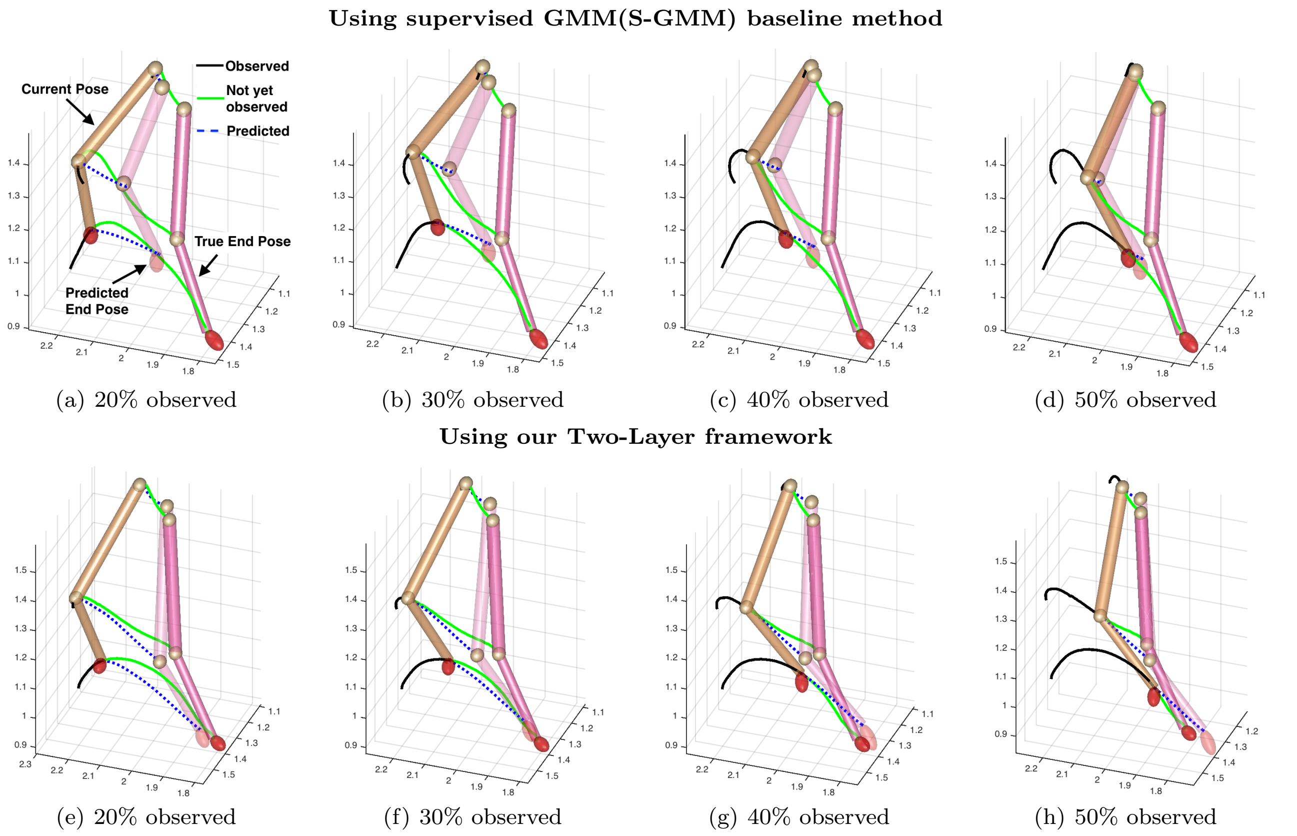
6 Application to Human-Robot Workspace Sharing

In this section, we present results from two human-robot experiments to show that our framework can help robots plan and re-plan their motions in order to avoid disturbing human motions. We used the proposed two layer framework with the ratio prior in these experiments. Note that in all experiments in this section, we ran our two-layer framework with ratio prior. The predicted trajectories are all generated from the second layer of our framework using the AJCP feature.

6.1 Experiment 1

6.1.1 Experimental setup

The first experimental setup is shown in Fig. 9. The subject and PR2 robot are on opposite sides of the table and must each reach into a shared workspace. An incorrect prediction of the subject's motion can lead to the robot contacting the human. The subject's motion is captured by the VICON system using the same setup as the previous experiment at 100 fps. The robot predicts the subject's reaching motion using our framework every 10 frames (0.1 seconds) and does planning (first 10 frames) or re-planning (if the predicted goal of the subject changes). The robot uses the last frame of the predicted trajectory and computes the Euclidean distance between the palm and centers of the color pads



(a) 20% observed (b) 30% observed (c) 40% observed (d) 50% observed

(e) 20% observed (f) 30% observed (g) 40% observed (h) 50% observed

Fig. 8 A comparison of our framework to S-GMM on the 90th trajectory. The first row uses S-GMM and the second row uses our proposed two-layer framework with ratio prior. Each column is given different percents of observed trajectory (20%, 30%, 40% and 50%, respectively). The yellow arm is the end pose of the observed part of the trajectory, which is also the start pose of the remainder trajectory. The pink arm is the end pose of the remainder of the trajectory and the translucent pink arm is the end pose of the predicted trajectory. The black curves are the observed part of the trajectories for each joint. The green curves are the remainder trajectories for each joint, which are the ground truth. The blue doted curves are the predicted trajectories for each joint. The goal of early prediction is to compute the blue dotted curves that are as close as possible to the ground-truth green curves.

(yellow and black) to determine which pad the subject is reaching for. It assumes the closest color pad as the predicted goal of the subject. At the end of each trajectory, the framework is updated with the subject's trajectory. As we are not considering automatic segmentation in this work, another human is required to tell the robot the start and end of each trajectory using a joystick controller. Also note that we simplify the motion planning problem as it is not the focus of this work; the robot motion planning and re-planning selects one of two pre-specified paths and, in the case of re-planning, transitions to the new path from its current configuration. The experiment was performed by 5 subjects. Each subject reaches the color pads 60 times (yellow 30 times, black 30 times). The average duration of each trajectory is 2.32 seconds (232 frames).

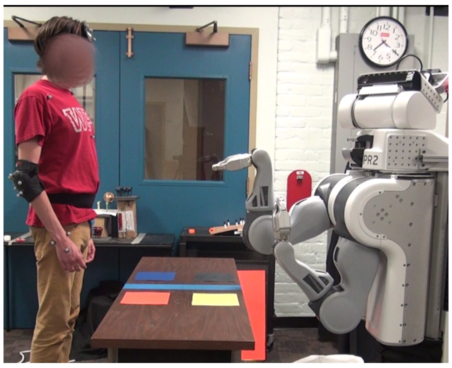
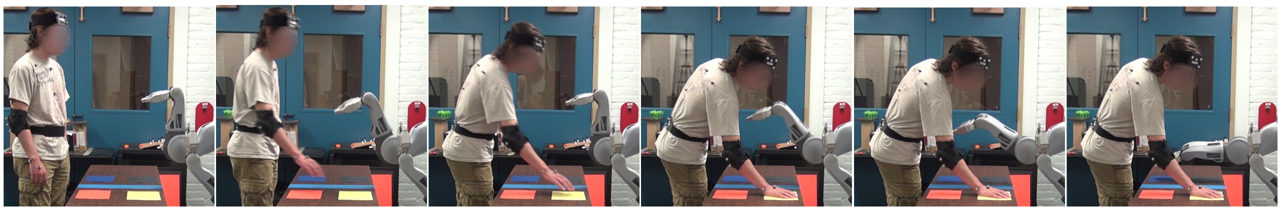


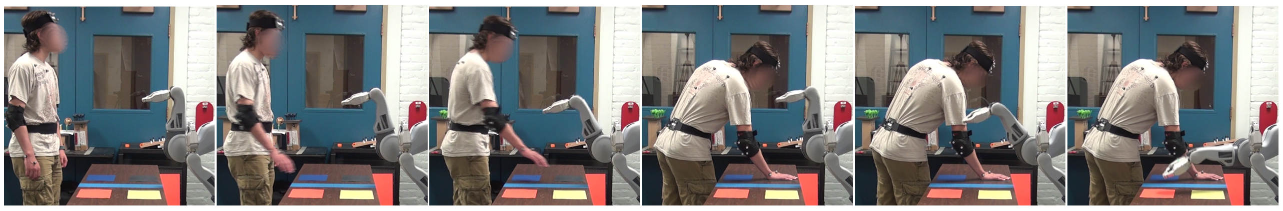
Fig. 9 Experiment 1 setup. The subject and robot are standing opposite to each other. The subject's task is to touch the yellow or black pads and the robot's task is to touch either the blue or orange pads.

6.1.2 Experiment 1 results

Of the 300 trials there were three trials that failed (the robot reached for the wrong color once and the robot



(a) Init/Orange dt=0.0s (b) Re-planning dt=0.55s (c) Init/Blue dt=0.92s (d) Blue dt=1.83s (e) Blue dt=2.48s (f) Complete dt=3.17s



(g) Init/Blue dt=0.0s (h) Blue dt=0.33s (i) Re-planning dt=0.80s (j) Init/Orange dt=1.56s (k) Orange dt=2.50s (l) Complete dt=3.47s

Fig. 10 Comparison of two trials of the experiment (time proceeds from left to right). "Init/Orange" means the robot is at the initial position and starts to reach for the orange pad. "Orange" means the robot is reaching for the orange pad. "Init/Blue" and "Blue" are likewise for reaching for the blue target. "Re-planning" means that the robot re-plans at this time, i.e. switching to the other target to avoid the human. "Complete" means the robot has reached one of the goals. The re-planning point in the top row occurs earlier in execution than that of the bottom row, allowing the robot to reach its target faster while avoiding the human.

re-planned correctly but still touched the human twice). There were 106 trials where the robot directly went to the correct target, which means that the framework successfully predicted the human motions using only the first 10 frames. There were 191 trials where the robot re-planned. Fig. 10 shows examples of two trials in which re-planning occurs due to incorrect initial predictions. The top row is a re-planned trajectory that is consistent with our findings of average re-planning times (Fig. 11(a)), while the bottom row shows an example of a later change in prediction resulting in a smaller distance between robot and subject.

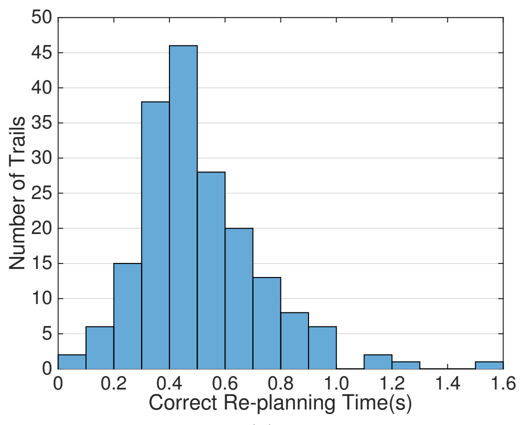
Fig. 11(a) shows a histogram of the time between robot execution starts and re-planning points. The average amount of time it takes our framework to change its initial prediction of the human reaching motion to the correctly predicted motion is 0.5032 ± 0.2195 seconds. Fig. 11(b) shows the average prediction accuracy of the human reaching motion along with the time. It shows that the accuracy reaches 70% after 0.5 second and reaches almost 100% after 1 second. Comparing with the average duration of each trajectory (2.3 seconds), shows that the proposed two-layer framework can perform well in the early prediction task for real-time applications.

Fig. 12(a) shows the histogram of the trajectory duration time. The average trajectory duration time is 2.33 ± 0.62 seconds. Fig. 12(b) shows the number of trajectories in each GMM in the palm motion library. Different colors indicate trajectories belonging to different subjects. It shows that the proposed algorithm can build new models on-the-fly as new motion style emerges and it can also handle the noise (e.g. atypical motions). It also shows that similar motions from multiple participants are indeed clustered into the same library components while the number of clusters with only a few motions is quite small relative to the number of trajectories (300 total). Overall this experiment demonstrates that our method is very effective at real-time prediction in a simple two-task scenario.

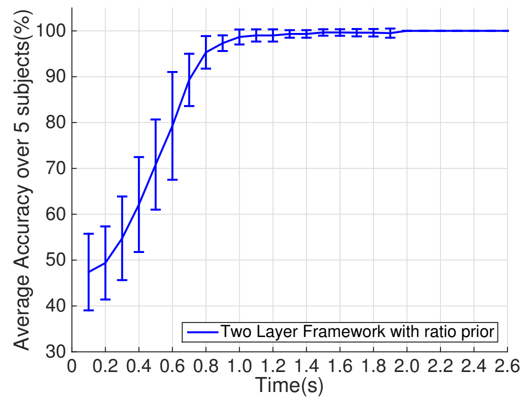
6.2 Experiment 2

6.2.1 Experiment Setup

The second experiment has a more realistic setup as might be seen in an assembly task (shown in Fig. 13). The subject and PR2 robot are on opposite sides of the table. Unlike the first experiment in which the subject is standing, the subject is sitting on a stool in this experiment as one would in a factory or laboratory. The table setup is shown in Fig. 13(a). The subject's task is to move a cylinder to one of the six goal regions while balancing a ball resting on the cylinder (i.e. transporting a delicate item). This specific task was chosen in order to simulate the types of constrained motions seen in industrial manipulation tasks. The robot collaborator has two tasks to perform based on the subject's predicted workspace occupancy. The robot's right arm task is to assist the subject by wiping the left part of the table when the left part of the subject's workspace is predicted to be free. The robot's left arm task is to

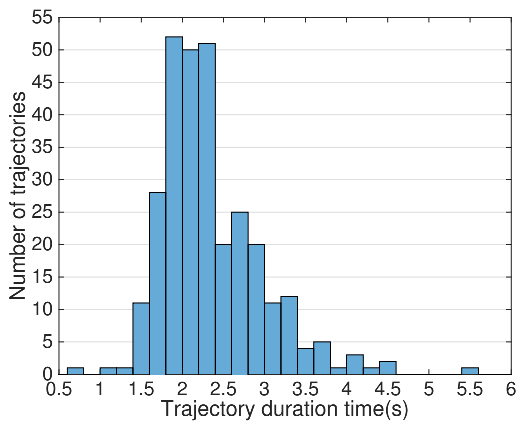


(a)

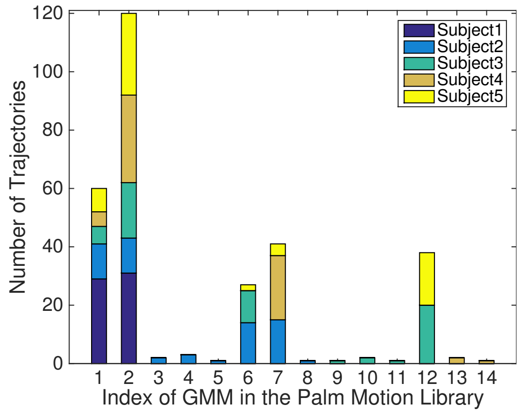


(b)

Fig. 11 (a) Histogram of time between robot execution starts and correct re-planning points. (b) Accuracy of prediction over the 5 subjects vs. time. A prediction is considered accurate if the last frame of the predicted trajectory is closer to the correct target than to the incorrect one.



(a)



(b)

Fig. 12 (a) Histogram of trajectory duration time. (b) Number of trajectories in each GMM in the palm motion library. Different colors indicate different subjects.

reach to the same region as the subject when the predicted motion is reaching to Goal 1-4 (i.e. to receive the object), or not to move when the predicted motion is reaching to Goal 5 and Goal 6 (i.e. the object is not meant for the robot). Note that the two arms of the PR2 robot will move simultaneously when necessary. We capture the subject's motion using the same VICON system recording at 100 fps and use a similar setup to the first experiment except that the threshold used in the first layer is -6. As in the first experiment, the robot predicts the subject's reaching motion every 10 frames (0.1 seconds) and does planning in the initial 10 frames or re-planning if the predicted goal changes. The robot computes the Euclidean distance between the palm position in the last frame of the predicted trajectory and centers of each goal region. The closest goal region is then selected as the predicted goal. As in the first experiment, another human segments the subject's motion and the framework is updated with the subject's trajectory at the end of each reaching motion. As above, the robot selects its motion from pre-specified paths and transitions to the path from its current configuration in the case of re-planning. This experiment was performed by 13 subjects. Each subject reaches each goal region 10 times (60 times in total) for a library of 780 human trajectories in total. The average duration of each trajectory is 2.21 ± 0.78 seconds.

6.2.2 Experiment Result

Fig. 14 shows two different example trials in the experiment. The top row shows that the human reaches towards Goal 3. In this case, the right arm should wipe the table for the human because the left part of the human workspace is free and the left arm should reach to Goal 2. In this trial, as the robot's initial prediction is correct, the robot can directly execute the correct trajectories as mentioned before. The bottom row shows that the human reaches towards Goal 2. In this case, the robot right arm should not move because the left part of the human workspace will be occupied by the

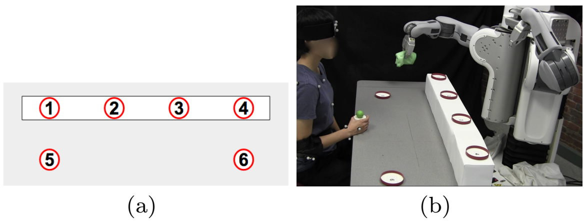


Fig. 13 Experiment setup. (a) Table setup. There are 6 goal regions on the table labeled as Goal 1-6. The Goal 1-4 are on a small platform close to the robot. (b)The subject and robot are sitting opposite to each other. The subject's task is to move the cylinder to one of the 6 goal regions while keeping the ball on the cylinder. The robot's right arm goal is to wipe the left part of table when the left part of the workspace is free. The robot's left arm task is to reach to the same region as the subject if the subject is going to reach Goal 1-4, or to not move if the subject is going to reach Goal 5-6.

human and the robot left arm should reach to Goal 2. In this trial, the robot's initial prediction is not correct, so the robot's right arm moves at the beginning (as shown in Fig. 14(h)). However, the robot changes the prediction correctly after observing more of the human motion. Thus, the robot re-plans its trajectory; the right arm moves back to the initial position and the left arm moves toward Goal 3.

Fig. 15(a) shows the histogram of human trajectory duration time. Note that there were only 2 trajectories of 780 total trajectories longer than 6.0 seconds and 17 trajectories longer than 4.0 seconds. Fig. 15(b) shows the number of trajectories in each GMM in the palm motion library. Different colors indicate trajectories belonging to different subjects. It shows that the proposed framework built 6 major GMM classes related to the 6 goal regions and 4 additional GMM classes. The average trajectory durations for GMM 9 and GMM 10 are 4.02 ± 0.78 seconds and 6.79 ± 2.84 seconds respectively. The trajectories belonging to these two GMMs are from the trials with a task failure or where the robot significantly interfered with the subject's trajectory.

Of the 780 trials, there were 725 success trials for a success rate of 93.0%. There were four types of failure cases: 1) the robot touched white platform (3 times); 2) the robot reached to the wrong position (29 times); 3) the robot interfered with the human but resolved later (36 times); 4) the robot significantly interfered with the human such that the human could not finish the task (4 times). Note that some trials may exhibit more than one failure case. The first failure case was not caused by our algorithm, but rather an anomaly produced by the interface with the PR2's controller. For the second failure case, there were only 7 failures caused by an incorrect prediction from our proposed framework. The other 29 failures were caused by the way we computed the predicted goal region relative to that region's position in the real world. However this problem can be easily solved by calibration of goal positions (discussed in section 6.3). If we ignore this predicted goal computation problem, there were 46 failed trials in total. For the third failure case, the interference is not significant and can be resolved later. Similar interferences between human and human were observed in the previous human workspace sharing data of the human-human collaboration experiment. For the fourth failure, the problem is that the initial prediction is not correct and the robot interferes with the human. The robot continued to change the prediction as the human moved back and forth. The human then ended the trial because they could not complete the task.

Only considering the prediction performance, there were 742 trials in which the robot predicted correctly at the end of the trajectory, 86 trials in which the robot predicted correctly after the first 10 frames of the trajectory and 656 trials in which the robot predicted correctly after re-planning. Fig. 16(a) shows the histogram of the time between robot execution starts and correct re-planning points. Here we only consider the trials that the robot predicted correctly after re-planning. The average time it takes our framework to change its initial predicted human reaching motion to the correctly predicted motion is 1.02 ± 0.77 seconds. Fig. 16(b) shows the average prediction accuracy over 13 subjects along with the time. It shows that the accuracy reaches 70% accuracy after 1.0 second and reaches 90% accuracy after 2.1 seconds. Finally, we achieved 95.3% accuracy.

6.3 Discussion

There are two reasons why the early prediction performance in the second experiment is not as good as in the first experiment. First, the experimental setup is more challenging, due to more goal regions which are spaced closer to each other. For example, the beginning part of reaching motions towards Goal 2 are similar to the beginning part of the motion towards Goal 1 or Goal 3, requiring more time to observe unambiguous data that would be necessary for a correct prediction. Second, as in the first experiment, we computed the Euclidean distance between the predicted palm position with the center of the goal regions and consider the closest goal region as the predicted goal. This simple method worked well in the first experiment because the subject's task was to put his or her hand on the center of the goal region. In this experiment however, the subject grasped a cylinder and put it on the center of the goal region. Thus there is a distance between the palm position and the center of the cylinder dependant

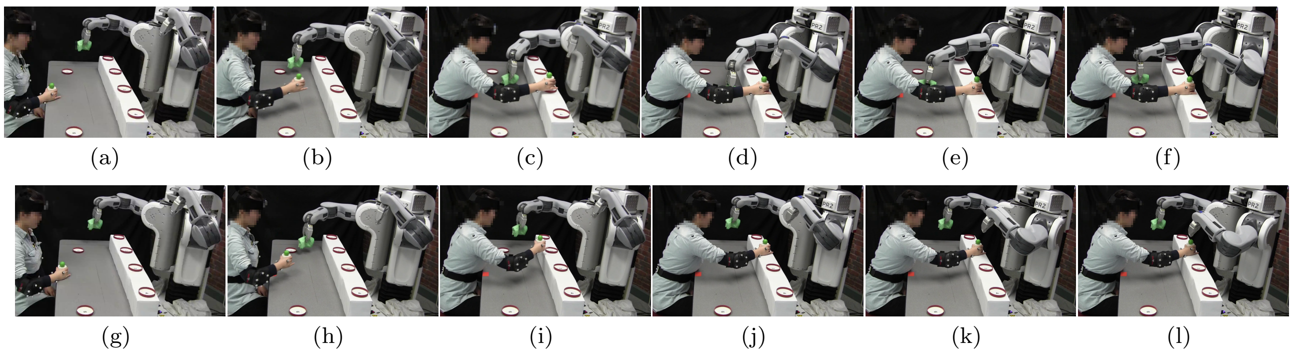


Fig. 14 Comparison of two trials of the experiment (time proceeds from left to right, dt = 0.8s). The top row shows that the human reaches towards Goal 3 and the initial prediction of human reaching motion is correct. Thus the robot left arm reaches towards Goal 3 and right arm wipes the table. The bottom row shows that the human reaches towards Goal 2 and the initial prediction of the human reaching motion is not correct. Thus the robot right arm moves to wipe the table for the human (as shown in (h)). However, the robot modifies its prediction correctly. Then the robot moves its right arm back and moves its left arm towards Goal 2.

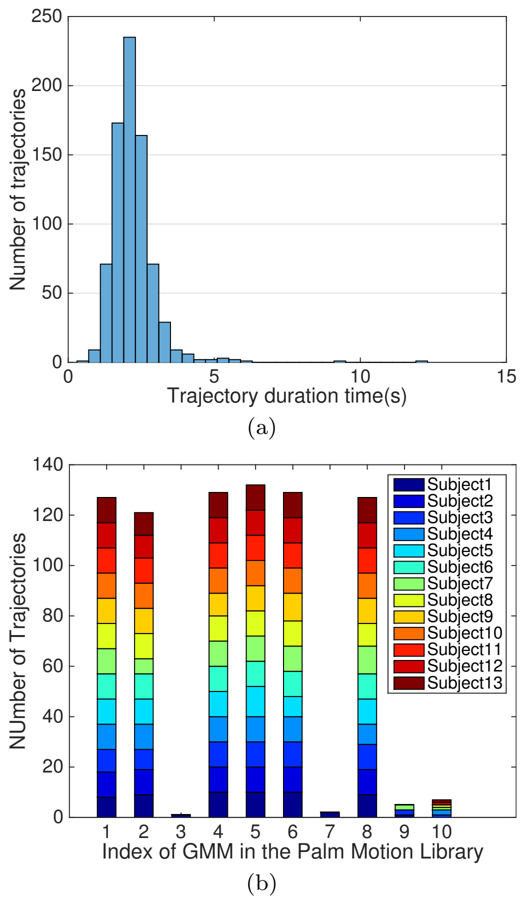


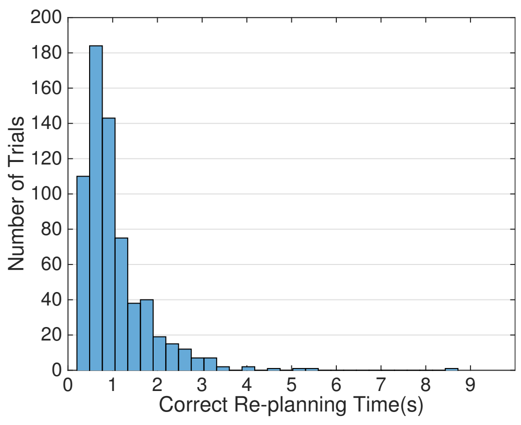
Fig. 15 (a) Histogram of human trajectory duration time. The average human trajectory length is 2.21 ± 0.78 seconds. Note that there were only 2 trajectories over 780 trajectories longer than 6.0seconds. (b) Number of trajectories in each GMM in the palm motion library. Different colors indicate different subjects.

on how the human handled the cylinder and the human hand size. This problem does not affect the algorithms in our framework but it does influence the computation of the predicted goal. This problem manifested itself in some trials when the human reached towards Goal 1. In such trials, our framework correctly classified the human trajectory into the motion class for Goal 1, but the subject's palm rested closer to the center of Goal 2 than that of Goal 1 causing an incorrect prediction. This problem can be solved by calibration of the goal position.

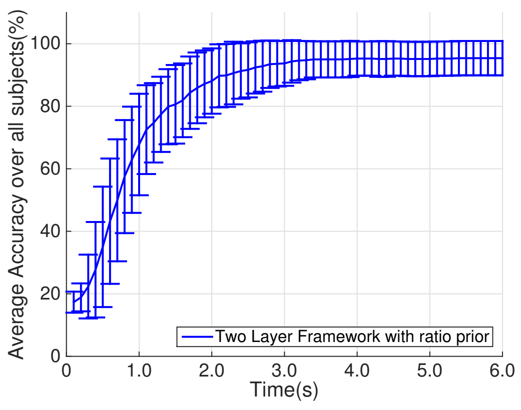
However, despite the calibration issues, the recognition performance of our framework is still good. There were 772 correctly recognized trajectories of all 780 trajectories which resulted in 99.0% recognition accuracy. In fact, only GMM 10 is a confused motion class (4 trajectories for goal 1 and 3 trajectories for goal 2). These trials are the cases in which the human hesitated to resolve a conflicting robot motion, swaying between two otherwise separate areas of the workspace. This is an example of the atypical motions we would expect to observe when working with humans.

7 Conclusion

We have presented a two-layer framework for unsupervised online human reaching motion recognition and early prediction. The framework consists of a two-layer library of GMMs. The library grows if it cannot "explain" a new observed trajectory by using the proposed unsupervised online learning algorithm. Given an observed portion of a trajectory, the framework can predict the remainder of the trajectory by first determining which GMM it belongs to, and then using GMR to predict the remainder of the trajectory. The proposed unsupervised online learning algorithm requires no offline training process or manual categorization of trajecto-



(a)



(b)

Fig. 16 (a) Histogram of time between robot execution starts and correct re-planning points. (b) Accuracy of prediction over the 13 subjects vs. time. A prediction is considered accurate if the last frame of the predicted trajectory is closer to the correct target than to the incorrect one. The prediction after the end of the trajectory is assumed to by the same as the last prediction of the end of that trajectory.

ries. The results show that our framework can generate models on-the-fly and adapt to new people and new motion styles as they arise. Results from two experiments where a human and robot share a workspace show that our framework can be used to decide on robot motions that avoid the human in real-time applications with high reliability. Future work will explore how to use GMR to generate smoother predicted trajectories and explore fast motion planning algorithms for the robot.

References

Barras C, Meignier S, Gauvain JL (2004) Unsupervised online adaptation for speaker verification over the telephone. In: *The Speaker and Language Recognition Workshop*

Bennewitz M, Burgard W, Thrun S (2002) Learning motion patterns of persons for mobile service robots. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol 4, pp 3601–3606

Bruce A, Gordon G (2004) Better motion prediction for people-tracking. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*

Calinon S (2009) *Robot programming by demonstration: A Probabilistic approach*. EPFL Press

Calinon S, Billard A (2007) Incremental learning of gestures by imitation in a humanoid robot. In: *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pp 255–262

Cederborg T, Li M, Baranes A, Oudeyer PY (2010) Incremental local online gaussian mixture regression for imitation learning of multiple tasks. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp 267–274

Jiang Y, Saxena A (2014) Modeling high-dimensional humans for activity anticipation using gaussian process latent CRFs. In: *Robotics: Science and Systems*

Kalakrishnan M, Chitta S, Theodorou E, Pastor P, Schaal S (2011) STOMP: Stochastic trajectory optimization for motion planning. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp 4569–4574

Koppula HS, Saxena A (2013) Learning spatio-temporal structure from rgb-d videos for human activity detection and anticipation. In: *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pp 792–800

Koppula HS, Saxena A (2016) Anticipating human activities using object affordances for reactive robotic response. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38(1):14–29

Koppula HS, Gupta R, Saxena A (2013) Learning human activities and object affordances from rgb-d videos. *The International Journal of Robotics Research* 32(8):951–970

Kulić D, Ott C, Lee D, Ishikawa J, Nakamura Y (2011) Incremental learning of full body motion primitives and their sequencing through human motion observation. *The International Journal of Robotics Research* 31(3):330–345

Luo R, Berenson D (2015) A framework for unsupervised online human reaching motion recognition and early prediction. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp 2426–2433

Luo R, Hayne R, Berenson D (2016) Early prediction of human reaching motion for long-term human-robot collaboration. In: *AI for Long-term Autonomy Workshop at IEEE International Conference on Robotics* *and Automation (ICRA)*

Maeda GJ, Neumann G, Ewerton M, Lioutikov R, Kroemer O, Peters J (2016) Probabilistic movement primitives for coordination of multiple human–robot collaborative tasks. *Autonomous Robots* pp 1–20

Mainprice J, Berenson D (2013) Human-robot collaborative manipulation planning using early prediction of human motion. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp 299–306

Mainprice J, Hayne R, Berenson D (2015) Predicting human reaching motion in collaborative tasks using inverse optimal control and iterative re-planning. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp 885–892

Müller M (2007) *Dynamic Time Warping*, Springer, pp 69–84

Nyga D, Tenorth M, Beetz M (2011) How-models of human reaching movements in the context of everyday manipulation activities. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp 6221–6226

Perez-D'Arpino C, Shah JA (2015) Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp 6175–6182

Ravichandar HC, Dani A (2015) Human intention inference and motion modeling using approximate em with online learning. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp 1819–1824

Sung C, Feldman D, Rus D (2012a) Trajectory clustering for motion prediction. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp 1547–1552

Sung J, Ponce C, Selman B, Saxena A (2012b) Unstructured human activity detection from rgbd images. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp 842–849

Weinrich C, Volkhardt M, Einhorn E, Gross HM (2013) Prediction of human collision avoidance behavior by lifelong learning for socially compliant robot navigation. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp 376–381

Wu G, Van der Helm FC, Veeger HD, Makhsous M, Van Roy P, Anglin C, Nagels J, Karduna AR, McQuade K, Wang X, Werner FW, Buchholz B (2005) {ISB} recommendation on definitions of joint coordinate systems of various joints for the reporting of human joint motionpart ii: shoulder, elbow, wrist and hand. *Journal of Biomechanics* 38(5):981 – 992

Xia L, Chen CC, Aggarwal J (2012) View invariant human action recognition using histograms of 3d joints. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp 20–27

Zhang H, Parker LE (2011) 4-dimensional local spatio-temporal features for human activity recognition. In: *Proceedings of the IEEE/RSJ international conference on Intelligent robots and systems(IROS)*, pp 2044–2049

Zhang Y, Scordilis MS (2008) Effective online unsupervised adaptation of Gaussian mixture models and its application to speech classification. *Pattern Recognition Letters* 29(6):735–744

Zhao Y, Liu Z, Yang L, Cheng H (2012) Combing rgb and depth map features for human activity recognition. In: *Proceedings of the Asia-Pacific Signal Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp 1–4