2023 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)
June 28-30, 2023. Seattle, Washington, USA

Kinodynamic Motion Planning for Robotic Arms Based on Learned Motion Primitives from Demonstrations

Joshua A. Ashley, Daniel J. Kennedy and Biyun Xie *Member, IEEE*

***Abstract*— Learning from Demonstration (LfD) is a powerful tool for users to encode information about a task for a robot to perform. LfD has been used with some success in specific types of tasks, however very few implementations consider dynamic features in demonstrations while exploring new environments. The goal of this paper is to propose a novel motion planning algorithm that can incorporate the dynamics of a demonstration and avoid obstacles using learned motion primitives. The method uses a combination of hidden semi-Markov models (HSMM) and neural network controllers to classify and encode motion primitives and their sequences. The encoded motion primitives and their transition probabilities are then used to design a discrete sample space to be utilized by a random tree search algorithm. To evaluate this method, a bar-tending task that includes important dynamic motions was recorded. The recorded demonstrations were used in this method to create the discrete sample space and generate a trajectory for the task in a new environment. The algorithm was run 100 times with a randomly selected set of obstacles and found a feasible trajectory with 91% success.**

I. INTRODUCTION

As one of the most intelligent species, human beings have developed brilliant knowledge and skills to perform numerous tasks from work to daily life. The field of robot learning from demonstration (LfD) has the ideal potential of being able to have a robot accomplish a complicated task from only observed human demonstration data [1]. In particular, most LfD solutions involve simply providing kinesthetic recordings for the robotic system and then deriving policies and skills of performing the tasks to teach the robot in a new environment [2]. As a fully implemented design, LfD can significantly reduce the programming efforts for enabling a robot to perform various assigned tasks. It can even be utilized by non-robotics experts to teach a robot to complete a task with little to no requirement of knowledge about robotic task and motion planning [3], [4]. The LfD technique plays a prominent role in broadening the range of the robot end-users and advancing the progress of bringing robots to our daily life.

The central problem of LfD is how to correctly extract and recognize skills of a demonstration to replicate the task in a unique environment. A variety of methods were developed to extract the skills and they can be sorted into two major categories, i.e., kinematic-based skills and dynamic-based skills. As an example of the first category, a method to learn hand waving motions is developed in [5], where inflection points are extracted from the demonstrated motion data and Gaussian mixture clustering is applied to represent those spaces. In [6], task parametrized Gaussian mixture models is introduced to extrapolate the demonstrations of cleaning a table to different task parameters such as movement locations, amplitude or orientations. In [7], a novel multi-output Gaussian process is proposed to encapsulate the variability retrieved from the demonstrations and efficiently modulate trajectories towards new start-, via- or end-points defined by the task.

The LfD technique gets increasingly non-trivial with the introduction of dynamics, where velocity, acceleration, and/or force features in demonstrations might also need to be derived from data. In [8], hidden semi-Markov models (HSMMs) are used as the high-level symbolic planner to the task with optimal controllers guiding the low-level motion primitives in each action. In [9], hidden Markov models (HMMs) were used where each state was represented by a stable linear parameter varying (LPV) system. This method results in an attractive system where the LPVs attract the current configuration to the demonstrated trajectory while retaining similar dynamics throughout the movement. A similar concept uses time-parameterized HMMs where models are generated for each demonstration using changepoint detection to differentiate each state [10]. A skill tree is then formed by merging well-fitting states from each of the demonstration HMMs. The result is a small, but well defined, set of possible skill sequences that can be performed for the robot to complete a task.

To enable a robot to perform a task in a novel environment after learning from demonstration data, avoiding obstacles not included in demonstration is another critical problem that needs to be solved. Most existing methods combined LfD methods and motion planning methods together to solve this problem. In [11], configuration motion features (robot joint angles) and landmark-based motion features (the location of the points attached to the robot) are first extracted form demonstrations. A sampling-based motion planner is then used to compute a motion plan to avoid obstacles and optimize a learnt cost metric. The method proposed in [12] is to create a Gaussian mixture model (GMM) to represent the demonstration and then uses Gaussian mixture regression (GMR) to traverse the GMM over time optimally. To avoid obstacles, the method partitions and resamples constraint-violating components of the motion. In [13], the skills are modeled as a linear time-varying set of stochastic differential

*This work was supported by the National Science Foundation under Grant #2205292 as well as NASA and the NASA Kentucky EPSCoR Program under NASA award number 80NSSC22M0034.

J. A. Ashley, D. J. Kennedy and B. Xie are with the Department of Electrical and Computer Engineering, University of Kentucky, Lexington, KY 40506 USA Joshua.Ashley@uky.edu, Daniel.Kennedy@uky.edu, Biyun.Xie@uky.edu

equations and a factor graph from one learned skill to another is constructed to traverse an optimal sequence to the goal while avoiding obstacles.

As described above, the problem of extracting and preserving dynamic features in LfD is difficult in its own right. Furthermore, very few existing LfD strategies have considered the problems of avoiding unforeseen obstacles and preserving dynamic features at the same time. To solve these limitations, this study aims to improve on the overall feasibility of LfD for collaborative robots by utilizing and integrating multiple concepts. In particular, the main contributions of this study include: (1) A method of combining the HSMM and neural network controllers is developed to describe complex tasks as discrete sequences of learned motion primitives. (2) Based on the learned motion primitives, a sample space is designed and a discrete sampling-based motion planner is introduced to compute a feasible trajectory in environments with novel obstacles.

The rest of this paper is organized as follows. In the next section a kinodynamic motion planning method based on learned motion primitives from demonstrations is presented. An application example robot bartender and the results are presented in Section III. Finally, the conclusions of this work are drawn in Section IV.

II. Method

A. Overview

The human arm can be modelled as a seven degrees of freedom (DOFs) robotic arm, and the joint angles can also be computed based on the recorded joint positions in the demonstration. The demonstration recording consists of logging joint angles and velocities at consistent time intervals. Given that each demonstration follows some consistent process, the problem is then easily described as a hidden Markov process, where there is an observable output in the joint angles and velocities that describe a hidden state machine. A state is multiple time steps long where the transition to a new state is dependent on the behavior of the motion in that time period and the length of the time period. Because of this, each state can be thought of as a type of motion primitive that has a trajectory distribution over a small duration. Therefore, the state types correspond directly with the motion primitive types. The data that the state encapsulates represents the trajectories of that given motion primitive type.

In order to generate a much larger class of motions to use them in novel environments with varying goal points, a learned motion primitive controller is created from deep neural networks for each motion primitive type. This is possible because motion primitive types describe some underlying dynamics problem that can be solved by a controller [14]. These neural network controllers can generate novel motion primitives that reach over a large distribution of endpoints from the same starting point while retaining the dynamics of that motion primitive type.

These generated motion primitives form a discrete space classified by each motion primitive type and variant of motion primitives belonging to that type. Using the knowledge of state transitions and quality of the learned motion primitives from the prior steps, a discrete sampling-based motion planner is developed to construct a valid trajectory from start to goal point that contains all of the necessary steps between the two. The flowchart including the main components of the proposed kinodynamic motion planning method based on learned motion primitives from demonstrations is shown in Fig. 1.

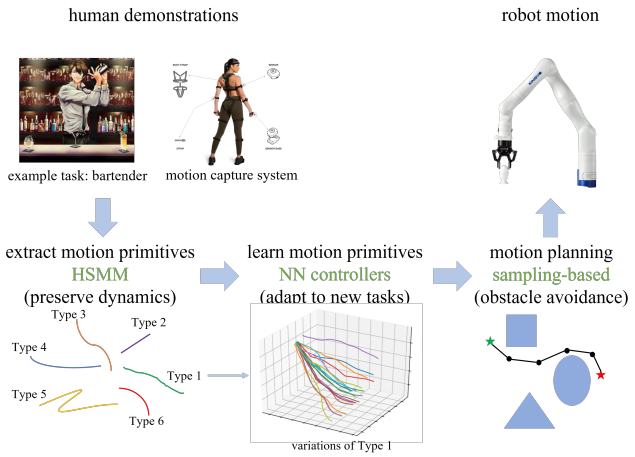


Fig. 1. The flowchart including the main components of the proposed kinodynamic motion planning method based on learned motion primitives from demonstrations is shown.

B. Extracting Motion Primitives from Demonstrations

For this method demonstrations are assumed to be given as joint angles $\boldsymbol{\theta}(t)$ and joint angle velocities $\dot{\boldsymbol{\theta}}(t)$ sampled across time. This temporospatial data serves as the observable input to a HSMM. The HSMM uses Bayesian nonparametric inference to cluster the demonstration data into states [15].

The HSMM is an offline regression process that optimizes a state machine to best fit the observed movement data. The resulting state machine encodes several important features. Let $\mathbb{S}$ be the state space with n_s number of states in the HSMM. The start state probabilities are given by, $\pi(s_i) = P(s_0 = s_i)$ where an arbitrary state $s_i \in \mathbb{S}$. The transition probabilities are represented as a matrix, $\mathbf{T}$, and its elements $\mathbf{T}(i,j) = P(j|i)$ is the transition probability from state i (current state) to state j (next state). The duration distributions $D_i(t_s) = P(j|t = t_s)$ that describes the probability to transition to any new state j given the time t_s in the current state i.

The HSMM segments the observations from demonstrations into states. Fig. 2 visualizes the structure of these state sequences formed from an example demonstration. The underlying data for each state in the sequence belongs to the corresponding motion primitive type. Therefore, from the demonstration data and state sequence, sets of motion primitives categorized by their type can be collected. Each set contains discrete sequences of $\Delta\boldsymbol{\theta}(t_s) = \boldsymbol{\theta}_{t_s} - \boldsymbol{\theta}_0$ that shows the change since the start of the state when $t_s = 0$. The set for each motion primitive type is then well defined

as $M_i = \{\Delta\boldsymbol{\theta}_i(t) : 0 < i \leq n\}$ for n number of occurrences of that motion primitive type.

To expand the possible dataset for new environments, these motion primitive sets can be used to train neural network controllers which will be discussed in the next section. For better fitting results in training the neural network controllers, the motion primitives under each type are normalized with each other with respect to time using dynamic time warping (DTW) [16]. The duration distribution will be applied in the final trajectory generation phase to maintain timing.

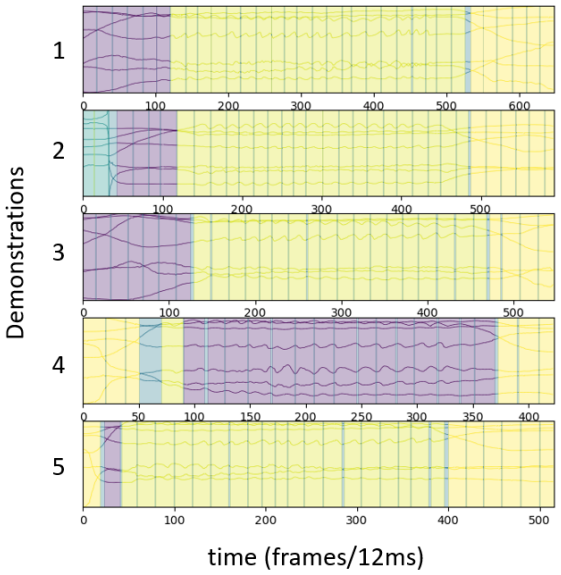


Fig. 2. The result of the HSMM is a classification of each demonstration as a state machine. This figure shows the joint angles and state transition as a function of time throughout the demonstrations. The color corresponds to the current state type and the lines are the current joint angles.

C. Motion Primitive Learning Using Neural Network Controllers

Each motion primitive type is represented by a neural network controller. This neural network controller is generated by a variation of the conditional neural movement primitives (CNMP) framework [14]. The training of these controllers is an offline process and only needs to be done once for a given set of demonstrations. It uses $Y_i \subseteq M_i$ as the training set for the motion primitive controller of type i. The results are checked against the verification set, V_i, for accuracy where $V_i \subseteq M_i$ and $V_i \cup Y_i = M_i$.

The model iterates over the training sets to create a neural network controller that most accurately represents the training and verification sets. Once this controller is found it is used to generate a set of trajectories. For each motion primitive type, a normal distribution $N(\mu_i, \sigma_i)$ is used to represent the observed set of endpoints where μ_i is the mean and σ_i is the standard deviation of the endpoints for each set in M_i. A sampled set of endpoints that is much larger than the observed set is generated. The neural network controller is then applied to fit a trajectory from the start point to each endpoint in the sampled set. These resulting trajectories are learned motion primitives for each type.

Each of these generated trajectories have a corresponding loss value $\mathcal{L}$, that shows the controller's ability to fit that specific endpoint to the desired motion behaviour. This loss value is given by the log probability of the generated trajectory through time, $y(t)_q$, given the expected distribution of the trajectory $\mathcal{N}(\mu_q, \sigma_q)$

$$\mathcal{L}(\rho, \phi) = -log(P(y(t)_q|\mu_q, \text{softplus}(\sigma_q))), \qquad (1)$$

where ρ and ϕ are the current parameters for the Encoder and Query network. μ_q and σ_q are the output of the conditional neural process (CNP) [14]. Therefore the loss accumulates throughout the trajectory as $y(t)_q$ deviates from the mean. In order to emphasize areas of very low variance within a primitives movement, such as reaching the endpoint with accuracy, $\text{softplus}(\sigma_q)$ is used in place of σ_q. This function zeroes out small variances while being proportional to larger variances drastically increasing loss for deviation from the mean trajectory in high variance areas.

D. Designing a Discrete Sampling Space based on Learned Motion Primitives

All the generated motion primitives create a discrete sample space from motion types and variants that represent possible motions that can be taken during a trajectory. Using collected information from the demonstrations processed by the HSMM and neural network motion primitives, a set of probabilities describing the transition relationship between all generated motion primitives can be formed. These sets of motion primitives store their trajectory in joint-space so that information about the manipulator configuration is not lost during planning. This results in an additional benefit of maintaining a differentiable and continuous joint-space trajectory without requiring an inverse kinematics solver to command the manipulator.

If n_m is the number of motion primitive types, the number of states n_s is equal to n_m. Let n_e be the number of endpoints that will be created for each motion type. The matrix $E \in \mathbb{R}^{n_m \times n_e}$ represents the set of all sampled endpoints where the element $E(k, j)$ corresponds to the endpoint of motion primitive type k and variant j. The transition probability matrix $\mathbf{T}$ for the HSMM is the probability of transition from state i to state l, $P(l|i)$, where each state represents a motion primitive. By definition of the conditional probability, each row of the transition matrix is a unit row vector.

Let $\mathcal{L}_{k,j}$ represent the calculated loss of generating the motion with the neural network controller for endpoint j and motion primitive type k. A score is used to evaluate this motion primitive's quality. A higher score corresponds to a better fitting motion primitive. Based on the loss value for a motion primitive, its score can be defined as $S_{k,j} = 1/\mathcal{L}_{k,j}$

and the normalized score as:

$$\hat{S}_{k,j} = \frac{S_{k,j}}{\sum_{i=0}^{n_m} S_{k,j}}. \quad (2)$$

Combining the motion primitive's quality and transition probabilities, the probability of picking motion primitive type k of endpoint variant j given the current state i can be computed as,

$$P(k, j|i) = \hat{S}_{k,j} * \mathbf{T}(i, k). \quad (3)$$

The discrete sampler will pick a node in the tree and its state will be used as the current state i. The probabilities $P(k, j|i)$ for all possible k and j form a probability mass function (PMF) for current state i that the sampler can use to extend from this node.

Particle filtering is used in the sampler to reduce unintended state transitions. It works simply by selecting a lower bound of probability P_{min} and for any combination of k and j, if $P(k, j|i) \leq P_{min}$, then that probability of taking that motion primitive type and variant is set to zero. The resulting PMF is then re-scaled to have the probabilities sum to 1. It is important to note that the value P_{min} should be dependent on number of motion types n_m and variants n_e. This is because the larger n_m and n_e are, the lower the average probability will be.

E. Motion Planning in the Discrete Sampling Space

After the discrete sampling space is formed, a new random tree search algorithm is designed in this subsection to find a feasible trajectory through this discrete space of motion primitives from the start to goal point.

Algorithm 1 shows the main loop sampling, checking, and adding a node to the tree. At the start of each iteration, an existing node in the tree is randomly selected to extend from. Nodes are constrained to have a maximum depth in the tree and maximum number of children. If a node violates these constraints, it is removed from the sampling pool. Once the current node is selected, it is extend from using a motion primitive determined by the discrete sampling space. The discrete sampling space utilizes the probability $P(j, k|i)$ described in the previous subsection. When given the state of the currently select node, the resulting probability is a PMF that can be sampled over. The function "SELECT_CHILD" represents this process. In practice, "SELECT_CHILD" samples each node with removal in order to increase efficiency. The function "IN_COLLISION" checks whether the corresponding sampled node is colliding with an obstacle. If the node is collision-free, it is then added to the tree. The algorithm will search until a reasonable goal configuration q_{goal} is found or a predefined maximum number of iterations N_{max} is reached. All motion planning is performed in a discrete space of motion primitives which encode their trajectories in joint space. When comparing points on the trajectory to obstacles and goal states each point in the motion primitive is converted to task space using forward kinematics.

Compared to conventional sampling-based motion planning, this algorithm is specifically suitable for this problem for multiple reasons. Firstly, depending on the task, it might not be advantageous to proceed towards the goal point and therefore the intermittent goal-point progression is not implemented. Secondly, the extension operation is strictly an extension to one of the endpoints of the motion primitives and not just linearly in a random direction in order to preserve dynamics. Lastly, maximum values for the number of children that a node can have and depth of the tree are pre-determined to reduce the complexity of the sample space.

Algorithm 1 Sampling-based motion planing using learned motion primitives

▷ **Input**: Initial configuration q_0, goal configuration q_{goal}, number of children K, maximum tree depth D, max iterations N_{max}, goal tolerance δ, start state set π
▷ **Output**: Joint trajectory $\boldsymbol{\theta(t)}$

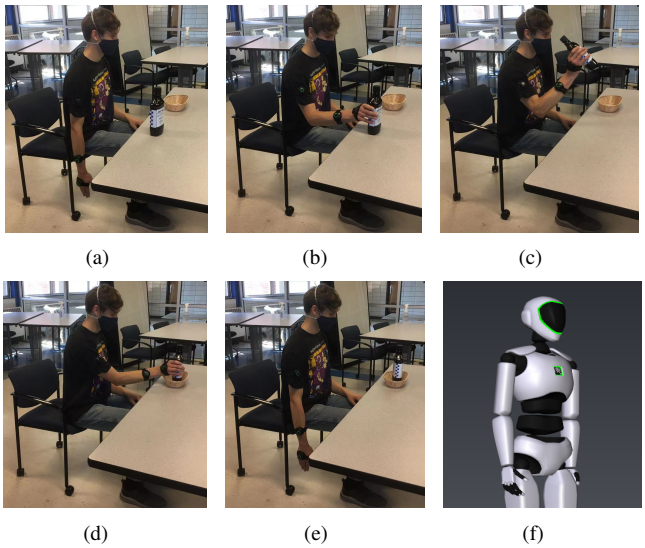
$k = \mathcal{U}(\pi)$
$j = PMF(P(j|k))(k)$
$node = N.init(q_0, k, j)$
$G.init(node)$
$SELECT\_CHILD = PMF(P(k, j|i))$
while $N_{max} > n$ & $currentN.q - q_{goal} > \delta$ **do**
  $parentN = \mathcal{U}(G)$
  if $parentN.children.length < K$ & $parentN.depth < D$ **then**
    $k, j = SELECT\_CHILD(parentN.state)$
    $\Delta q = E(k, j)$
    if $IN\_COLLISION(parentN.q + \Delta q)$ **then**
    else
      $currentN = N.init(parentN.q + \Delta q, k, j)$
      $currentN.depth = parentN.depth + 1$
      $currentN.parent = parentN$
      $parentN.children.append(currentN)$
      $G.append(currentN)$
    end if
  else
    $G.remove(parentN)$
  end if
  $n = n + 1$
end while
$\boldsymbol{\theta(t)}.append(currentN)$
while $\exists\ currentN.parent$ **do**
  $currentN = currentN.parent$
  $\boldsymbol{\theta(t)}.append(currentN)$
end while

III. RESULTS

A. An Application Example: Robot Bartender

In order to test the developed kinodynamic motion planner, a bar-tending task is employed in this work as an application example. The bar-tending task includes a sequence of subtasks, i.e., picking a bottle from a natural posture, shaking the bottle, placing the bottle into a tray, and resetting back

to the initial posture, as shown in Fig. 3. Among these sub-tasks, the shaking sub-task would give intermediate dynamic requirements between the two goals, so the dynamic features extracted from the demonstration need to be well preserved. A volunteer demonstrated this bar-tending task five times, and each demonstration is approximately 30 seconds long. An IMU-based Perception Neuron motion capture system (shown in Fig. 1) is used to record these demonstrations.



(a) (b) (c)

(d) (e) (f)

Fig. 3. The bar-tending actor started from a rest posture in (a), picked up the bottle in (b), shook the bottle in (c), placed the bottle in a tray in (d), and finally reset in (e). The demonstration was recorded using a Perception Neuron motion capture system, and (f) shows the recorded reset posture in the Axis Studio.

The robot arm that will learn from demonstration is a Kinova Gen3 robot arm. It is a 7-DOF anthropomorphic robot arm (shown in Fig. 1) and its kinematic structure is shown in Fig. 4. The human arm can be modeled as a kinematic chain with seven rotational joints, where the first three intersected joints model a spherical shoulder joint, one joint at the elbow and the last three intersected joints model a spherical wrist joint. To map human arm motion to the Kinova robot arm directly, the human arm was modeled to have the same kinematic structure as the Kinova robot arm with scale conversion between the link lengths.

B. Experiments

After the demonstrations are recorded, first the joint angels are computed based on the positions of the elbow joint, wrist joint and key points on the palm by using inverse kinematics. Then, based on the significant goal points of the demonstration, i.e., the bottle position and the tray position, the demonstrations are segmented into three slices: picking, placing, and resetting. The HSMM was given 20 potential states for each segment of the motion to optimize a state machine over the demonstrations. For this experiment 5-12 states were typically utilized for each segment of the demonstration. These extracted motion primitives are used to train the neural network controllers to generate learned motion primitives for 20 new endpoints. Fig. 5 shows four

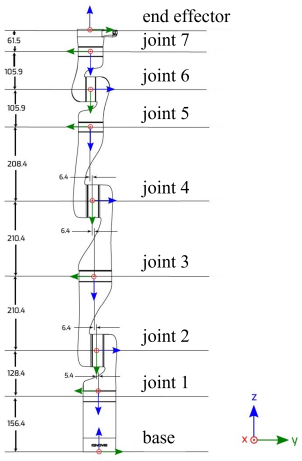


Fig. 4. The kinematic structure of the Kinova Gen3 robot and its dimensions are shown.

examples of different types of learned motion primitives plotted in the workspace by using forward kinematics, where (a), (b), (c), and (d) are the learned motion primitives during the picking, shaking, placing and resetting movement, respectively. For comparison, the same type of extracted motion primitives were plotted in cyan alongside the generated motion primitives.

Figs. 5(a), 5(c), and 5(d) represent some intermediate motion where the manipulator is trying to reach a new point in the workspace, while Fig. 5(b) is trying to carry out a shaking motion with more specific dynamic requirements. It can be seen from each figure that the motions are very consistent with one another. This shows that the HSMM generated well associated clusters of data in the form of motion primitives that the neural network controllers were able to learn a well-fitting trajectory over. For example, in Fig. 5(c) the HSMM found a similar movement from multiple demonstrations that takes an indirect trajectory from the start to end point and the neural network controller was able to follow this trajectory instead of using a simpler and more direct trajectory.

Furthermore, it can be seen from each figure that the generated trajectory preserves dynamic features that are present in the extracted motion primitives. Motions with different dynamic features were chosen to show the algorithms ability to fit controllers to a diverse set of motion primitives. In particular, dynamic constraints to each motion are associated by the HSMM such that Fig. 5(b) represents the shaking motion taken in all of the demonstrations. In this case, the neural network controller was also able to generate trajectories very similar to the extracted shaking motions.

After the motion primitives are learned, the developed discrete sampling motion planning algorithm was tested 100 times on the set of goal points. For each test, two obstacles are randomly selected among a set of obstacles created to interfere with the demonstrated trajectory. Fig. 6 shows four example trajectories from these tests alongside the original demonstrated trajectories. To evaluate the algorithm, the

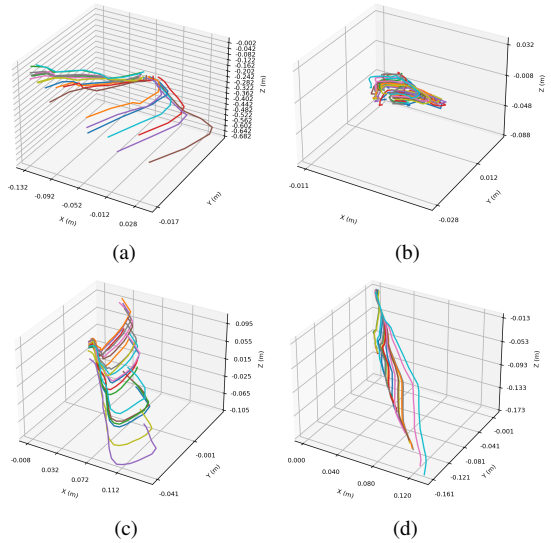


Fig. 5. The learned motion primitives from the neural network controllers for (a) picking, (b) shaking, (c) placing and (d) resetting are shown in the workspace.

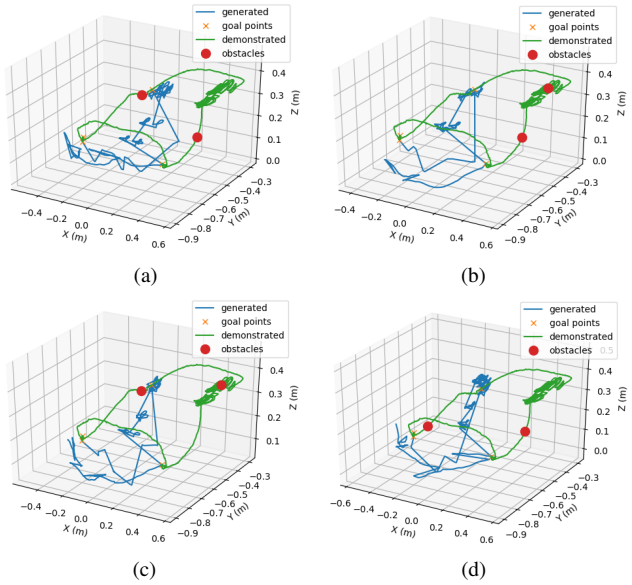


Fig. 6. Four experiments with varying environments are shown. The red circles represent obstacles, and the cross symbols represents the goals points for picking, shaking, placing and resetting. The demonstration trajectory and the generated trajectory are shown in green and blue, respectively.

success of each experiment is defined by reaching the goal point with some tolerance while also avoiding obstacles and being in a valid finishing state defined by the HSMM. The success rate of the algorithm to do so within a set number of iterations is 91 percent over 100 runs. As shown in Fig. 6, the algorithm takes a path in the state machine such that it always accomplishes significant dynamic motions conveyed in the demonstrations, namely the ”shaking” portion of the demonstrations is replicated by the motion planner consistently. Each of these trajectories also follow a similar path to the demonstrations while being able to avoid the new obstacles.

The result of the trajectory was simulated in ROS Gazebo on the Kinova Gen3 7-DOF robotic arm. Fig. 7 shows the simulation results at different times in the trajectory. Visually, the trajectory completes all of the major steps of the task in correct sequence while maintaining natural arm postures which should be derived from the human demonstrator.

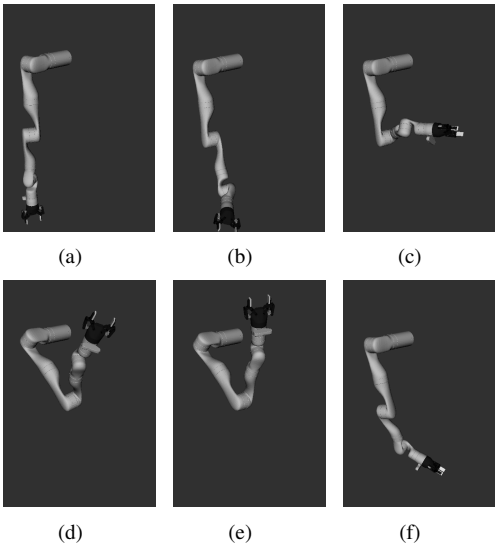


Fig. 7. The simulation for the experiment shown in Fig. 6(b) is conducted in ROS Gazebo. Fig. (a) is the initial posture, (b) occurs during picking, (c), (d), and (e) occur during shaking, and (f) occurs during placing.

IV. CONCLUSIONS

The problem of robot learning from human demonstration with the ability of preserving dynamic features and avoiding unforeseen obstacles in new environments is studied in this paper. HSMMs and neural network controllers are used in combination to generate a discrete sample space of motion primitive types and variants that is able to be explored by a sampling-based motion planner. A bar-tending task conducted by a human demonstrator and applied to a Kinova robot arm was used as a potential application of this proposed algorithm. Results show that the HSMM is able to categorize clusters of demonstration data as states that can then be represented accurately by neural network controllers. Finally, the resulting discrete sampling-based motion planner is able to construct a trajectory for new environments that replicate the sequence of motions within the bar-tending task.

Although the motion planner is able to find a feasible trajectory, observation of the generated trajectories show more work is needed in the areas of enable trajectory optimization and informed sampling of the discrete sampler. A potential optimization algorithm would need to take into account multiple aspects of the trajectory: (1) progress of a given trajectory towards a goal state in the HSMM and goal configuration; (2) a metric evaluating how ”well” the trajectory traversed the state machine and motion primitive space as a whole. These problems will be studied in our future work.

REFERENCES

[1] S. Ekvall and D. Kragic, "Robot learning from demonstration: a task-level planning approach," *International Journal of Advanced Robotic Systems*, vol. 5, no. 3, p. 33, 2008.

[2] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[3] C. Mueller, J. Venicx, and B. Hayes, "Robust robot learning from demonstration and skill repair using conceptual constraints," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6029–6036.

[4] A. Sena and M. Howard, "Quantifying teaching behavior in robot learning from demonstration," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 54–72, 2020.

[5] J.-H. Seo, J.-Y. Yang, and D.-S. Kwon, "Generation of various hand-waving motion of a humanoid robot in a greeting situation," in *2014 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. IEEE, 2014, pp. 374–378.

[6] J. Kim, N. Cauli, P. Vicente, B. Damas, F. Cavallo, and J. Santos-Victor, ""icub, clean the table!" a robot learning from demonstration approach using deep neural networks," in *2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2018, pp. 3–9.

[7] N. Jaquier, D. Ginsbourger, and S. Calinon, "Learning from demonstration with model-based gaussian process," in *Conference on Robot Learning*. PMLR, 2020, pp. 247–257.

[8] G. Canal, E. Pignat, G. Alenyà, S. Calinon, and C. Torras, "Joining high-level symbolic planning with low-level motion primitives in adaptive hri: Application to dressing assistance," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3273–3278.

[9] J. R. Medina and A. Billard, "Learning stable task sequences from demonstration with linear parameter varying systems and hidden markov models," in *Conference on Robot Learning*. PMLR, 2017, pp. 175–184.

[10] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot learning from demonstration by constructing skill trees," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 360–375, 2012.

[11] G. Ye and R. Alterovitz, "Demonstration-guided motion planning," in *Robotics research*. Springer, 2017, pp. 291–307.

[12] X. Li, H. Cheng, and X. Liang, "Adaptive motion planning framework by learning from demonstration," *Industrial Robot: the international journal of robotics research and application*, 2019.

[13] M. Rana, M. Mukadam, S. R. Ahmadzadeh, S. Chernova, and B. Boots, "Towards robust skill generalization: Unifying learning from demonstration and motion planning," in *Intelligent robots and systems*, 2018.

[14] M. Y. Seker, M. Imre, J. H. Piater, and E. Ugur, "Conditional neural movement primitives." in *Robotics: Science and Systems*, vol. 10, 2019.

[15] M. J. Johnson and A. S. Willsky, "Bayesian nonparametric hidden semi-markov models," *Journal of Machine Learning Research*, vol. 14, pp. 673–701, February 2013.

[16] M. Müller, "Dynamic time warping," *Information retrieval for music and motion*, pp. 69–84, 2007.