Robot Learning from Demonstration: Trina via Mocap

TVM Team: Sihui Li, Tianyu Cheng, Arjun Ram, Sanjuksha Nirgude, Yudong Yu
Haowei Zhao, Gaurav Vikhe

April 9, 2018

1 Introduction

When we have a robot in a factory, household, or hospital, how would the robot learn new tasks and new environments? Specialist could program the robot to complete the task, but that is not optimal. End users are not robotics specialist in most case. We want to have a generic and human friendly way to teach the robots. Robot learning from demonstration(LfD) is natural outcome from human experiences. Just like children would try to mimic their caretaker, learn new skills and apply them to different situations, we want the robots to do the same.
In this study, we mainly think in the use case of nursing robot. Nursing robot has wide applications in today's hospitals. When dealing with medical emergency or patients with infective diseases, nursing personnel is facing high infection rate[12]. Also, with today's aging society, the need for personal care is increasing. There are often cases where the patients need constant caring or monitoring, even though nursing tasks are sparsely distributed during a long period of time. In all the above cases, the help of a intelligent nursing robot would be highly valued for lower the risk of infection or making nursing work more efficient.
For the current Trina tele-nursing robot, because there are so many degrees of freedom (14 on arms, 8 for hands, 2 for the mobile platform), it is hard to control them all together. Input devices such as game pad, keyboard, mouse, haptic devices are used. However, they are limited by their own control abilities. For example, if the human operator wants to use game pad to command Trina, because the number of buttons on a game pad is smaller than the degrees of freedom of Trina, the game pad works in a mode-switching way. For dexterous manipulations, the human operator would need to constantly switch the control modes, which is time consuming, hard to manipulate, and not intuitive for human. Also, for such control strategies, there is always a learning curve for novice users. The task performance depends on the human operator's expertise heavily.
To address these problems, we wish to use Motion capture system(Mocap) and LfD method on Trina. By using Mocap, the human body is used directly as a input device to teleoperate Trina. This control strategy has the potential to perform dexterous manipulations in teleoperation scenarios. With LfD, the robot would be able to extent its learned knowledge to new situations, thus frees the human operator from tedious repeated operations. In all, we want to establish an intuitive teleoperation and human-robot teaching-learning interface.
Reinforcement learning could be added to existing LfD algorithms to allow the robot to adapt to the situations by itself and search for optimal solutions based on given demonstrations. Human demonstrations could serve as a start point for RL thus shorten the searching time for RL. Ideally, the robot becomes an intelligent assistant to the human operator through its learning and adapting abilities.
Based on these over all goals, we have the following objectives:
1. We want to setup teleoperation of nursing robot from Mocap system, here we need to solve data transmission problem and correspondence problem.
2. On the algorithm side, we need to investigate and implement learning from demonstration algorithms such as DMP, which enable robot to learn reaching to grasp tasks with obstacle avoidance, especially in cases of bi-manual coordination.
3. Moreover, we need to investigate how to integrate LfD with reinforcement learning (RL), which enable robot to explore manipulation motion based on the sub-optimal human demonstration.
The remaining sections is organized as the following: Section 2 gives a literature review of basic concepts and algorithms of LfD, including a brief history, the LfD pipeline, DMP and GMR algorithms. Section 3 talks about proposed algorithms of our project. Section 4 talks about our

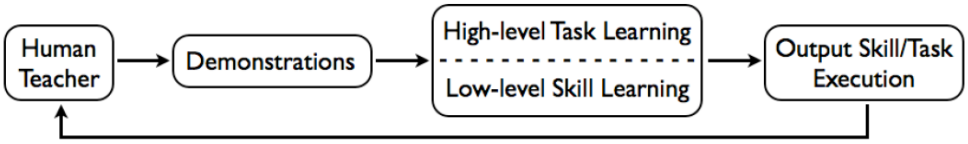


Figure 1: Learning from demonstration pipeline.

experimental setup, experiment results and data analysis.

2 Literature Review

2.1 Brief history

Robot learning from demonstration(LfD), or programming by demonstration(PbD) arises in early 80's [5]. At the every beginning, most work uses a recode and playback method. When considering the variability of human motion and sensing errors, people want to have some generalization of the motions from demonstration. Then there comes the symbolic method, in which actions are encoded by rules such as "left of", "close to". This gives us some kind of generalization but not enough information about the complete trajectory of motions. For this purpose, probabilistic models and dynamic models are being studied, including Gaussian Mixture Regression(GMR), Hidden Markov Models(HMMs) and Dynamic movement primitives(DMPs).

2.2 LfD pipeline

First, we want to frame the work of LfD. Figure 1 is a simplified version of LfD pipeline [6]. The human teach block includes the interface setups, and other human-robot interaction related issues. The demonstration block is the process where human subject demonstrates the task and the movements data is recorded. Generally, there are two kinds of demonstration methods: learn-by-doing and learn-by-observation. The key difference between the two methods is that, learn-by-doing uses data acquired from the robots own sensors so that there is no correspondence problem in the learning process. Demonstrations using joysticks, teleoperation, kinesthetic teaching all fall into this category. On the other hand, learn-by-observation means that the robot is not simultaneously moving when human perform demonstrations. Human demonstration data is recoded through accurate sensing. There would be correspondence problems in the learning process. Also, depending on the raw data, several procedures might by needed, including noise canceling, feature selection, segmentation, data normalization between different trails, etc. After this process, the resulting data is what can be used for the robot to learn.
In the big picture, the learning algorithms can be categorized into two parts, the low-level motor/trajectory learning and the high-level decision/plan learning. For the low-level learning, the robot learns motor skills, which are motion units, they are the bricks for more complex tasks. In high-level learning, the robot learns how to make decisions and plan for the next step, that is, how to use the learned motor skills to complete more complex task. This algorithm part is our primary focus.
After the learning process, the robot needs to reproduce the learn skills. There might be a feedback process, where the robot ask for help from the human teacher.

2.3 Motion modeling

There are two main categories of movement modeling: 1. probabilistic models and 2. Dynamic movement primitives.

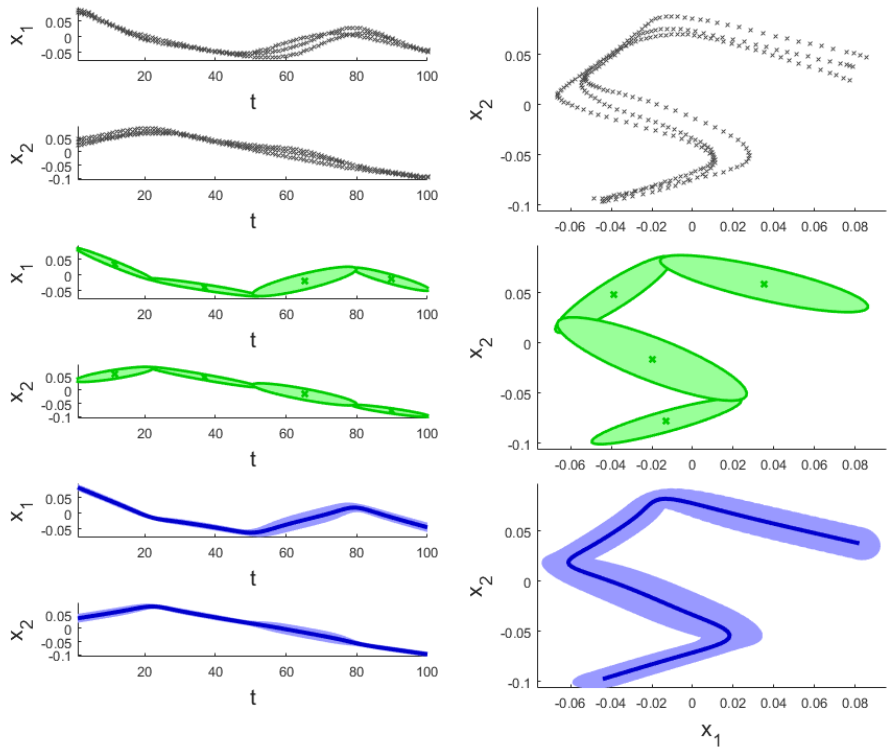


Figure 2: From GMM to GMR

2.3.1 Probabilistic based models

The most important motivation for having probabilistic based models is that they provide us a generic method to generalize from more than one demonstrations. Commonly used method includes GMR and HMM.

Gaussian Mixture Model
Gaussian mixture model(GMM) serves as a base for the GMR and HMM. The key of GMM is to use K Gaussian distributions to model probability of data points belong to the GMM model. The probability of one data point ξ_i belongs to the GMM is defined by

$$P(\xi_i) = \sum_{k=1}^{K} P(k)P(\xi_i|k) \tag{1}$$

A GMM model is described by $\{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{K}$, where π_k is the weighting of the corresponding Gaussian component, μ_k and Σ_k are mean and variance for the Gaussian component.
Batch learning of GMM generally uses a Expectation-Maximization(EM) method, the goal is maximize the sum of each of the data point's probability belonging to the GMM model. EM method is a iterative method, and the computing time is not negligible.

Gaussian Mixture Regression
Using GMM, we can have a probabilistic analysis of the data, but we do not have a trajectory. GMR is used to generalize a trajectory and its constraint that can be used by the robot for reproduction. The goal of GMR is to combine the components of GMM to form a single distribution. After we have GMM of a data set, the computation of GMR can be fast. Overall, GMR provides a fast and analytical solution to produce smooth and reliable trajectories from multiple demonstrations. There is no need to verify or smooth the results of GMR. Figure 2 shows the process of forming a GMR from GMM [4].

Hidden Markov Models
Hidden Markov Models has been used to many different areas other than robotics. In the context of LfD, a continuous form of HMM is discussed. Continuous HMM has three parameters, states, which is represented by a Gaussian distribution model $N(\mu_i, \Sigma_i)$, state transit, $a_i j$, which describes the probability of being at state i at time t, and being at state j at time $t+1$, starting probability Π_i, which is the probability of starting from state i.
Batch learning algorithm of HMM is called the Baum-Welch algorithm. It is also a EM based algorithm. HMM encapsulates spatial variability through Gaussian distribution and temporal variability through states transition. Compared with GMR, HMM cannot provide us a smooth trajectory result. It usually needs some post pressure to further smooth the trajectory to be reproducible by robots.
Probabilistic models, including GMR, HMMs, have been widely studied in literatures [2, 22, 25]. In the work by Sylvain Calinon[4, 5], GMR model is applied to a humanoid robots, the results showing that GMR models can be used to learn and generalize trajectory movements. Besides GMR, HMM is also a generative model that can provide robust reproduction. In the work of Aleksandar Vakanski [21], the notion of "key points" are used with HMM for robust trajectory learning. In the work of Gruger Volker [11], HMM is used to represent the discovered action primitives.
In some cases GMR and HMM are combined. In this work by Calinon's group [3], a combination of HMM and GMR is experimented on iCub humanoid robots, showing a capability of the system to handle several constraints at the same time.

2.3.2 Dynamic Movement Primitives

The initial work of the Dynamic Movement Primitives (DMPs) is published by Schaal [19]. The basic idea is to learn the motion feature for a controller to achieve a desired attractor behavior from a single demonstration. Here we choose a one dimension Mass-damper system to exhibit the basic approach of DMPs.

$$\tau\ddot{x} = K(g - x) - D\dot{x} + f(w, s). \tag{2}$$

Here, $x, \dot{x}, \ddot{x}$ are the position, velocity and acceleration of the system. g is the desired position, K and D are the gain coefficients, τ is the temporal scaling coefficient. Thus the first two terms performs as a PD controller to draw the system back to desired position. A forcing term f is added to modify the trajectory. The usual form of f function is the sum of multiple weighted Gaussian basis functions with phase s:

$$f = \frac{\sum_i \psi_i(s)\omega_i s}{\sum_i \psi_i(s)}. \tag{3}$$

The phase variable s has its own dynamics as $\tau\dot{s} = -\alpha s$, which decays exponentially from 1 to 0, guarantees the system will eventually attract to goal position. The Gaussian basis function ψ_i have distributed centers c_i at s domain, and variance h_i :

$$\psi_i(s) = exp^{(h_i(s-c_i)^2)}$$

Learning a DMP
First, By rearranging Eq. 2 we get the forcing term:

$$f(\omega, s) = \tau\ddot{x} - K(g - x) + D\dot{x} \tag{4}$$

where $x, \dot{x}, \ddot{x}$ are acquired from the demonstration trajectory. Then the problem reduced to learning weight coefficients ω_i to fit the given $f(\omega, s)$. The most commonly implemented method to calculate ω is Locally Weighted Projection Regression (LWPR) [23].
In order to alter the output of DMP, five parameters could be tuned: the number of basis function ψ, the decay term α, PD gain coefficients, temporal scaling coefficient and spatial scaling term $(g - x)$. Notice that the more number of basis function are chosen, the better f term will approach its desired one.

Since the first development of DMPs in [19], the author and his group have investigated further on updating DMPs and its applications. In 2008, [16] presented a framework of mid-flight avoidance to obstacles by adding a repellent force to the system and demonstrated it in different potential

field. Later in [17], the author modified the previous DMPs to overcome inherited drawbacks, including the system would remain at initial position when $(g - x) = 0$, small perturbation of g would results in dramatic acceleration change when $(g-x)$ is close to 0, as well as trajectory mirror issue when $(g - x)$ change its sign. Recently the group published a probabilistic representation of DMP in [14], where derivation, learning and executing Probabilistic DMP was discussed.
During the decade, DMPs have proven its generalization, robustness and ability to imitate from human demonstration in different areas. [26] achieved DMP in a leader-follower configuration, and the relationship between witch is explicitly represented. In [15], A force feedback based on DMP is utilized on a novel approach of upper-limb stroke rehabilitation, which allows a easy and flexible rehabilitation exercise. DMP is also applied to soccer-playing humanoid robot in [1], where in specific the kick motion is described by DMP.

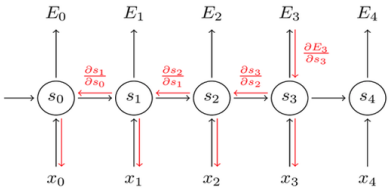
2.4 Policy Models

Policy models help us in system control by abstracting a process as a set of states with transitions between them. Every state has an output, and transitions are based on the current state and the current input. This is the essential definition of a Markov process. Reinforcement learning uses the Markov Process to find the optimal policy to select the actions used in a particular tasks. However, these models can be quite shallow in their inference and thus their transitions. Basing transitions off of not only the last state, but also some other states from the past would allow for more complex state transition relations.
LSTM networks would thus be able to learn more complex relations between states, by making use of data from multiple past states and non-linearities native to neural networks.

2.4.1 LSTM Networks

LSTM networks are essentially neural networks with knowledge of past states. They are trained by employing methods involving the unrolling of the neural network, or training through time.

One method of training is back-propagation through time: We can employ LSTMs to learn



Backpropagation Through Time

complex tasks both in the lower and higher strata of policy models. At the lower level, LSTMs can learn movement primitives from demonstartion data. LSTMs can provide appropriate control inputs for a given task by learning a pattern from the data provided by teleoperation.

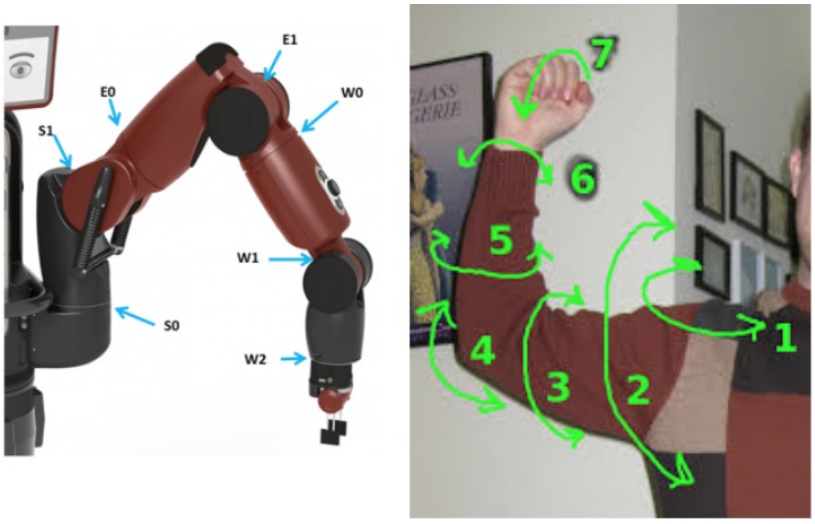
[6] mentions a novel method of conglomerating low-level actions together to obtain a Finite State Automaton, which has to be learned. The model used in [6] is quite effective and can possibly be improved by using LSTMs.
[6] further expands on the idea by mentioning goal-based learning which uses Reinforcement Learning techniques. Inverse Reinforcement Learning techniques are also presented. However, due to a lack of data this may be infeasible.
Robots can also be taught high-level states through affordance learning, i.e. classifying objects by learning interactions available with them.
We can make use of LSTMs to model high-level policy models. In particular, they can be used to learn task plans, or transitions between states.

2.4.2 Reinforcement Learning

Reinforcement learning is like learning via feedback from the environment in the form of rewards and penalties. The agent in the environment updates the set of actions to be taken or the policy depending on the estimated reward from the specific policy. One of the challenges of learning from demonstration is that no guarantees can be provided for the quality of the demonstrations, and thus the learned behavior. Also,a limiting factor in reinforcement learning as employed in artificial intelligence is the need for an often prohibitively large number of environment's samples before the agent reaches a desirable level of performance. While imitation learning provides an excellent means to start a movement skill at a high performance level, many movement tasks require trial-and-error refinement until a satisfying skill level is accomplished.[20] There are different ways of achieving this. Some of them are, input from human with the help of interactive shaping, guidance from human teacher by natural language advice and learning by human feedback (TAMER)[13, 24]. After learning to perform a particular action a control strategy is needed since high gain control is not a viable alternative due to frequent contacts of the robot with an unknown environment. Motion planning in high dimensional motor system is challenge. The complexity of the humanoid robot and the huge number of DOF's make it difficult to plan the trajectory and motion, especially when it comes to real-time performance on a reactive robot. Hence, here we propose the use of theory of control, planning, learning and imitation with motion primitives. Dynamic movement primitives has a potential in planning, movement recognition, imitation learning and general reinforcement learning.
Reinforcement Learning can be combined with dynamic primitive models[20]. From the viewpoint of DMPs, we need non-supervised learning methods to further improve the weights w=wi, guided by a general reward or optimization criterion. We can use the Natural Actor Critic (NAC) algorithm for the reinforcement learning of DMP's. The NAC is a stochastic gradient method, hence it injects noise in the control policy to provide necessary exploration of the learning. We basically use NAC algorithm to optimize the DMPs. Given a reward at every time step of the movement, the goal of learning is to optimize the expected accumulated reward where the expectation is taken with respect to all trajectories starting at the same start state and following the stochastic policy.

2.5 Evaluating a LfD Study

Last but not the least, we need to verify and validate the performance of our robot system. The first thing for us to do is formulating the hypotheses in order to find out what independent and dependent variables are necessary to address the hypotheses. Another thing we need to consider is how participants will be involved in our experimental conditions. One option is a between-subjects design, where each participant will only do one trial from all conditions. The other option is a within-subjects design, where each participant does all experimental conditions. The decision of which choice to use often comes down to which option will lead to a better controlled experiment, which is of vital significance.
To evaluate the algorithmic performance, we could use standard Machine Learning performance measures as dependent variables. Such as accuracy, error rate, and so on. Additionally, to measure how well our robot system performs, it is also important to consider what it takes to achieve a certain level of performance. The primary measure of importance is the number of training samples we need to provide for training. From a theoretical perspective, sample complexity is used. For algorithms that refine the policy over multiple iterations, such as reinforcement learning, the convergence rate of an algorithm is often reported as an empirical estimate of sample complexity for a particular domain.
In addition to evaluating the performance of our algorithm, we need to evaluate the interaction. There are two main type to implement. First, subjective measures include any measure in which you ask the participant to report their subjective experience of the interaction, e.g., a survey. Second, objective measures, such as sample complexity and precision/accuracy of the learned model over time if learning is occurring incrementally during the interaction.
In ideal situation, we hope all other factors, e.g. task, experimenter, practice and familiarity, in an experiment will be controlled, so that the only thing that distinguishes the difference between experimental conditions in our study is the manipulation of our independent variable.
For dependent measures, we make use of parametric methods to analyze data. Through these methods we expect to model the distribution of the data we collected to infer whether there is a significant difference between the distribution of the data measured from different experimental



Mapping Chart	
S0	1
S1	2
E0	3
E1	4
W0	5
W1	7

Figure 3: Mapping on joint angles of human and robot

conditions. And finally, the result of the inferential method will offer us a p-value and a statistical value.

3 Methodology

Following the LfD pipeline, the first problem we need to solve is the how to perform the demonstration and how to get the data for the robot to learn. Here we propose to use learn-by-doing method through teleoperation using Mocap. With learn-by-doing, as stated in section 2.2, there is no correspondence problem in the learning process because all the correspondence problem need to solved in teleoperation.
On the learning algorithm side, we use a combination of GMR and DMP for trajectory learning from multiple demonstrations. GMM and GMR give us a mean trajectory from multiple demonstrations. DMP uses this mean trajectory to learn. We use this combination method mostly because traditional DMP cannot learn from multiple demonstrations. But at the same time, DMP is a good modeling method, because DMP can generalize reproductions to new goal positions and at different speeds from single demonstration. Moreover, DMP is designed to model human motor control, it is a good way to generate human like motions.
We organize our methodology into two parts. The first part is on teleoperation using Mocap stated in section 3.1. The second part is on learning methods discussed in section 3.2.

3.1 Teleoperation

We want to use the Mocap system to teleoperate Trina. There are 7 degrees of freedom on each robot arm and 4 degrees of freedom on each hand. With Mocap system, we can get position information of markers put on the human arms and hands. The goal is to find a mapping between movements of human arm and robot arm. The mapping needs to be intuitive for the human operator to perform dexterous tasks, thus shortens the learning curve for novice users. The mapping also needs to produce movements on the robot arm that looks similar to the human movements. All of these depends on how we solve the correspondence problem.
Correspondence problem exists because there are physical differences between the human arm and the robot arm. There are many ways to solve the correspondence problem. One way to do it is to do mapping in joint space, calculate the joints angles from human motion data and match it to Baxter [18]. Another way to do it is to match end-effector positions and solve inverse kinematic problem with constraint so that the robot arm performs in a way similar to human postures [10]. We decided to use joint angle mapping for teleoperation. Human joint angles are calculated from marker positions. These joint angles are mapped to Baxter's corresponding joint angles. See figure 3 for detail angle mappings.

3.1.1 Hand mapping

For our learning from demonstration experiment, we will need to map the human hand to a three fingered robotic hand. This needs us to figure out the synergy of the human fingers while performing a grasping task so that the robot arm can effectively grasp the object despite having lesser degrees of freedom. The main approaches that have been used in the past to deal with this problem are three: joint-to-joint mapping, fingertip mapping, and pose mapping. One more approach to solving the mapping problem is to map human joint synergies to the robot hand. We will be using this approach, as it is more general and provides flexibility with regards to how many fingers the robot hand can have. This approach does have some limitations like the fact that the object shape needs to match the finger placement geometry for the object and that the mapping is defined for uniformly squeezing and moving an object. Given the needs of our project, these issues can be trivialized.

The process to be followed is primarily to first capture the human motion data for grasping tasks with high fidelity. This would be achieved with using Kinect RGB-D data in tandem with the Vicon motion capture system. The fusion of these two sensor data would enable to create a high fidelity skeleton of the human hand. The reason for following this approach over a glove based system is because a glove based system may lead to a lot of noise and thereby would need manual cleaning of data. It also has the problem of not providing the absolute position of the arm in the world frame.

The data to be collected for this experiment would be of grasping n everyday objects over five to six subjects. Three samples for each grasp would be taken to average readings and even out the noise. This would then be followed by performing principal component analysis over the set of grasping tasks done by the human user. The PC from here would then be used to define a paradigmatic hand which would then be used to map the human hand motion to the robot hand which has dissimilar kinematics.

3.2 Robot Learning

This is the major part for our project. The goal is to implement trajectory learning (low-level learning) algorithms on Trina, with RL method to refine grasping of different objects.

3.2.1 Implementation of DMPs to multi-dimensions

To implement DMP on Trina, the first thing we need do is to have DMP working in multi-dimensional cases. According to the one dimensional DMP code [9], we have implemented multi-dimensional DMP using the same canonical phase term and different transformation functions for each of the degrees of freedom. The results is showed in figure 4 using three dimensional DMP. Each horizontal line of pictures corresponds to each degrees of freedom. On each line, the first three pictures represent position, velocity and acceleration respectively. The fourth picture is showing the change in forcing term and the last picture is the basis functions. A 3D view of comparison of the demonstration data and the reproduction of DMP is shown in figure 5. The blue line represents the demonstration data and the black line represents the reproduction from DMP. It can be seen in the pictures that DMP works well in multi-dimensional cases. If we change the goal position, DMP reproduces the trajectory similar to the demonstration that can reach the new goal position. The code can be easily extended to work for more than three degrees of freedom.

3.2.2 The use of DMP on multiple demonstrations

Get trajectory from demonstrations with GMR
In our tasks, marker positions are acquired by tracking human movement with Vicon Mocap system, and smoothed by Kalman filter. Human joint angles are calculated from marker positions. Six of human joint angles will be mapped to six of Baxter's joint angles. Baxter's joint angles trajectories are recorded. Here we are going to start with a toy problem to demonstrate our algorithm. As shown in figure 6, we collected 5 2-D trajectories using graphic tablet, which are the demonstrations for later learning process. Notice that the units of x and y positions are the pixels of the graphic board.

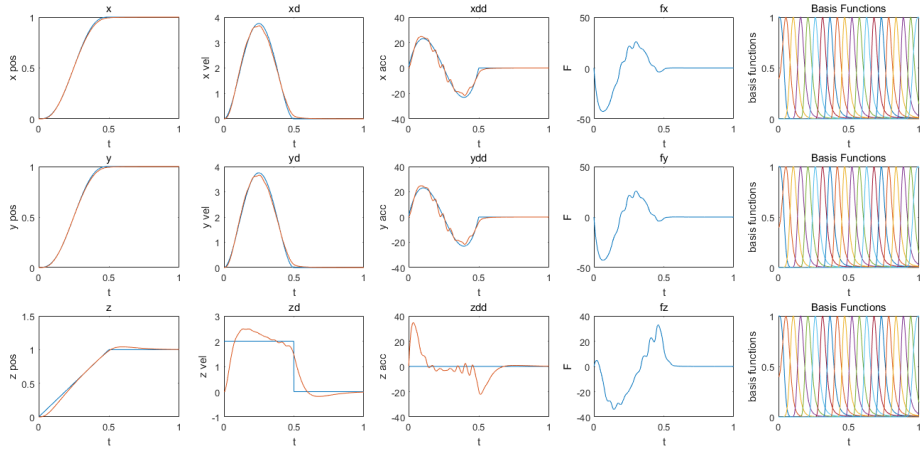
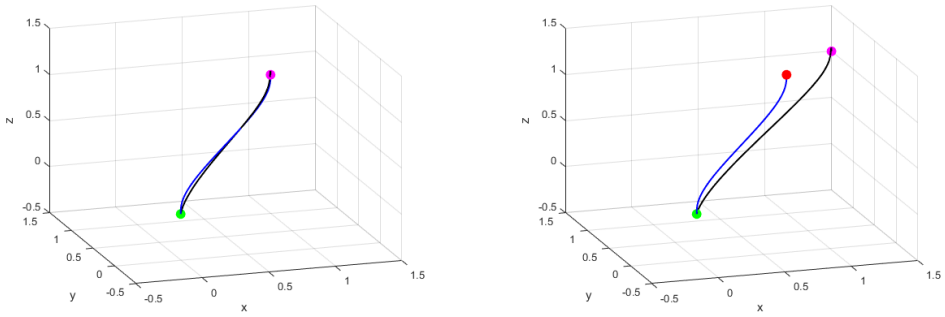


Figure 4: Three dimensional DMP demonstration



(a) Reproduction on the same goal

(b) Reproduction on a different goal

Figure 5: Direct view in 3D

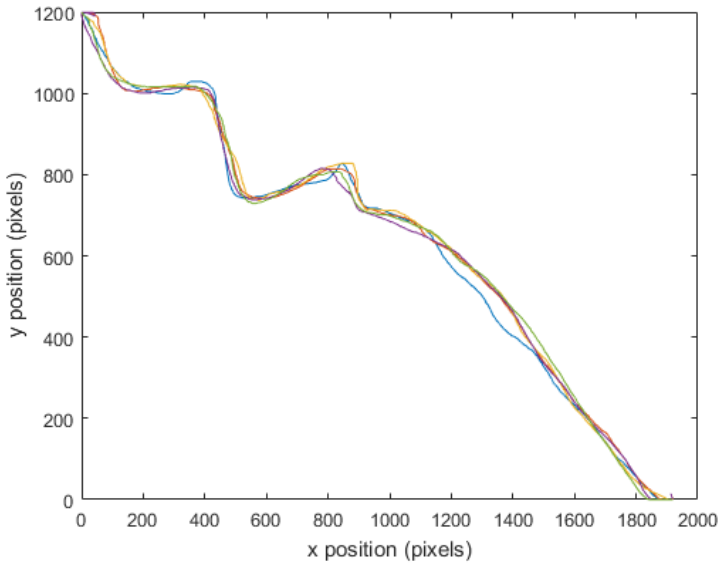


Figure 6: Multiple demonstrations for DMP to learn.

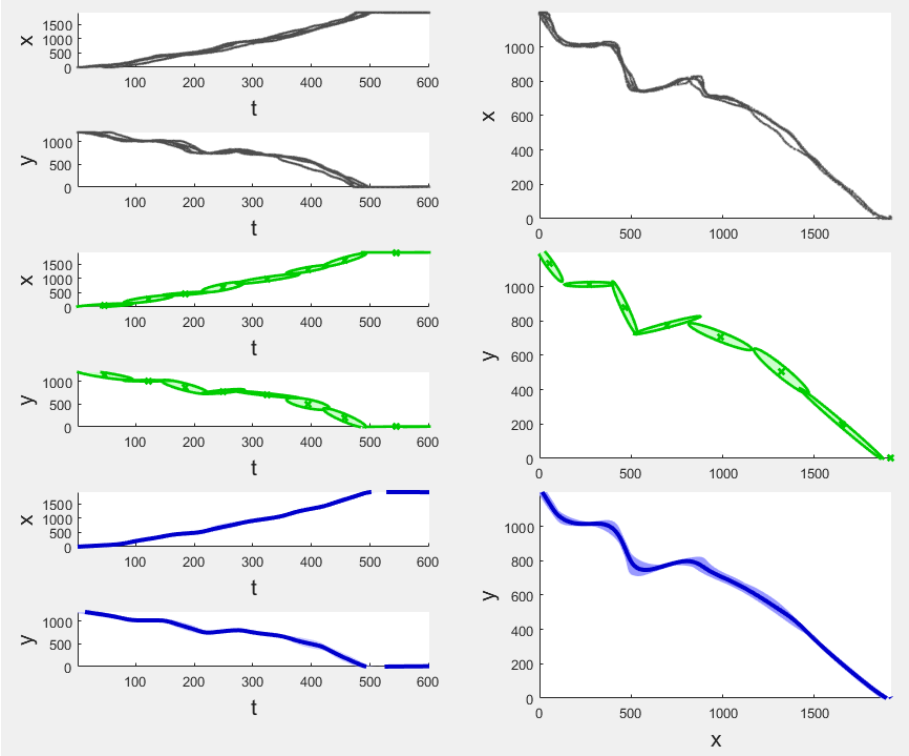


Figure 7: Results of GMM and GMR.(the left column are the decomposed trajectories at each dimension; Black plots are the demonstrations; Green plots are the GMM clustering results. Blue plots are the GMR regression results)

Calculate the forcing term from trajectory

First, In order to learn DMP from multiple demonstrations, our approach is to use GMM/GMR to find the mean trajectory of multiple demonstration to feed the DMP learning process. At the beginning we learn the GMM of the trajectories, and then based on the GMM results we learn the mean trajectory by GMR. As shown in figure 7, the Green plots are the GMM of multiple demonstrations, and the blue plots are the regression trajectory with the variance presented as the shaded area. As shown in equation (4), forcing term can be acquired by rearrenging the DMP dynamics. In our case, since we already have the end-effector trajectory from human, we can easily calculate the $x, \dot{x}, \ddot{x}$ in time sequence. Gain coefficients K, D can be tuned as same as the PD controller tuning. temporal scaling τ and spatial scaling $(g - x)$ can be tuned by actual demands on tasks speed and tasks range. The number of basis function ψ can be determined by balancing the trade-off between the approximation of the forcing term and the learning complexity.

Learn the weights

Locally weighted projection regression (LWPR) will be used to learn the weights of basis function ψ. Different from the peer DMP researches that the basis functions are evenly distributed on the canonical system, the locations of the basis functions in our approach are determined by the GMM results. In detail, the center axises of the basis functions (the Gaussians) are respectively aligned with the centers of the confidence ellipsoids, which is known as the GMM results. One benefits of such alteration is that, during the whole trajectory, those parts with more dense features will be synthesized by more dense weighted basis functions of DMP, since such parts will have more dense confidence ellipsoids from GMM and more dense basis functions of DMP will be aligned.

The learning results are presented in figure 8. In the position, velocity and acceleration subplots, we can see that the reproduced trajectories (orange) align well with training data. The errors can be reduced by increasing the number of basis function, which in this case we use 16 basis functions. In the basis function subplots, it can be observed that the Gaussian basis functions are not evenly distributed.

Derive new trajectory

The first step is providing the DMP with the initial position of robots x_0 as well as the goal position g. The initial position can be read out from ROS, the goal position can be provided by graphic

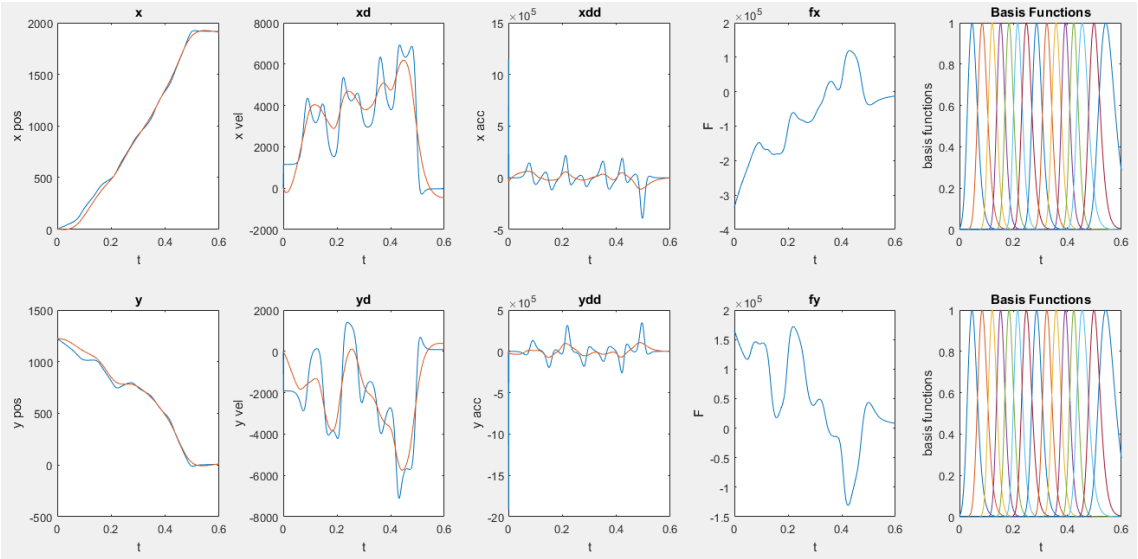
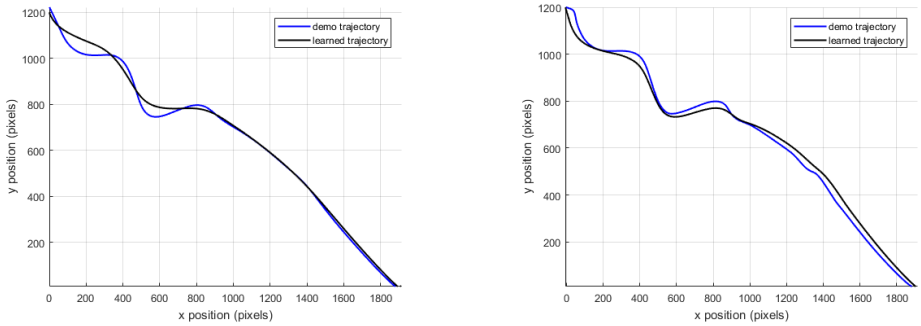


Figure 8: Results of learned DMP.(The top row and bottom row respectively represent x and y axis; The subplots from left to right are respectively position, velocity, acceleration, forcing terms and basis function distribution; Blue line represents the training trajectory and the orange line represents the reproduced trajectory)



(a) 16 Basis functions

(b) 40 Basis functions

Figure 9: Learned trajectories based on different numbers of basis functions

identification from Baxter. After the initial position and the goal position pluged in the DMP, DMP would return a new trajectory with desired movement features learned from demonstration. In figure 9 the learned trajectories are presented. It can be observed that with increased numbers of basis functions used during learning process, the beginning part of the learned trajectory is more approximate to the demonstration, however the ending part of the trajectory lose some certain of approximation.

3.2.3 Learning of different grasping mode with RL

After the robot learns the reaching movement from DMP, the next step is to learn how to grasp. The paper by Personal Robotics[8] provides an adequate solution to the grasp-detection problem, thus allowing us to spend more time on training grasps. It has been tested and verified to work on a Baxter robot and thus can confidently be used for our purpose.
After detecting the grasp for the object and creating the bounding box for the grasp of the object, reinforcement learning can be used to manipulate the controls of the robot to actually grasp the object. The robot hand that we use has three fingers. Initial position of the three finger will be used to create a new bounding box. This bounding box will be the box that should be matching the grasp bounding box. If the two bounding boxes overlap then an accurate grasp can be implemented. The more overlapping of the the two bounding boxes the better. As the three fingers can be positioned accordingly to grasp the object. To complete this task Reinforcement Learning can be used.
Using reinforcement learning the two bounding boxes can be made to learn to overlap each other so that the grasp is accurate. The procedure for this task would be something like this, first the object will be detected and a bounding box would be made around it's grasp. The vision frame would be the environment of the reinforcement learning agent. Another bounding box, or the bounding box of the gripper would be a part of the environment. The grasp bounding box would be final position of the goal of the agent. Maximum reward would be given for obtaining this position in the environment. The position of the gripper bounding box would the state of the agent. Movement of the bounding box would be the actions of the agent. The maximum points of the gripper bounding box would be the state of the agent. Movement of the bounding box would be the actions of the agent. When maximum points of the gripper bounding box overlap on the grasp bounding box, it would be considered the optimal policy. If not actions can be taken to change the positions and orientation of the gripper bounding box, which in turn changes the robot hand position, so that the gripper can grasp the object accurately.

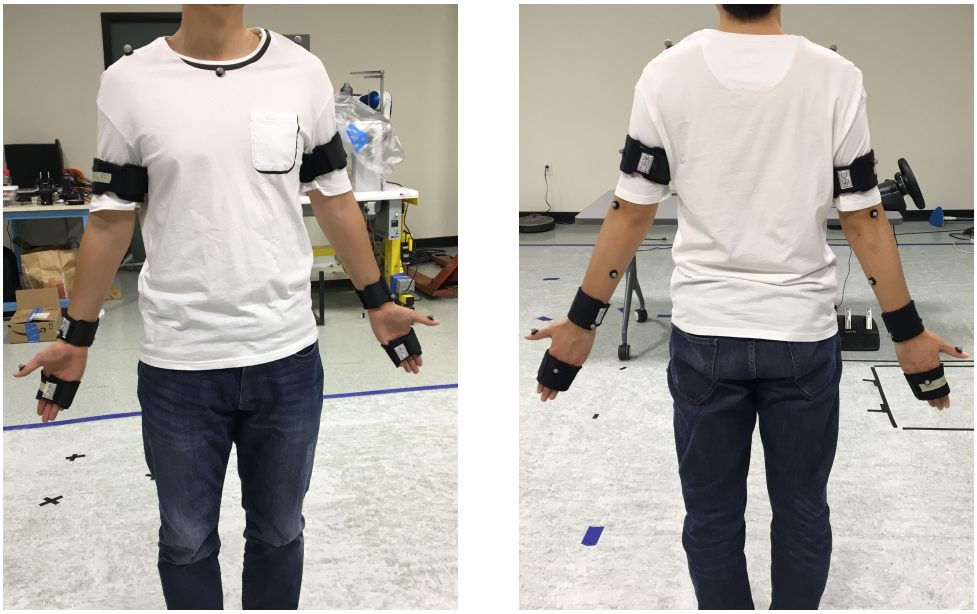
4 Experimental Results

4.1 Experiment Setup

The experiment has two parts: 1. teleoperation of Baxter using vicon motion capture systsm. 2. use the collected data from Baxter to learn DMPs for each joint and reproduce. The following subsections describe the hardware setups of our system.

4.1.1 Vicon System

To set up the Vicon system through Nexus2.5, there are three main steps that we followed in Vicon manual. They are system preparation, Creating a Vicon Template, Creating a Vicon skeleton. During system preparation, we need to adjust the position, height and orientation of that two movable Vicon camera standing on the ground to focus the key motion area of the subject, which is good for making the best use of their vision. For the same reason, the subject's motion range had better within the central part of the whole motion capture area.
In the second step, because of our specific requirements, we customized our own template, other than its plug-in template. The marker placement of our template on human subject is shown in figure 10 When we create our template, we write down the specific displacement between neighbor markers to make sure we can get a relative accurate mark placement next time. What's more, we have some asymmetry in the marker placement between left and right arm. This is very important for Vicon recognizing and automatic labeling. Another thing is that, we put bigger marks at the points where the mark may be occluded easily during task motion. For the accuracy issue, we prefer to create a new template for a different subject. In the third step, we create a skeleton based on the template we just created. And then, this is where we can start to record the motion



(a) The front (b) The back

Figure 10: Marker placement

of our subject. An interface of Vicon Nexus2.5 in shown in figure 11 For our subject, vest or tight T-shirt is preferable, which plays an important role in getting stable marker position.

4.1.2 Transparency

According to our test trials of the teleoperation, one big issue is the transparency problem. That is, the operator cannot see Baxter's movements to decide where the arm should head for. To solve this, we mapped Baxter's head camera to a screen, and put it in the view of the operator. See figure 12 for the experiment setup.

4.1.3 Experimental Protocol

The purpose of this experiment is to collect data for Baxter to do batch learning of DMP. We utilize Vicon motion capture system to capture human motion to perform teleoperation on Baxter. Until this moment, we have finished two tasks. The first is 'Reaching'. The human subject teleoperates one Baxter arm to reach to one fixed point in Baxter's working space, and for 10 trials. The second task is 'Bimanual hold'. The human subject teleoperates Baxter left arm and right arm to hold up a empty box.
We collected three set of data. The first is the recorded trails of the "reaching" manipulation in the Vicon software system. The second is the set of recorded joint angles we calculated from the markers as human joint angles. The third is the set of joint angles of Baxter.

4.1.4 Data transmission and Data processing

1. For real-time data transmission between the vicon system and the trina PC, we have used vicon-bridge. It is a ros package that connects to the vicon system over the local network; for this, the vicon Nexus software was set-up with Data Streamer SDK

2. The joint-state data from Baxter is noisy and thus needs to be filtereed to be usable. We implemented a Kalman Filter that subscribes to the joint-states and publishes Kalman-Filtered data with negligible delay

3. For moving the arms to the default position, we also developed a program that moves both arms

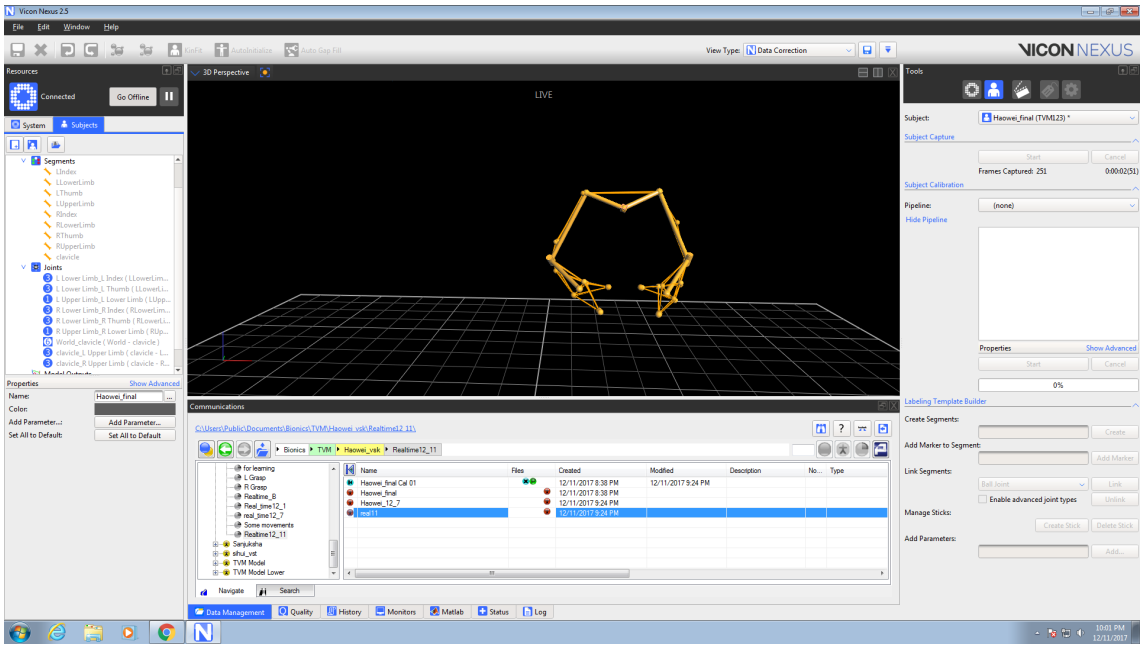


Figure 11: Interface of Vicon Nexus

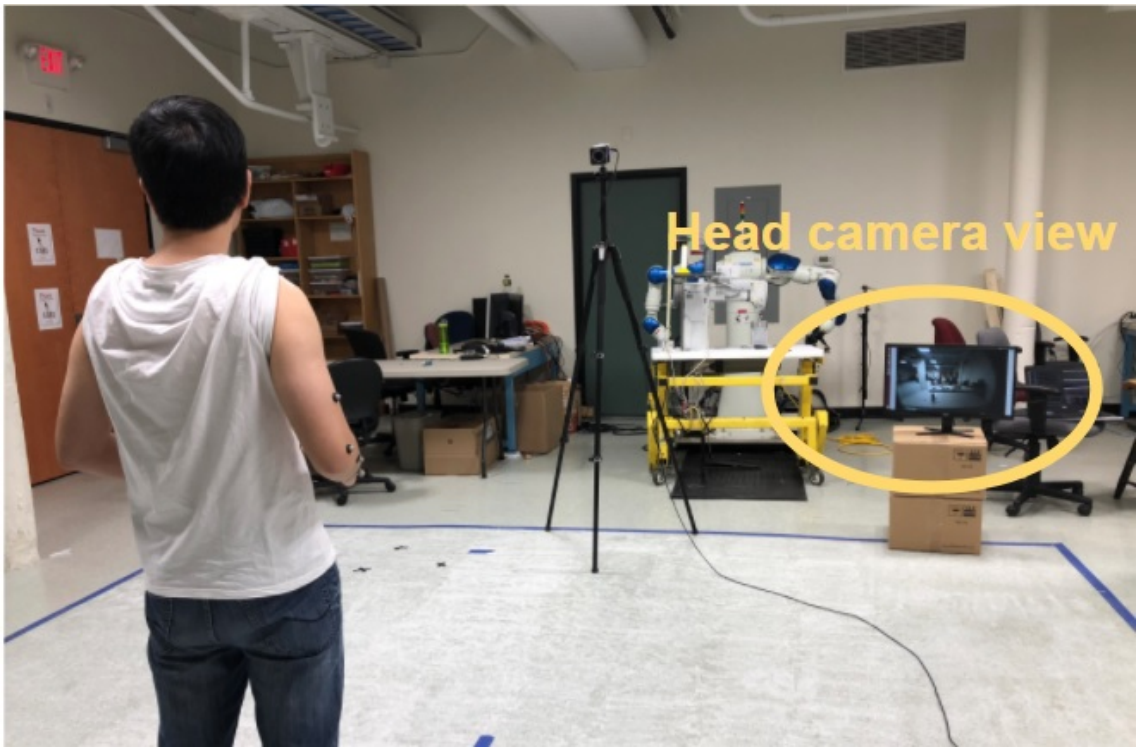


Figure 12: Head camera setup

4.1.5 Mapping and teleoperation

When mapping motion between Baxter and a human-subject, one should consider the anatomical difference betwen the Baxter arm and the Human arm. Baxter's arm anatomy is not anthroplogically homeomorphic. The deviations from classic human anatomy occur at the elbow joint, in that it is inverted; and at the wrist, in that final degree of freedom offers no twist about the wrist, but a rotation about the lower arm axis.
For the problem of mapping human motion to baxter, we considered various approaches. The first approach was to calculate human joint-angular velocities and set baxter's joint-angular velocities to be the same. This method was obviously not a viable method as the joint-angle velocities were:

1. Noisy
2. Suffered the effects of euler-approximation
3. The model itself suffered from angular-drift, due to a lack of position feedback

Essentially, it made more sense to make use of absolute joint-angle positions and map those to baxter's joint angles. The primary challenge in this case was the inverse-kinematics of a human arm, which proved to be quite challenging. The noise that was characteristic of the system also further complicated the system and was filtered out using a simple digital FIR-filter and low-passing only the relevant frequency components.
The criteria for deciding a valid signal bandwidth was done by observing the frequency spectrum in the Discrete-Fourier Domain. We were aware that by setting reasonable limits to the rate of change of human joint-angle variation, we could set a somewhat less arbitrary signal bandwidth. From observation, one can claim that the maximum frequency with which one can swing any joint to be around 10 Hz, allowing for two times this value to reasonably allow higher harmonics, one could arrive at a proposed low-frequency signal bandwidth of 20 Hz. From observation, we can see that some higher frequency components lie much further from the reasonably set frequency bandwidth derived from the working approximations above. These can be safely assumed to be random high-frequency noise, and be discarded.
After having filtered the angles values, the simply commanded to the Baxter arm.

4.2 Experimental Results

With the teleoperation setup, we successfully conducted the reaching movement and also tried out bi-manual manipulation. Click here for the video record.
After we got the 10 demonstrations, 7 are chosen for learning because of available data qualities. GMM and GMR is applied on each of the 6 DOFs with these 7 demonstrations. The trained GMM and GMR results are show in figure 13.
DMP is learned for each Joint in Baxter. See figure 14 for the plot Reproduction results show that when the goal position is changed, the robot's arm can still get to the new goal position. Click here for the video record.

4.2.1 Human Subject Report- By Haowei Zhao

We had totally 3 trails to test the mapping of Baxter.
First trail: Different joint range and initial position made the motion correlation between me and Baxter awful. The shoulder0 would reach the limit of Baxter when I reach my limit but in this situation the joint angles were not the same. There were some joints that didn't response in right way, Which were wrist0, wrist1 and elbow0. When I moved downwards my wrist continuously, the wrist1 of the Baxter would move downwards at first, but inverse after a certain angle. It was also the case if I move wrist upwards. And shoulder0 was correlated with elbow0, which mean if I only want to move the shoulder0, my motion would make elbow0 move as well. It was the same for wrist0 and wirst1. All the problems mentioned above make teleoperation teaching impossible.
Second trial: After the improvement made by Sihui, who set the joint limitaion aligned with human limitation, the mapping of human became far better. Most of the joint worked well, but the correlation between wrist0 and wrist1 was not fixed. But this only happened when I was near my joint limitation so some task could be done away from the it. The wrist1 had the same problem as last time. I tried to touch the cup on Yudong's hand, which was a little bit difficult because I could not see things from Baxter's view. It made me hard to determine the distance between

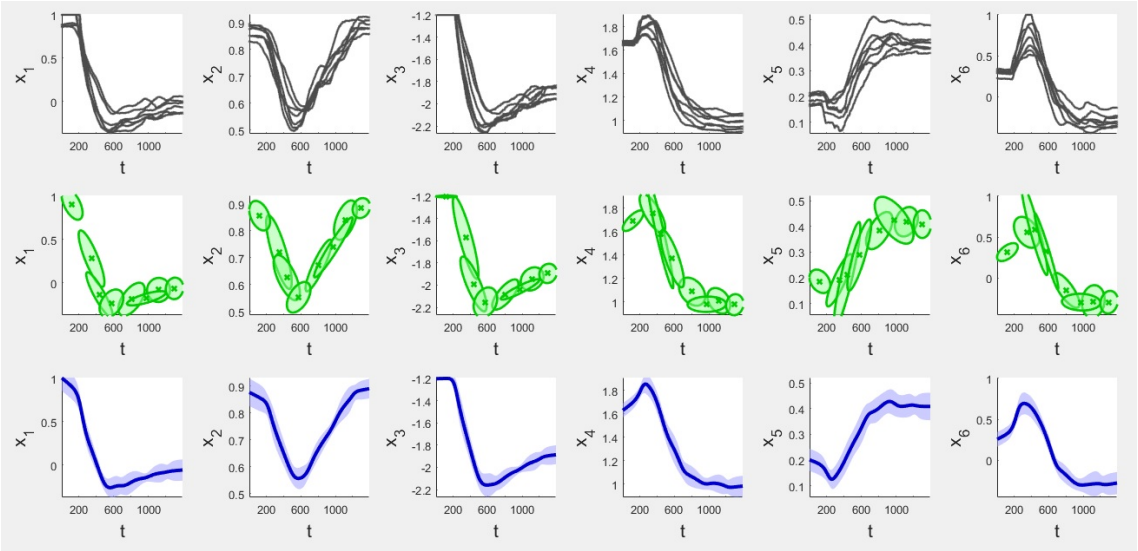


Figure 13: GMM and GMR results on each DOF: first line is data plot, second line is GMM results, third line is GMR results.

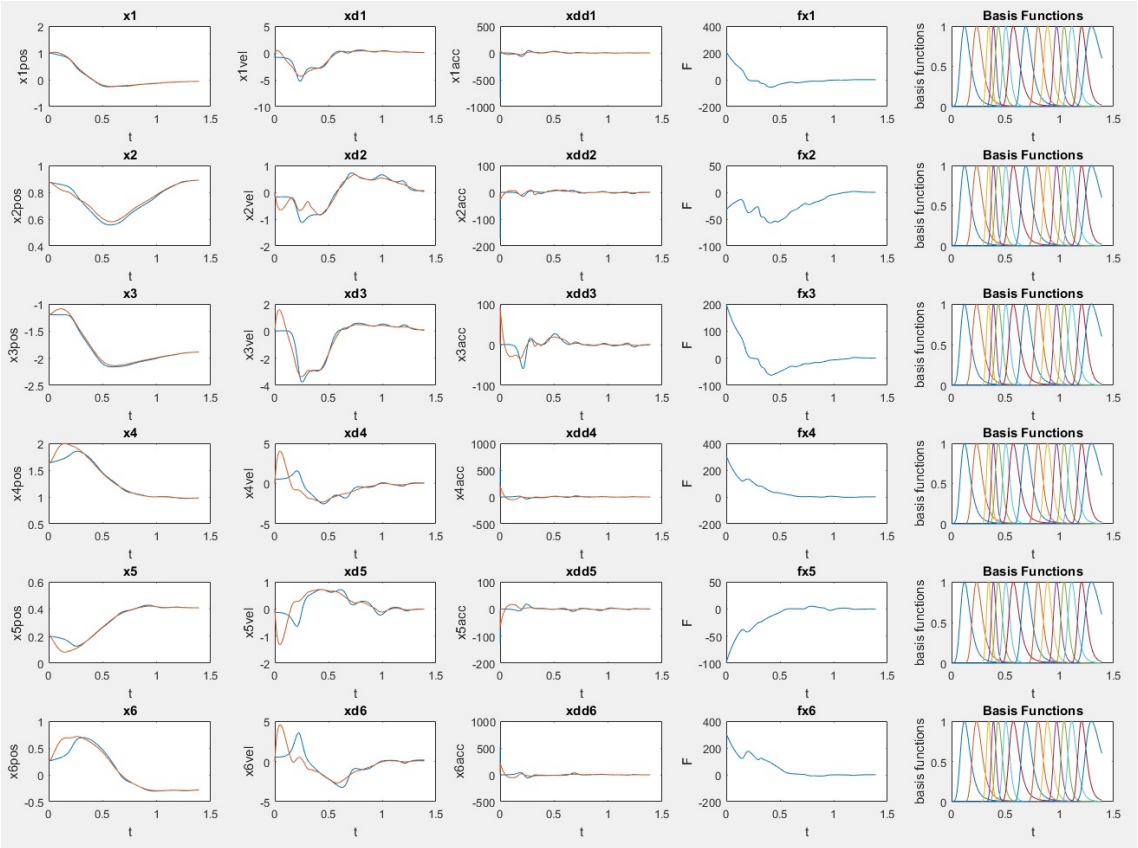


Figure 14: DMP learning results for each DoF.

Bexter's hand and objective.
Third trial: After fixing the problem of wrist, the mapping became intuitive.Helped by the visual feedback from the front camera of Baxter, I could do the bimanual pick-up tesk without practice. But there were some joint correlations, like elow1 and wrist1 which can be seen in the video when I was waving. Meanwhile,the orientation of the wrist was not right for most of the time. If I want to grasp something, I would keep the axis of my wrist perpendicular to the table. But when I was perpendicular, the wrist1 of Baxter was not. This makes the mapping of accurate hand-arm motion impossible. Moreover, the lacking between my motion and Baxter's bothered me.

4.3 Data Analysis

We have collected one primitive movement and the primitive is recorded for several trials from motion capture system. Therefore we can not only build a primitive library for high-level learning, but also compare the data from different trials and scenario to learn the features of human and robot motion. However the raw data are scattered and disconnected, so that data processing is required to get clean and meaningful data for analysis.

The data analysis process for this paper is decomposed to four parts. First we will mention how the data are collected from Vicon system, as well as the gap filling for discontinuous trials, before they learned by DMP algorithms. Next we will present how the scattered trails are matched and learned the weights values. Then we will present the results of a left hand motion and its weights. In the final part the results will be discussed.

Data preparation

In Vicon system, after the data is recorded and loaded, in the pipeline tools panes, an export ASCII operation is created. In the property section of the export operation, file name, file extension, first frame, last frame and numeric format need to be set. After the operation is executed, an csv file should be created in designated folder, including all the position of markers attached on the operator, as well as the joint angles that was pre-defined in the kinematics models in Eular angle form.

It can be observed from the raw data in csv files that the trajectories are sometimes discontinuous in some region and the lost part are shown as NaN, which is mainly because of the lost of visual detection during the teleoperation in Vicon system. In order to fill the gap, a penalized least squares method based algorithm is utilized. The algorithm was proposed to deal with occurences of missing and outlying values, which also allows fast smoothing of data in multiple dimension. More details and the code of this algorithm can be found in [7].

Among the demonstrations, Some trails are unqualified to feed the DMP learning. One principle is that the data cannot drift too much away from the average. The other principle is that the training trails cannot lose key features results from demonstration false or data filtering.

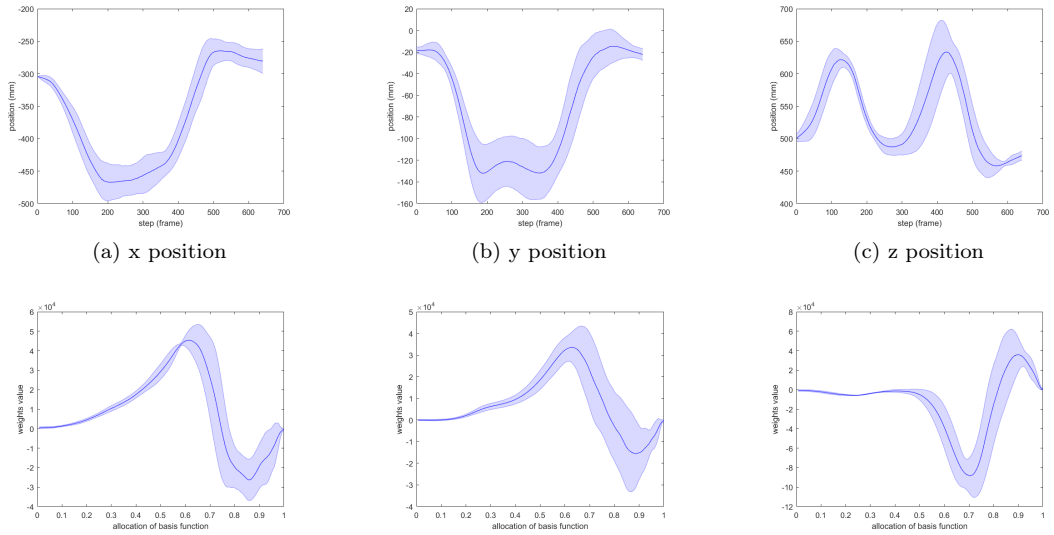
Data processing

Although the data is prepared, there are still two problems with the trajectories: the trajectories of one single motion primitive are scattered to different locations at space span and time span. In order to solve these problems, one trajectory is selected as a canonical trajectory, and the rest trajectories are relocated in both space and time span. The principle of data processing is to make sure the initial position of each trajectories are coincided, and the inflection points, which represents the features of the motion, are close to each other as much as possible. All the tuning works are finished by human labor.

Results

In the beginning 7 out of 10 reaching demonstration are chosen. The GMM and GMR results is presented in figure13. The results of Robot recored training trajectories and reproduce trajectories are presented in figure14.

The human reaching and withdraw arm motion is recorded for both left arm and right arm, and each have repeated four times. Figure 15.(a-c) presents the right arm wrists (considered as the end of the arm) position in 3-D space. The average values of four repeats is shown as solid line, and the standard deviation is represented as the shaded area. After the four repeated trajectories is



(a) x position (b) y position (c) z position

(d) x weights (e) y weights (f) z weights

Figure 15: hand trajectory and weights values learned from DMP; figure 5.(a-c) shows the operator's hand reaching motion in 3-D space; figure 5.(d-f) presents the learned weights value with respect to correspondent dimension. The solid line represent the average value and the shaded area represents the standard deviation

learned with DMP, four sets of weights is acquired, and the results are presented in Figure 15.(d-f).

Discussion

In figure13, the Gaussian Mixtures ellipsoids results are more disordered and overlapped compare to figure7. This is because that the 7 reaching training data is more scattered than the training data from the graphic tablet. However since we have 8 basis, which is much more than feature complexity of the training data, the regression results still remains reliable.

It can be observed from figure 15 that all the trajectories follows the similar trend, which can validate the fact that human have some regularities in motion primitives. However the trajectories are no precise, a certain degree of standard deviation exist, which means that even well learned human allows some errors in motion primitives.

Notice that near the start and the end of the trajectories, the Std is relatively smaller than the middle part. One possible explanation is that the middle process is not as much cared as the start and the end, which makes sense in such reaching and withdrawing tasks. However two possible kinds errors may cause the increased Std: the errors in both space span and time span.

4.4 Reinforcement Learning Simulation

In this project the final task is of grasping which can be achieved only by the movement of the hand. Here we use the three fingered hand from RightHand robotics called the reflex sf hand. The reflex sf hand consists of dynamixel servos with which the fingers are moved . Hence controlling the angle of these motors gives us a control of the hand. As our objective is to reach and grasp an object, in which the object is a cylindrical bottle, we use reinforcement learning to grasp the bottle with appropriate angle of the fingers.

As explained before, we use two bounding boxes, the object bounding box and the grasping bounding box to apply the reinforcement learning algorithm. The object bounding box can be obtained using the work in the paper Deep Learning for robotic grasps [8]. The resultant boundary would be a rectangle with 4 points, but in our case to create a gripper bounding box we have only three fingers. Hence, we make a symmetrical bounding rectangle using the three points we have from the finger tips. These finger tips points are obtained using visualization in rviz. The state of the hand i.e. open or closed can be controlled by the sinusoidal signal provided to the hand which further provides finger angle and the preshape angle. The Finger angle is the angle between finger and the base hence when finger angle is zero it only moves in horizontal plane and do not

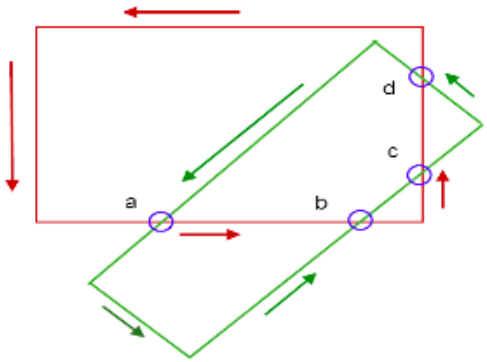


Figure 16: Overlapping bounding boxes

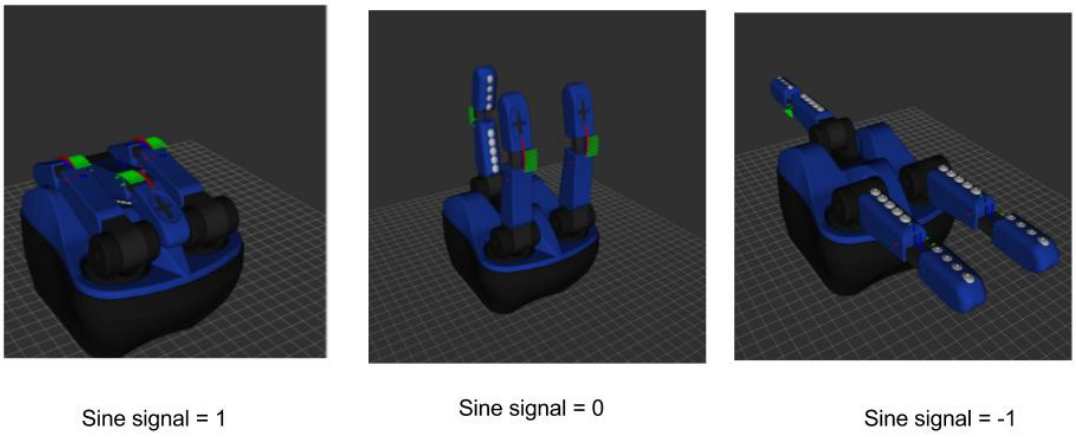


Figure 17: States of Hand at respective sine signals

close and open. The preshape angle is the one which makes it move horizontally, hence it is the angle between the two fingers. If the sine signal is greater than zero the hand closes and when its less than zero . Hence we take a range of sine signal from 0.5 to -1. In between these two states there are a total of 189 states which give different sine angle values. We have created a states.csv file which has indexing in the first column from 1 to 189 and the sine signal values in the second column. So when we identify the current state of the hand we find the current value of sine signal in the states.csv file and get the number of value in the list so let's say that the current state is 56th value in the states.csv Hence, we consider the current state of the system to be 56. The rest of the columns in this csv file give us the values of the points of the gripper bounding box. We consider these 189 states so that we can find the appropriate angle by which we can find the best angle of finger to grasp the object. This appropriate angle can be determined by overlapping points of the bounding boxes.

Initially we have a current state of the hand which gives us a bounding box points value only if the hand is open. If the hand is closed then a random action of changing the sine signal is taken and the current hand position is checked again. Once, the hand is open and we get some bounding box values of the gripper, we now take the input bounding box values which are the values of the object bounding box. We now take the ranges of x and y co-ordinates of both of these bounding boxes. So that we can compare both the ranges of the co-ordinates and find the number of overlapping points on the bounding boxes. If the number of overlapping points is greater than our threshold then we give the reinforcement learning agent a reward.

Finally, after certain number of episodes the state with maximum reward will be repeated and the grasp will be completed.

5 Future Work

Future work can be done in each of following sections.

5.1 Hardware setup

To extend from our case of the tele-nursing robot, visual sensors and LfD method working together also have the potential to tele-manufacturing robot and housing robot. Mocap system is expansive, and requires a long setup time and a large space, Kinect like sensors and motion gloves could be used as substitution for Mocap.

5.2 Learn Task Space

For this project, what we are learning is joint angle trajectories. For teleoperation, joint angle mapping can provide some level of intuitiveness according to our experimental results, but is not easy or accurate enough for dexterous manipulation. A better method for human to control Baxter's arms might be mapping in task space. Moreover, DMP learnings in joint space lost the point of the learning generic movements. The DMP framework is not only a trajectory reproduction and planning tool but also a way to record movement patterns. For this purpose, learning in task space makes more sense.

5.3 Learning Algorithm

On the algorithm side, online learning is a trend. With online learning, it is possible to make the learning process interactive with human input. For example, robot with previous knowledge can be customized with new human demonstrations, and at the same time accept the human critique of whether the movement is as desired or not.

5.4 Grasping

Grasping will be incorporated with reaching movement. Reaching movement is responsible for going to the right position and orientation, and grasping movements will use reinforcement learning to find the optimal grasping mode. Finger movement will be

5.5 Evaluation

We haven't finished the completed evaluation of our experiments. More subjects data are needed to make a complete evaluation of our system. We need to evaluate the teleoperation system on its ease of control, intuitiveness, comfort level and transparency.
For the learning part, we need to evaluate on the following:

1. Accuracy and Error Rate or Success Rate

   In the *Learning from Human Teachers*, accuracy is considered as the proportion of actions that are correct; its inverse, the error rate, describe the number of incorrect actions. In our experiment, we would evaluate the performance of the Baxter by measuring how much of the maneuver is successfully completed, which would give us the success rate.

2. Sample Complexity

   When comparing different methods, it is important to compare what it takes to achieve that level of performance. We would measure how much samples that we need to include in this algorithm to decide this. If only small amount of sample is required, then the algorithm is optimal.

3. Quality of the Demonstrations

   It is important to realize that every demonstration is different and always suboptimal. So these demonstrations would influence the result of the experiment. At very least, we must report the number and type of interactions the teacher had with the robot. Meanwhile we would measure the quality of the demonstration by measuring how well the human teacher can perform the target task, how long the teacher takes to finish, and how consistent the demonstrations are from run to run. If our algorithm only need suboptimal demonstration, then it is better.

4. Evaluating the Interaction

   In HRI, it is important to determine how the human feel when he or she is interacting with the robot, which in our case is to use the teleoperation to control Baxter. Most intuitively, this can be done by survey. We would design the survey using Likert-scale2 to measure the feelings like whether the teacher feels tired after teaching. Furthermore, to avoid the subjective distortion, we would let a third party person to watch the teaching video and then describe if he or she feels the teaching process is natural.

A Appendix: record of issues and solutions in motion capture system

1. Q: When recording the data, the makers are lost or flickering.

   A: At first, you may want to make the best use of Vicon cameras through changing the cameras' position, height and orientation to provide a better vision range. At second, When recording the data, try to slow down the motion and try to limit the range of the motion. If it is still does not work, recalibrate the system again. This solution is straightforward and we used it many times. For an optimal recording result, we suggest others to use the plug-in template or make changes based on it at least. Meanwhile we prefer to capture the motion from the human subject who is modeled as the template. And after recording, try to manually label the makers again and do gap fill.

2. Q: After go live, in 3D perspective, the cameras all disappear in the volume.

   A: Normally, this is because another account on this PC is running the Vicon Nexus. Contact the last person who used it. If it is not the reason, reboot the Nexus. If it is still nor working, please just restart the PC.

3. Q: While linking the segment with joints, the segments and joints in lists under the Subject tab may turn into grey and show that 'This segment has no kinematic to this frame'.

   A: Try to unlink joints and create again. If it does not work, try to segment and create it again. But do not unlink the joints linked with world frame.

4. Q: Why does one of the cameras become video camera?

   A: Try to reboot the system and PC. If it is not working, contact the support team of Vicon.

5. Q: What kind of materials you refers to?

   A: Basically, there are two. The one is a basic tutorial with detailed steps, here. The another is Vicon Nexus2.5 documentation from the official website, here.

References

[1] Arne Böckmann and Tim Laue. Kick motions for the nao robot using dynamic movement primitives. *arXiv preprint arXiv:1606.00600*, 2016.

[2] S. Calinon and A. Billard. Stochastic gesture production and recognition model for a humanoid robot. 3:2769–2774 vol.3, 2004.

[3] S. Calinon, F. D'Halluin, EL Sauser, DG Caldwell, and AG Billard. Learning and reproduction of gestures by imitation an approach based on hidden markov model and gaussian mixture regression. *IEEE ROBOTICS & AUTOMATION MAGAZINE*, 17(2):44–54, 2010.

[4] S. Calinon, F. Guenter, and A. Billard. On learning, representing and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 37(2):286–298, 2007.

[5] Sylvain Calinon. *Robot programming by demonstration: a probabilistic approach*. EPFL Press, Lausanne, Switzerland;Boca Raton, FL;, 1st edition, 2009.

[6] Sonia Chernova and Andrea L. Thomaz. *Robot learning from human teachers*, volume 28;28.;. Morgan & Claypool, 2014.

[7] Damien Garcia. Robust smoothing of gridded data in one and higher dimensions with missing values. *Computational statistics & data analysis*, 54(4):1167–1178, 2010.

[8] Ashutosh Saxen Ian Lenz, Honglak Lee. Deep learning for detecting robotic grasps. *International Journal of Robotics Research (IJRR)*.

[9] Auke J. Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328, 2013.

[10] Jonas Koenemann, Felix Burget, and Maren Bennewitz. Real-time imitation of human whole-body motions by humanoids. pages 2806–2812. IEEE, 2014.

[11] Volker Kruger, Dennis Herzog, Sanmohan Baby, Ales Ude, and Danica Kragic. Learning actions from observations. *IEEE Robotics & Automation Magazine*, 17(2):30–43, 2010.

[12] Zhi Li, Peter Moran, Qingyuan Dong, Ryan J. Shaw, and Kris Hauser. Development of a tele-nursing mobile manipulator for remote care-giving in quarantine areas. pages 3581–3586. IEEE, 2017.

[13] Richard Macliny, Jude Shavlikz, Lisa Torreyz, Trevor Walkerz, and Edward Wildz. Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression. 2005.

[14] Franziska Meier and Stefan Schaal. A probabilistic representation for dynamic movement primitives. *arXiv preprint arXiv:1612.05932*, 2016.

[15] Jacob Nielsen, Anders Stengaard Sørensen, Thomas Søndergaard Christensen, Thiusius Rajeeth Savarimuthu, and Tomas Kulvicius. Individualised and adaptive upper limb rehabilitation with industrial robot using dynamic movement primitives. 1:40, 2017.

[16] Dae-Hyung Park, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. pages 91–98, 2008.

[17] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. pages 763–768, 2009.

[18] Guangzhu Peng, Chenguang Yang, Yiming Jiang, Long Cheng, and Peidong Liang. Teleoperation control of baxter robot based on human motion capture. pages 1026–1031. IEEE, 2016.

[19] Stefan Schaal. *Dynamic Movement Primitives -A Framework for Motor Control in Humans and Humanoid Robotics*. Springer Tokyo, Tokyo, 2006.

[20] Stefan Schaal, Jan Peters, Jun Nakanishi, and Auke Ijspeert. Learning movement primitives. 2004.

[21] A. Vakanski, I. Mantegh, A. Irish, and F. Janabi-Sharifi. Trajectory learning for robot programming by demonstration using hidden markov model and dynamic time warping. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(4):1039–1052, 2012.

[22] Aleksandar Vakanski, Farrokh Janabi-Sharifi, Iraj Mantegh, and Andrew Irish. Trajectory learning based on conditional random fields for robot programming by demonstration. pages 401–408, 2010.

[23] Sethu Vijayakumar and Stefan Schaal. Locally weighted projection regression: An o (n) algorithm for incremental real time learning in high dimensional space. 1:288–293, 2000.

[24] Peter Stone W. Bradley Knox. Interactively shaping agents via human reinforcement. 2009.

[25] Jie Yang, Yangsheng Xu, and C. S. Chen. Human action learning via hidden markov model. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 27(1):34–44, 1997.

[26] You Zhou, Martin Do, and Tamim Asfour. Coordinate change dynamic movement primitives—a leader-follower approach. pages 5481–5488, 2016.