# Rule-Based Safe Probabilistic Movement Primitive Control via Control Barrier Functions

Mohammadreza Davoodi, Asif Iqbal, Joseph M. Cloud, *Student Member, IEEE*, William J. Beksi, *Member, IEEE*, and Nicholas R. Gans, *Senior Member, IEEE*

***Abstract*— In this paper, we develop a novel and safe control design approach that takes demonstrations provided by a human teacher to enable a robot to accomplish complex manipulation scenarios in dynamic environments. First, an overall task is divided into multiple simpler subtasks that are more appropriate for learning and control objectives. Then, by collecting human demonstrations, the subtasks that require robot movement are modeled by probabilistic movement primitives (ProMPs). We also study two strategies for modifying the ProMPs to avoid collisions with environmental obstacles. Finally, we introduce a rule-base control technique by utilizing a finite-state machine along with a unique means of control design for ProMPs. For the ProMP controller, we propose control barrier and Lyapunov functions to guide the system along a trajectory within the distribution defined by a ProMP while guaranteeing that the system state never leaves more than a desired distance from the distribution mean. This allows for better performance on nonlinear systems and offers solid stability and known bounds on the system state. A series of simulations and experimental studies demonstrate the efficacy of our approach and show that it can run in real time.**

***Note to Practitioners*—This paper is motivated by the need to create a teach-by-demonstration framework that captures the strengths of movement primitives and verifiable, safe control. We provide a framework that learns safe control laws from a probability distribution of robot trajectories through the use of advanced nonlinear control that incorporates safety constraints. Typically, such distributions are stochastic, making it difficult to offer any guarantees on safe operation. Our approach ensures that the distribution of allowed robot trajectories is within an envelope of safety and allows for robust operation of a robot. Furthermore, using our framework various probability distributions can be combined to represent complex scenarios in the environment. It will benefit practitioners by making it substantially easier to test and deploy accurate, efficient, and safe robots in complex real-world scenarios. The approach is currently limited to scenarios involving static obstacles, with dynamic obstacle avoidance an avenue of future effort.**

***Index Terms*— Motion control, motion and path planning, learning from demonstration, optimization and optimal control, robot safety.**

Manuscript received 28 June 2022; revised 19 August 2022; accepted 18 October 2022. This article was recommended for publication by Associate Editor X. Xie and Editor X. Xie upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation (NSF) under Grant 1728057 and in part by The University of Texas at Arlington Research Institute. The work of Joseph M. Cloud was supported by the NSF Graduate Research Fellowships Program under Grant 1746052. *(Corresponding author: Nicholas R. Gans.)*

Mohammadreza Davoodi was with The University of Texas at Arlington Research Institute, Fort Worth, TX 76118 USA. He is now with the Department of Electrical and Computer Engineering, The University of Memphis, Memphis, TN 38152 USA (e-mail: mdavoodi@memphis.edu).

Asif Iqbal and Nicholas R. Gans are with The University of Texas at Arlington Research Institute, Fort Worth, TX 76118 USA (e-mail: asif.iqbal@uta.edu; nick.gans@uta.edu).

Joseph M. Cloud and William J. Beksi are with the Department of Computer Science and Engineering, The University of Texas at Arlington, Arlington, TX 76019 USA (e-mail: joe.cloud@mavs.uta.edu; william.beksi@uta.edu).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TASE.2022.3217468.

Digital Object Identifier 10.1109/TASE.2022.3217468

## I. Introduction

ROBOTS are adept at structured repetitive tasks. However, modern robotic tasks are moving towards reduced structure where items to be handled are in various locations and obstacles may enter the workspace. Completing such tasks in dynamic environments with fixed, preprogrammed movements is not feasible. Moreover, increased proximity between robots and humans working in shared spaces inevitably raises the issue of safety. In fact, safety is a major limiting factor for the development of autonomous robotic partners [1]. Consequently, robot systems need to be capable of detecting task variations, and their motion planning and control algorithms must be flexible enough to allow for variation while guaranteeing safety [2]. In this work, we address the following problem: how to synthesize a controller by learning from and imitating a human teacher to simultaneously enable a robotic arm to accomplish manipulation tasks while ensuring safety constraints.

### A. Motion Planning via Learning From Demonstration

Robot motion planning research has provided a variety of successful frameworks to generate trajectories [3]. Yet, these frameworks can be difficult to implement in environments without a predefined map. In addition, motion planning requires the selection of algorithms, the design of cost functions, and other operations that are outside the expertise of nontechnical users. Learning from demonstration is a paradigm that has played a significant role in addressing the issue of scalability in robot learning [4], [5]. By relying on the presence of a human teacher, it can avoid the drawbacks inherent in traditional robot motion planning. One approach to learning via demonstration is the use of parameterized movements, known as movement primitives (MPs), to encode and generalize human demonstrations for training robots. Probabilistic movement primitives (ProMPs) provide a *distribution* of trajectories learned from multiple demonstrations.

In [6], the design of a stochastic ProMP feedback controller was studied by exploiting the property of the covariance derivatives, which can be explicitly computed. A model-free ProMP controller that adapts movement to force-torque input was designed in [7]. In [8], a model predictive control-based ProMP controller was created for a linear discrete-time system model. Nonetheless, ProMP techniques have notable deficiencies. In the aforementioned works, a linearized model of the system is used for designing the controller. This makes the controller less applicable for nonlinear systems (e.g., robotics, autonomous vehicles, etc.). ProMPs and their resulting controllers are nontrivial to implement, sensitive to noise, and highly dependent upon design parameters and initial conditions. This significantly reduces their usability for non-experts. In addition, since ProMPs are by definition stochastic, the distribution of trajectories may have large support and result in trajectories that deviate from the training set.

On the other hand, it is difficult or even impossible to model a robot's complex manipulation scenarios in unstructured and dynamic environments with a single, fixed MP/ProMP. ProMPs can be combined and blended in various ways. They can be smoothly switched between and modified to force the mean trajectories through specific via-points. This provides a wide variety of trajectory distributions to handle different situations [9]. Investigators have leveraged this capability in numerous ways. A novel method for segmenting demonstrations, recognizing repeated skills, and generalizing complex tasks from unstructured demonstrations was proposed in [10] and [11]. The authors of [12] and [13] developed a new method for learning to sequence single movements from kinesthetic demonstrations in order to reproduce a complex task. A graph structure, called a sequence graph, was utilized for representing sequences of robot movements. Learning the transition behavior was formulated as a classification problem. In [14], a finite-state machine (FSM) and DMPs were combined for the purpose of task representation, and a Bayesian nonparametric hidden Markov model was used for robot movement recognition. In this work, we incorporate a FSM and ProMPs to be able to represent the whole complex task in the environment. Furthermore, we integrate concepts from optimal and safe control theory to execute the learned trajectories.

### *B. Safe Robot Control Approaches*

Real-time safety in safety-critical dynamical systems is an important problem that has received considerable attention [15]. This issue has been investigated by designing polynomial barrier certificates/functions, using offline iterative algorithms based on sum-of-squares optimization, to verify safety for a given dynamical system [16]. For example, the notion of barrier certificates/functions was extended for synthesizing real-time safe control laws for dynamical systems via control barrier functions (CBFs) in [17].

CBFs seamlessly integrate with control Lyapunov functions (CLFs) to offer stability while respecting limits and safe regions of the state space [18], [19], [20]. In addition, CBF and CLF controllers typically solve a constrained quadratic program (QP) to find an optimal controller at runtime. This allows the system to minimize the control effort while ensuring stability and safety. Other tasks formulated as cost functions or constraints can be included as well. A downside to CBFs and CLFs is the complexity in defining barriers and trajectories. Recent efforts to automate the definition of CBFs and CLFs include mapping temporal logic statements with respect to performance requirements [21] and fitting piecewise-linear barrier functions to trained obstacles or regions [22].

The primary goal of CLF/CBF controllers is to enable robots to satisfy certain joint space, workspace, and velocity/force constraints, or to guarantee obstacle avoidance. CBFs were utilized in [23] to design a constraint-enforcing controller for a seven degrees of freedom (DoF) anthropomorphic manipulator. In [19], the concept of robust CBFs was proposed to ensure constraint satisfaction for mechanical systems in the presence of sampling effects and model uncertainties. Moreover, the application of the proposed method to the problem of robotic grasping was studied to ensure an object remains inside the grasp while manipulating it to the reference trajectory. CBFs were also used in [24] to build safety barriers around each robot link and guarantee that the body of the robot can avoid collisions with obstacles. A new method using CBFs was proposed in [25] to achieve smooth transitions between sequential reachability tasks for a continuous-time mobile robotic system.

In contrast to the aforementioned works, we introduce a novel means of automating the design of CLFs and CBFs from ProMPs. We aim to design a CLF/CBF-based framework that guarantees a robot manipulator stays in a safe distribution obtained from the ProMPs. Specifically, the ProMP mean is used to define a CLF, and barriers for the CBFs are defined using the standard deviation of the distribution.

### *C. Contributions*

The overarching goal of this work is to develop a novel, safe control architecture by learning to imitate a human expert for solving complex robot manipulation tasks. To do this, we first divide a whole task into multiple simpler subtasks. Tasks that require movement are modeled by ProMPs. Next, we combine a FSM with ProMPs, whereby the current and probable future states dictate a sequence of ProMPs. Finally, we introduce a unique control method that uses the distribution delivered by each ProMP to define a set of CLFs and CBFs.

Our controller not only enjoys the stability and robustness guarantees possessed by CLF and CBF controllers, but it can also guarantee that the system never leaves a neighborhood defined by the training set. Furthermore, we demonstrate the effectiveness and computational efficiency of our approach through simulations and experiments with a two-link and a six-link robot. Examples of generated control trajectories for a Universal Robots UR5 are shown in Fig. 1.

A preliminary version of this work was presented in [26] where we first described how to design a CBF/CLF controller based on a ProMP trained in the joint space of a robot. Simulations of the approach with multiple robot systems were

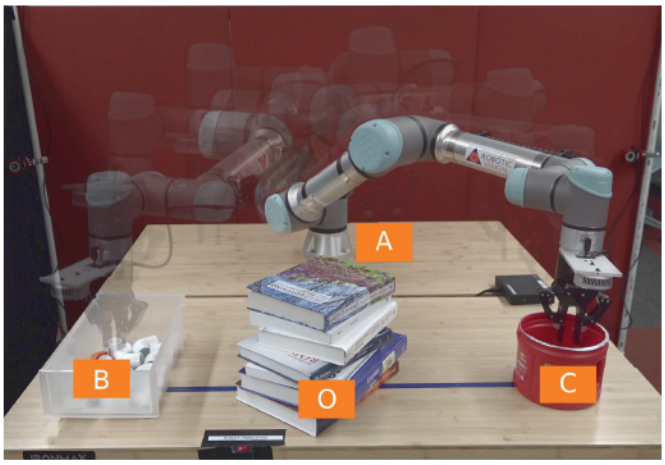



Fig. 1. Robotic setup used in the experimental evaluation. Our controller guarantees that the system never leaves a neighborhood defined by the training set and allows for a straightforward way to define trajectories that enforce safety constraints. "A" denotes the starting point, "B" is the pick location, "C" indicates the place location, and "O" are the obstacles.

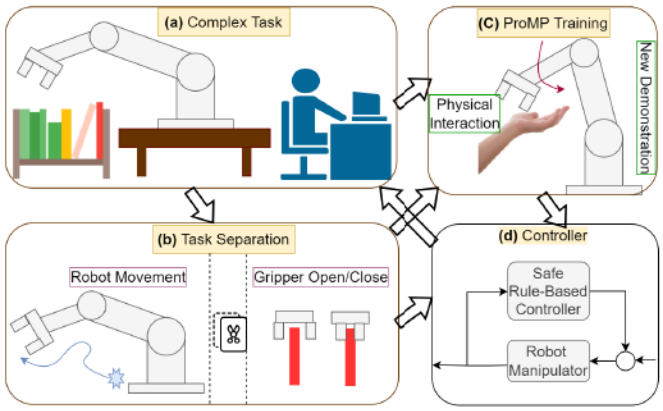



Fig. 2. Overall structure of the proposed system where (a) shows a pick-and-place task, (b) is the process of breaking the complete task into a number of simpler subtasks, (c) demonstrates the ProMP training, and (d) is the control architecture for accomplishing the complete task as well as assuring safety.

presented for verification and demonstration. In this paper, we make the following extensions.

- We add a high-level FSM to govern the robot across a task by switching between ProMP-defined controllers.
- Using our method, a CBF/CLF controller can be designed based on ProMPs trained in the joint space or workspace.
- We develop multiple approaches to obstacle avoidance such as adding via-points or modifying the ProMP.
- We include extensive simulations and experimental validation through a pick-and-place task.

The remainder of this paper is structured as follows. In Section III, ProMPs, CBFs, and CLFs are reviewed. The problem is described in Section II. Our approach for a ProMP controller based on CLFs and CBFs is detailed in Section IV. In Section V, simulation results are presented. The experimental evaluation is discussed in Section VI. Section VII provides a discussion on possible future work. Finally, we conclude the paper in Section VIII.

## II. Problem Formulation

*Notation:* Given a matrix $A$, let $A^\top$ denote its transpose. We indicate the identity and zero matrices, with appropriate dimensions, by $I$ and 0, respectively. Let $\star$ be the symmetric entries of a matrix. For a vector field $f_i(x)$ and vector of vector fields $F(x) = [f_1(x), \ldots, f_n(x)]$, let $L_{f_i}$ and $L_F$ denote, respectively, the Lie derivative along $f_i(x)$ and the vector of Lie derivatives in the directions $f_i(x) : L_F = [L_{f_1}, \ldots, L_{f_n}]$. A zero-mean i.i.d. Gaussian distribution with mean $m$ and covariance $\Sigma$ is denoted $\mathcal{N}(m, \Sigma)$.

### A. System Modeling

Let $p \in \mathbb{R}^{n_p}$ be the Cartesian coordinates of the robot's end-effector. By considering a serial $n_q$ DoF manipulator, $q \in Q$ is a generalized coordinate (joint position) where $Q \subset \mathbb{R}^{n_q}$ denotes the configuration space of the robot. We consider the equations of motion for a robot given in the general form by the Euler-Lagrange equations

$$D(q)\ddot{q} + H(q, \dot{q}) = \tau, \tag{1}$$

where $D(q)$ is the inertia matrix, $H(q, \dot{q}) = C(q, \dot{q})\dot{q} + K(q)$ is a vector containing the Coriolis and gravity terms, and $\tau$ is a vector of applied torques. We consider the forward kinematics map $f_p : \mathbb{R}^{n_q} \to \mathbb{R}^{n_p}$ that determines task coordinates $p$ as a function of the generalized coordinates $q$, i.e.,

$$p = f_p(q). \tag{2}$$

(1) and (2) both encompass the dynamic and kinematic models of the robotic manipulators.

### B. Problem Overview

We first segment a task into simpler subtasks, define the subtasks via ProMPs, and then switch between them using a FSM to achieve a motion primitive representation of the whole task. Consider the pick-and-place task illustrated in Fig. 2(a). *First*, the complete task is broken into a multiple simpler subtasks, as illustrated in Fig. 2(b). The subtasks are categorized into two groups: (i) tasks that require robot joint movement (e.g., moving to the place position from the pick position); (ii) tasks that do not require robot movement (e.g., closing or opening the gripper). *Second*, for each subtask, we collect human demonstrations and then train a ProMP distribution that captures the mean behavior and the teacher's variability over time (Fig. 2(c)). *Third*, we devise a new control approach for accomplishing the task while guaranteeing the safety of the robot (Fig. 2 (d). This approach incorporates a FSM, trained ProMPs, a CLF, and CBFs (Fig. 3).

The FSM considers each subtask as a different state and switches between them automatically according to the commands of the human user and information from environmental sensors. The states of the FSM, in Fig. 3, are categorized into two groups: (i) states that implement open-loop control (shaded blue); (ii) states that implement our CLF/CBF-based ProMP controller (shaded orange). The CLF/CBF-based ProMP controller is formulated such that the system

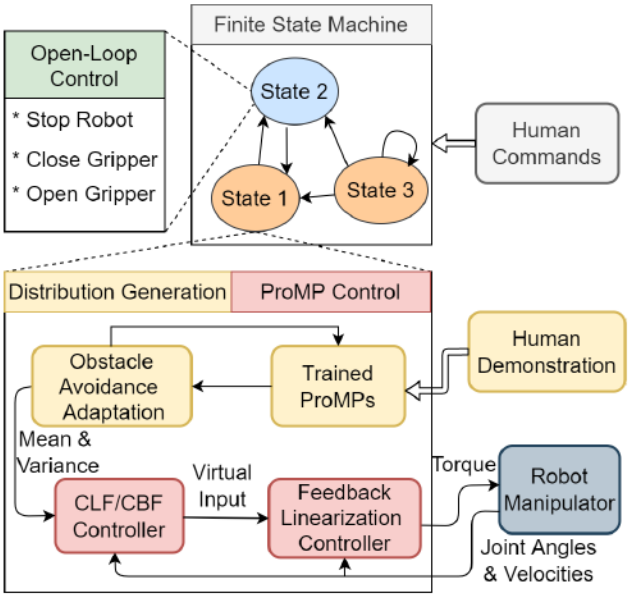



Fig. 3. Block diagram illustrating the general structure of the proposed controller.

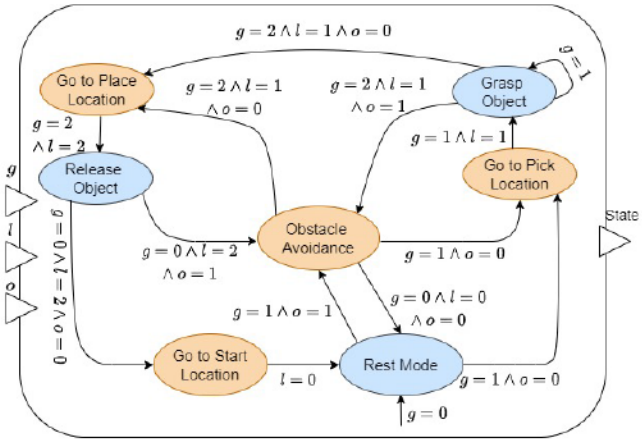



Fig. 4. FSM for a pick-and-place scenario where $\wedge$ denotes the "and" operator. Orange states correspond to the CLF/CBF-based ProMP controllers and blue states are associated with the open-loop controllers.

output tracks a trajectory within the distribution generated by a *joint space* or *workspace* ProMP. To this end, we first construct a nonlinear inner-loop control law founded on the feedback linearization of (1), or (1) and (2). Then, an outer-loop controller based on a CLF/CBF is designed using the distribution parameters of ProMP. The general structure of our system is shown in Fig. 3.

## III. High-Level Control and Trajectory Generation

In this section, we introduce the FSM, ProMPs, as well as ProMP-based obstacle avoidance.

### A. Finite-State Machines

A FSM represents a finite number of states along with the transitions between those states. A system can be in exactly one state at a given time, and the transitions are governed by the current state, system measurements, inputs, etc. [27]. For example, a FSM representing a pick-and-place task for a robot manipulator is shown in Fig. 4. From a starting position, the robot first moves towards the pick position to grab an object. Next, the robot grabs the object and moves towards the place position. The robot places the object and finally moves back towards the starting position. It should be noted that during the process, the robot must avoid collisions with static/dynamic obstacles in the environment.

Let $(g, l, o)$ be the input signals that allow the FSM to switch between different states, defined as follows.

*1) Goal Information ($g$):* The human's expected action for the robot is denoted by this signal. If there is no new pick-and-place goal or when the robot completes its assigned pick-and-place task, then $g = 0$. When $g = 1$, a new pick-and-place goal will be set for the robot and it will move to the pick location to grasp an object. If $g = 2$, then the act of object grasping has been completed and thus the robot should move to the place position.

*2) Positional Information ($l$):* This signal indicates the most recent location the robot has reached, i.e., the *start*, *pick*, or *place* location. When the robot reaches the starting position, $l = 0$. If the robot reaches a pick or place location, then $l$ will change to 1 or 2, respectively.

*3) Obstacle Information ($o$):* The presence of an obstacle near the robot is indicated by this signal. Specifically, $o = 0$ means there is no obstacle, while $o = 1$ indicates an obstacle has been detected by the robot's camera.

Based on these input signals, the FSM will switch between different states in order to complete the assigned task. The following states represent the output of the FSM.

*4) Go to Start Location:* The robot moves to its starting position.

*5) Obstacle Avoidance:* Given the type of obstacle (e.g., static or dynamic), the robot will choose between the following two strategies: "Stop" or "Avoid Obstacle". In case of dynamic obstacles, the robot will stay in its current position until the path is cleared to avoid a collision. For static obstacles, the robot will find a new ProMP to avoid the obstacle through an approach discussed in III-B2.

*6) Go to Pick Location:* The robot moves toward the picking location to grab the assigned object.

*7) Go to Place Location:* The robot is expected to go to its final position to deliver the object.

*8) Grasp Object:* The robot's gripper picks the object.

*9) Release Object:* The robot's gripper places the object.

*10) Rest Mode:* The robot stays in its starting position and is ready to receive new commands and execute the assigned tasks.

### B. Distribution Generation

*1) Probabilistic Movement Primitives:* ProMPs provide a parametric representation of trajectories that can be executed in multiple ways through the use of a probability distribution. In addition, basis functions are used to reduce the model parameters and aid learning over the demonstrated trajectories.

The trajectory distribution can be defined and generated in any space that accommodates the system (e.g., joint space or workspace). In this work, we consider joint space trajectories.

Within a ProMP, the execution of a trajectory is modeled as a set of robot positions, $\zeta_i = \{q_i(k)\}$ (or $\zeta_i = \{p_i(k)\}$), where $k = 0, \ldots, K$, $q_i(k) \in \mathbb{R}$ (or $p_i(k) \in \mathbb{R}$) is the state variable sampled at time $k$, and $i \in \{1, \ldots, n_q\}$ (or $i \in \{1, \ldots, n_p\}$) is the joint (or Cartesian) index of a robot. Let $\Phi(k) = \left[\phi(k)\ \dot{\phi}(k)\right]^\top \in \mathbb{R}^{2\times L}$ be a time-dependent basis function matrix where $L$ is the number of basis functions and let $w_i \in \mathbb{R}^{1\times L}$ be a matrix of weighting terms. A linear basis function model is then given by

$$x_i(k) = \begin{bmatrix} q_i(k) \\ \dot{q}_i(k) \end{bmatrix} = \Phi(k) w_i + \xi_{x_i}, \tag{3}$$

or

$$x_i(k) = \begin{bmatrix} p_i(k) \\ \dot{p}_i(k) \end{bmatrix} = \Phi(k) w_i + \xi_{x_i}. \tag{4}$$

In (3) and (4), a common variable $x_i$ is used to represent the system (robot) states in either joint space $(q_i, \dot{q}_i)$ or workspace $(p_i, \dot{p}_i)$. This is due to the fact that our approach works in either the joint space or workspace with no modifications. Gaussian noise is denoted by $\xi_{x_i} \sim \mathcal{N}(0, \Sigma_{x_i})$. Thus, the ProMP trajectory is represented by a Gaussian distribution over the weight vector $w_i$ and the parameter vector $\theta_i = \{\mu_{w_i}, \Sigma_{w_i}\}$, which simplifies the estimation of the parameters. We marginalize out $w_i$ to create the trajectory distribution, i.e.,

$$\mathcal{P}(\zeta_i, \theta_i) = \int \mathcal{P}(\zeta_i \mid w_i)\mathcal{P}(w_i; \theta_i) dw_i. \tag{5}$$

In (5), the distribution $\mathcal{P}(\zeta_i, \theta_i)$ defines a hierarchical Bayesian model over the trajectories $\zeta_i$ [6] and $\mathcal{P}(w_i \mid \theta_i) = \mathcal{N}(w_i \mid \mu_{w_i}, \Sigma_{w_i})$. In an MP representation, the parameters of a single primitive must be easy to obtain from demonstrations. The distribution of the state $\mathcal{P}(x_i(k)\,;\,\theta_i)$ is

$$\mathcal{P}(x_i(k); \theta_i) = \mathcal{N}(x_i(k) \mid \Phi(k)\mu_{w_i}, \Phi(k)\Sigma_{w_i}\Phi(k)^\top + \Sigma_{x_i}). \tag{6}$$

The mean trajectory $\tilde{\mu}_i(k) \in \mathbb{R}^2$ and trajectory covariance matrix $\Sigma_i(k) \in \mathbb{R}^{2\times 2}$ can be generated from the ProMP distribution using $w_i$, $\Phi(k)$, and (6) as

$$\tilde{\mu}_i(k) = \Phi(k)\mu_{w_i}, \tag{7}$$

$$\Sigma_i(k) = \Phi(k)\Sigma_{w_i}\Phi(k)^\top + \Sigma_{x_i}. \tag{8}$$

Multiple demonstrations are required to learn a distribution over $w_i$. We use a combination of radial and polynomial basis functions for training the ProMP. The basis function is chosen based on the type of robot movement, which can be either rhythmic or stroke-based. From the demonstrations, the parameters $\theta_i$ may be estimated using maximum likelihood estimation [28]. However, this can result in unstable estimates of the ProMP parameters when there are insufficient demonstrations. Our method utilizes regularization to estimate the ProMP distribution similar to Gomez-Gonzalez et al. [29]. We maximize $\theta_i$ for the posterior distribution over the ProMP using expectation maximization,

$$\mathcal{P}(\theta_i \mid x_i(k)) \propto \mathcal{P}(\theta_i)\mathcal{P}(x_i(k) \mid \theta_i). \tag{9}$$

In addition, we use a normal-inverse-Wishart distribution as a prior distribution $\mathcal{P}(\theta_i)$ to increase stability when training the ProMP parameters [29].

*2) ProMP-Based Obstacle Avoidance:* Since all trajectories in the training set are by definition obstacle free, we would expect that $\tilde{\mu}_i(k)$ will also be obstacle free. However, there could be an unusual training set where the operator trains a robot to move multiple ways around an obstacle or an obstacle could be added to a previously safe workspace and require a new $\tilde{\mu}_i(k)$. In such cases, obstacle avoidance can be accomplished by adding via-points or modifying $\tilde{\mu}_i(k)$ through an optimization process.

The via-points modification causes the trajectory to move through a desired position $\hat{q}_i(k)$ (or $\hat{p}_i(k)$) and covariance $\hat{\Sigma}_i$ at a specific time $k$ in the ProMP trajectory. A Gaussian trajectory distribution $p(\zeta_i|w_i)$ can be modified as

$$\hat{\mu}_{w_i} = \mu_{w_i} + L(\hat{q}_i(k) - \Phi(k)\mu_{w_i}), \tag{10}$$

$$\hat{\Sigma}_{w_i} = \Sigma_{w_i} - L\Phi(k)\Sigma_{w_i}, \tag{11}$$

where

$$L = \Sigma_{w_i}\Phi(k)(\hat{\Sigma}_i + \Phi(k)\Sigma_{w_i}\Phi(k)^\top)^{-1},$$

and $\hat{\mu}_{w_i}$ and $\hat{\Sigma}_{w_i}$ is the new mean and covariance for the ProMP distribution. After modulation of the ProMP through via-points, the revised trajectory will stay within the original ProMP distribution.

We also propose another form of trajectory modification using an optimization process. Optimization-based ProMP trajectory modification has been demonstrated by [30] and [31]. However, our method produces a trajectory within the ProMP distribution that can avoid point obstacles with a specified distance. To do this, we find a new weight vector $\hat{w}_i$ for a ProMP represented by $\theta = \{\mu_{w_i}, \Sigma_{w_i}\}$ such the new ProMP trajectory stays within the ProMP distribution while maintaining a safe distance from the obstacle. Let $b$ be the threshold of how far the mean of the new trajectory can deviate from the mean of the old trajectory with respect to the covariance. Let $\mathcal{O}_k$ be the position of the obstacle at time $k$ on the ProMP trajectory. The obstacle can be represented by multiple points. Let $D$ be the distance threshold for the modified trajectory to avoid the obstacle. We propose the following constrained optimization problem,

$$\begin{aligned} \arg\min_{\hat{w}_i} \quad & (\hat{w}_i - \mu_{w_i})^\top(\hat{w}_i - \mu_{w_i}), \\ s.t.\ & (\mathcal{O}_k - \Phi(k)\hat{w}_i)^\top\lambda(\mathcal{O}_k - \Phi(k)\hat{w}_i) \geq D, \\ & d(\Phi(k)\hat{w}_i, \Phi(k)\mu_{w_i}) \leq b, \\ & \Phi(k)(\hat{w}_i - \mu_{w_i}) \geq 0, \end{aligned}$$

where $\lambda$ is a matrix used to prioritize the position and velocity to avoid the obstacle and $d$ is the Mahalanobis distance between the new trajectory and the ProMP mean, i.e.,

$$d(\Phi\hat{w}_i, \Phi\mu_{w_i}) = (\Phi\hat{w}_i - \Phi\mu_{w_i})^\top(\Phi\Sigma_{w_i}\Phi)^{-1}(\Phi\hat{w}_i - \Phi\mu_{w_i}),$$

where the $\Phi(k)$ dependence on $k$ was dropped for clarity.

Both the via-point and optimization methods can be used for obstacle avoidance while ensuring the mean of the modified trajectory is inside the original ProMP distribution.

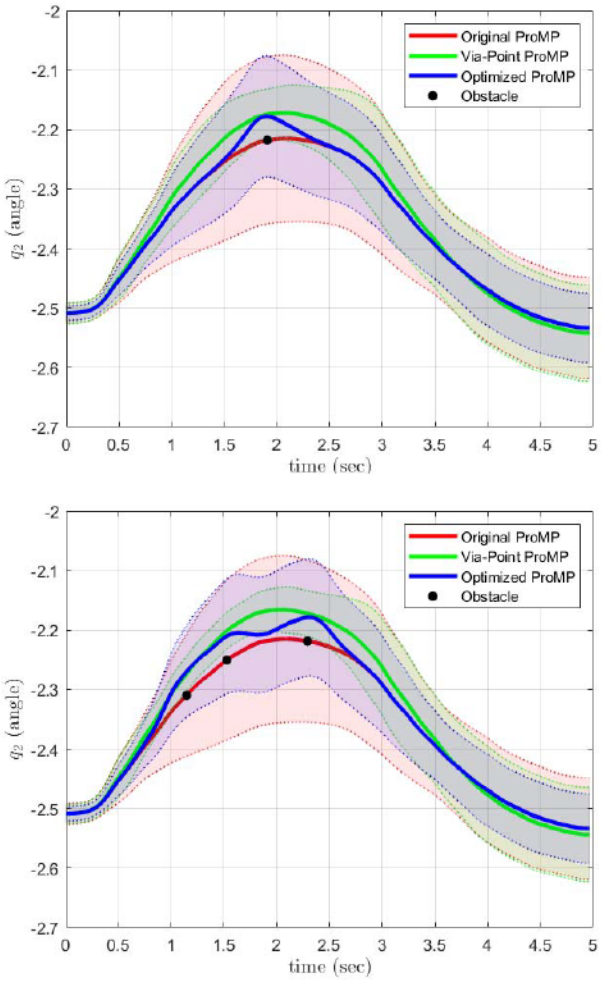



Fig. 5. Comparison between via-point and optimization-based ProMP modifications for obstacle avoidance. The top plot shows the modified trajectories for one obstacle while the bottom plot shows the updated trajectories for three obstacles.

Fig. 5 shows modified ProMPs based on the via-point and optimization process. The via-point method has a smoother transition to avoid a point obstacle while the optimization-based ProMP technique remains closer to the original ProMP when clearing an obstacle.

## IV. Control Construction

In this section, we introduce the core components of our control architecture including the feedback linearization controller, CLF/CBF-based ProMP controller, condition for switching between ProMP controllers, and open-loop controllers. First, we design the inner-loop (feedback-linearization controller) and outer-loop (CLF/CBF controller) such that the following problem objectives are satisfied.

1) Design a feedback linearization controller to obtain a linear and decoupled input-output closed-loop relationship for the joint space or workspace error signal.
2) Create a CLF to stabilize the system such that $\forall i$, $q_i \to \mu_i$ or $p_i \to \mu_i$, where $\mu_i$ is the first element of $\tilde{\mu}_i$.
3) Develop a CBF to guarantee that the error $e_i^q = q_i - \mu_i$ or $e_i^p = p_i - \mu_i$ satisfies the safety constraint $\forall i, |e_i^q| < \sigma_i$ or $|e_i^p| < \sigma_i$, where $\sigma_i$ is the $(1,1)$ element of $\Sigma_i$.

Then, we discuss the required conditions for switching between different CLF/CBF-based ProMP controllers using a FSM. Finally, we elaborate on our strategy for designing open-loop controllers.

### A. Inner-Loop Control Design

To address the first problem objective, we design a feedback linearization controller for obtaining a linear relationship on the joint space error $e_i^q$, $i = 1, \ldots, n_q$, or workspace error $e_i^p$, $i = 1, \ldots, n_p$.

*1) Joint Space Nonlinear Control:* We define the joint space error and trajectory vectors as $e^q = [e_1^q, \ldots, e_{n_q}^q]^\top$ and $\mu = [\mu_1, \ldots, \mu_{n_q}]^\top$, respectively. In this work, we limit ourselves to relative degree-two systems. Using (1), we obtain the error system as

$$\ddot{e}^q(q, \dot{q}) = -D^{-1}(q)H(q, \dot{q}) + D^{-1}\tau - \ddot{\mu}. \tag{12}$$

We can prescribe the following control law $\tau$ to linearize the nonlinear error system (12),

$$\tau = D(q)\big(D^{-1}(q)H(q, \dot{q}) + \ddot{\mu} + \nu\big), \tag{13}$$

where $\nu$ is an auxiliary feedback control value. This yields the second order linear system from input $\nu$ to output $e^q$,

$$\ddot{e}^q = \nu. \tag{14}$$

*2) Workspace Nonlinear Control:* The Jacobian matrix $J(q) = \frac{\partial f_p(q)}{\partial q} \in \mathbb{R}^{n_p \times n_q}$ determines the relation between task space and joint space velocities as well as between task-space forces/torques $f$ and joint torques $\tau$,

$$\dot{p} = J(q)\dot{q}, \tag{15}$$
$$\tau = J(q)^\top f. \tag{16}$$

Considering the robot dynamics (1) and the Jacobian matrix $J$, the relationship between $p$ and $q$ can be obtained as

$$\ddot{p} - \dot{J}\dot{q} + JD(q)^{-1}H(q, \dot{q}) = JD(q)^{-1}\tau. \tag{17}$$

Defining the workspace error vector as $e^p = [e_1^p, \ldots, e_{n_p}^p]^\top$, we have

$$\ddot{e}^p = \dot{J}\dot{q} - JD(q)^{-1}H(q, \dot{q}) + JD(q)^{-1}\tau - \ddot{\mu}. \tag{18}$$

We establish the control law $\tau$ to linearize (18),

$$\tau = H(q, \dot{q}) + D(q)J^\dagger(-\dot{J}\dot{q} + \ddot{\mu} + \nu), \tag{19}$$

where $J^\dagger$ is the generalized inverse of $J$ and $\nu$ is the additional control input. This yields the second-order linear system from input $\nu$ to output $e^p$,

$$\ddot{e}^p = \nu. \tag{20}$$

*3) Decoupled Linearized Models:* For simplicity, we abuse notation by using $e$ in place of $e^p$ or $e^q$. By defining $\eta = [e,\ \dot{e}]^\top$, (14) or (20) can be written as a linear time-invariant system,

$$\dot{\eta} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \eta + \begin{bmatrix} 0 \\ I \end{bmatrix} v. \tag{21}$$

From there, $n_q$ (joint space) or $n_p$ (workspace) decoupled systems can be obtained from (21),

$$\dot{\eta}_i = F\eta_i + Gv_i, \tag{22}$$

where $\eta_i = [e_i,\ \dot{e}_i]^\top$, $F = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$, $G = [0\ 1]^\top$, $i = 1, \ldots, n_q$ in case of joint space, and $i = 1, \ldots, n_p$ in case of workspace.

### B. Outer-Loop Control Design

The second problem objective is accomplished by ensuring $e_i \to 0$. This is done by designing an appropriate CLF. To satisfy the third problem objective, it is sufficient to make $|e_i| < \sigma_i$. This objective is satisfied by defining appropriate CBFs. In the ensuing subsections, the appropriate CLFs and CBFs are defined for the system in (22). Moreover, the $i^{\text{th}}$ controller for each system in (22) is designed by combining the corresponding CLFs and CBFs through a QP problem.

*1) Control Barrier Functions:* Consider the affine, nonlinear system

$$\dot{x} = f(x) + \tilde{G}(x)u, \tag{23}$$

where $x \in \mathbb{R}^n$ denotes the state, $u \in \mathbb{R}^m$ is the control input, $\tilde{G} = [g_1, \ldots, g_m]$, and $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g_i : \mathbb{R}^n \to \mathbb{R}^n$ are locally Lipschitz vector fields. It is assumed that the system in (23) is controllable. Let $C$ be a *safe set* such that if $x(t) \in C\ \forall t$, then the system remains safe. A smooth function $h(x) : \mathbb{R}^n \to \mathbb{R}$ is defined to encode a constraint on $x$ such that

$$\begin{aligned} C &= \{x : h(x) \geq 0\}, \\ \partial C &= \{x : h(x) = 0\}, \\ \text{Int}(C) &= \{x : h(x) > 0\}, \end{aligned} \tag{24}$$

where $\text{Int}(C)$ and $\partial C$ denote the interior and boundary of $C$, respectively.

*Definition 1: [32]* Given $C$ and $h$, a function $B : C \to \mathbb{R}$ is a *reciprocal CBF* if there exists class $\mathcal{K}$ functions $\alpha_1$, $\alpha_2$, and a constant scalar $\gamma > 0$ such that

$$\begin{aligned} &\frac{1}{\alpha_1(h(x))} \leq B(x) \leq \frac{1}{\alpha_2(h(x))}, \\ &L_f B(x) + L_{\tilde{G}} B(x)u - \frac{\gamma}{B(x)} \leq 0. \end{aligned} \tag{25}$$

We propose two safety constraints for each system in (22). More specifically, each system should satisfy $-\sigma_i < e_i < \sigma_i$. Consequently, we have the following two safety constraints

$$\begin{aligned} h_{i1} &= e_i + \sigma_i, \\ h_{i2} &= -e_i + \sigma_i. \end{aligned} \tag{26}$$

From (26), we have multiple time-varying constraints that should be satisfied simultaneously. For each system in (22), it is easy to verify that $L_G e_i = 0$ and $L_G L_F e_i \neq 0$. Thus, the safety constraint has a relative degree of two. For the relative degree-two constraints, reciprocal CBF is defined as [33],

$$B_j(\eta_i) = -\ln\left(\frac{h_{ij}(\eta_i)}{1 + h_{ij}(\eta_i)}\right) + a_{Eij}\frac{b_{Eij}\dot{h}_{ij}(\eta_i)^2}{1 + b_{Eij}\dot{h}_{ij}(\eta_i)^2}, \tag{27}$$

where $j \in \{1, 2\}$ and $a_{Eij}$, $b_{Eij}$ are positive scalars. Note that the smaller the values of $a_{Eij}$ and $b_{Eij}$ are, the farther the system will remain from the constraint surfaces. The following control barrier condition should be satisfied for time-varying constraints, which leads to time-varying CBFs,

$$L_F B_j(\eta_i) + L_G B_j(\eta_i)v_i + \frac{\partial B_j(\eta_i)}{\partial t} - \frac{\gamma_i}{B_j(\eta_i)} \leq 0. \tag{28}$$

*2) Control Lyapunov Function:* A CLF can be used to design the control inputs of a dynamical system (23) to ensure objectives such as stability, convergence to the origin (or other set points), or convergence to a desired trajectory. In order to have a construction similar to CBFs, we will consider exponentially stabilizing CLFs [32].

*Definition 2:* In a domain $X \subset \mathbb{R}^n$, a continuously differentiable function $V : X \to \mathbb{R}$ is an *exponentially stabilizing CLF (ES-CLF)* if $\forall x \in X$ there exists positive scalar constants $c_1$, $c_2$, $c_3 > 0$ such that

$$\begin{aligned} c_1||x||^2 \leq V(x) \leq c_2||x||^2, \\ L_f V(x) + L_{\tilde{G}} V(x)u + c_3 V(x) \leq 0. \end{aligned} \tag{29}$$

For each system in (22), we consider the ES-CLF [34],

$$V_{\epsilon_i}(\eta_i) = \eta_i^\top \begin{bmatrix} 1/\epsilon_i I & 0 \\ 0 & I \end{bmatrix} P \begin{bmatrix} 1/\epsilon_i I & 0 \\ 0 & I \end{bmatrix} \eta_i, \tag{30}$$

where $\epsilon_i$ is a positive scalar and $P$ is a symmetric positive definite matrix that can be obtained by solving the continuous-time algebraic Riccati equation,

$$F^\top P + PF - PGG^\top P + I = 0. \tag{31}$$

In order to exponentially stabilize the system, we want to find $v_i$ such that

$$\dot{V}_{\epsilon_i}(\eta_i) = L_F V_{\epsilon_i}(\eta_i) + L_G V_{\epsilon_i}(\eta_i)v_i \leq -\frac{c_{3i}}{\epsilon_i} V_{\epsilon_i}(\eta_i), \tag{32}$$

where $c_{3i}$ is a positive constant value.

*3) Quadratic Program (QP):* For each linearized, decoupled system, a control input should be obtained that guarantees adherence to both safety constraints given in (26). To compute a single safe control, the CBF and CLF conditions are combined through a QP by solving a constrained optimization problem at each point in time [23]. To guarantee a feasible solution for the QP, the CLF constraint in (32) can be relaxed by $\delta_i > 0$ [32] giving

$$L_F V_{\epsilon_i}(\eta_i) + L_G V_{\epsilon_i}(\eta_i)v_i + \frac{c_{3i}}{\epsilon_i} V_{\epsilon_i}(\eta_i) \leq \delta_i. \tag{33}$$

The QP should be designed to minimize the relaxation parameter while finding a feasible solution. Furthermore, the QP should include a tunable weighting factor on $\delta_i$ such that the user can mediate a trade-off between reducing tracking error and control effort minimization in a way that safety is always satisfied. In the following, $n_q$ (or $n_p$) QPs are proposed to unify ES-CLF and CBFs for each system in (22) into a

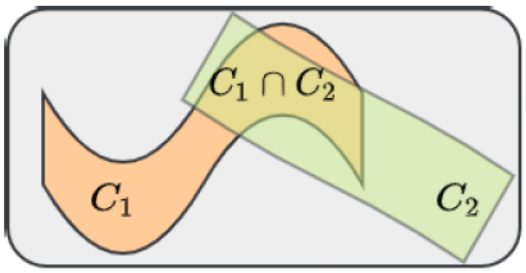



Fig. 6. Switching between the safe sets. The orange and green regions show the safe sets while the gray region represents the unsafe set. The system's state must be in the intersection of the safe sets when switching between them.

single controller. The $n_q$ (or $n_p$) QPs for $i \in \{1, \ldots, n_q\}$ (or $i \in \{1, \ldots, n_p\}$), and $j \in \{1, 2\}$ are defined as

$$\begin{aligned} \min_{\mathbf{v}_i=[v_i,\,\delta_i]^\top \in \mathbb{R}^2} \quad & \mathbf{v}_i^\top H_i \mathbf{v}_i, \\ \text{subject to } & L_F V_{\epsilon_i}(\eta_i) + L_G V_{\epsilon_i}(\eta_i) v_i \\ & + \frac{c_{3i}}{\epsilon_i} V_{\epsilon_i}(\eta_i) \leq \delta_i, \quad \text{(CLF)} \\ & L_F B_j(\eta_i) + L_G B_j(\eta_i) v_i \\ & + \frac{\partial B_j(\eta_i)}{\partial t} \leq \frac{\gamma_i}{B_j(\eta_i)}, \quad \text{(CBFs)} \end{aligned} \tag{34}$$

where $H_i = \begin{bmatrix} 1 & 0 \\ 0 & p_{sci} \end{bmatrix}$ and $p_{sci} \in \mathbb{R}_+$ is a variable that can be chosen based on the designer's assessment of weighting the control inputs.

### C. Switching Between CBFs

In our proposed controller, switching between different CBFs occurs when the FSM changes state. The required condition to guarantee system safety when switching between different CBFs is obtained as follows. Let $C_1$ and $C_2$ be the respective safe sets before and after the occurrence of a switch as depicted in Fig. 6. When the first CBF $h_1(x, t)$ is replaced with the second CBF $h_2(x, t)$, the required condition for guaranteeing system safety (i.e., assuring the invariance of $C_2$ after the switch) is that the system state $x$ should be in the intersection of both sets $C_1$ and $C_2$. Formally, a set $C$ is forward invariant if for every $x_0 \in C$, $x(t, x_0) \in C$ for all $t \in I(x_0)$, where $I(x_0)$ is the maximal interval of existence of $x(t, x_0)$. Based on this fact, if the system's state $x$ is in the intersection of $C_1$ and $C_2$, then the new barrier function $h_2(x, t)$ will make the new safe set $C_2$ *invariant* after the switch. As a result, this condition assures the safety of the system and it will be used to design the switches between different CBF constraints in this work.

It is worth noting that there could be situations that prevent the robot from entering the intersection of the two safe sets in the planned time. Such situations might be due to the appearance of new dynamic obstacles or changes in the environment. Generally, in such cases the solution should wait until the intersection is reached.

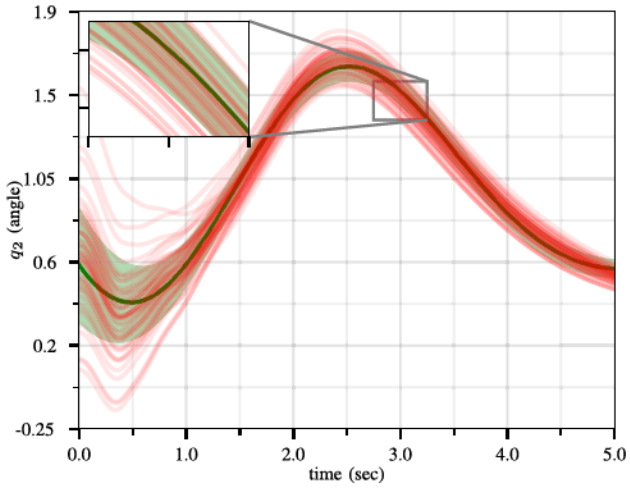



Fig. 7. Training of the ProMPs for the second joint. The 50 input trajectories are shown in red and the ProMP mean joint trajectories ($\mu_2$) are shown in dark green. A light-green fill shows $\mu_2 \pm \sigma_2$.

### D. Open-Loop Control Design

Within the FSM, some states do not require the effort of training a ProMP or the complexity of the CBF/CLF controller. For instance, the CLF/CBF-based ProMP controller can move the robot within close proximity of an item, and a simple open-loop or closed-loop controller (e.g., visual servoing) can handle a grasping task. In this work, we define an open-loop controller to actuate the gripper to pick up an object, release the object, and halt the robot. Each ProMP ends at the desired location to perform an open-loop gripper task. During a halt state, all joints are locked.

## V. Simulations

In this section, we demonstrate different aspects and capabilities of our methodology for the ProMP control of a robotic system. As previously stated, our methodology is capable of controlling robot manipulators based on ProMPs trained and generated either in the joint space or the workspace. Accordingly, the problems of joint space and workspace control of a two-link robot are respectively considered in Sections V-A and V-B. The system models and proposed real-time controller are simulated using MATLAB 2019a. All computations were run on a Dell OptiPlex 7050 machine with an Intel Core i7-7700X CPU and 8 GB of memory.

### A. Case Study 1: Joint Space Control of Two-Link Robot

We consider a rigid, two-link robot with the dynamic model of (1) and the following parameters [35]

$$D(q) = \begin{bmatrix} m_1 l_1^2 + m_2(l_1^2 + l_2^2 + 2 l_1 l_2 \cos(q_2)) & \star \\ m_2(l_2^2 + l_1 l_2 \cos(q_2)) & m_2 l_2^2 \end{bmatrix},$$
$$C(q, \dot{q}) = \begin{bmatrix} -m_2 l_1 l_2 \sin(q_2) \dot{q}_2 (2\dot{q}_1 + \dot{q}_2) \\ m_2 l_1 l_2 \dot{q}_1^2 \sin(q_2) \end{bmatrix},$$
$$K(q) = \begin{bmatrix} (m_1 + m_2) g l_1 \sin(q_1) + m_2 g l_2 \sin(q_1 + q_2) \\ m_2 g l_2 \sin(q_1 + q_2) \end{bmatrix},$$

where $m_1$ and $m_2$ are the link masses, $l_1$ and $l_2$ are the lengths of the links, and $g$ is the gravitational acceleration.

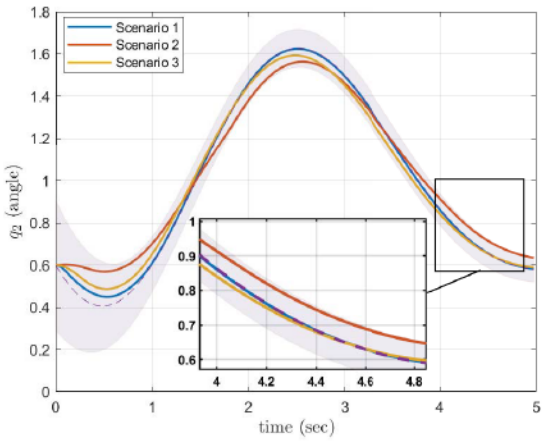



Fig. 8. Results of the CLF/CBF-based ProMP controller for the second joint. The safe region of $\mu_2 \pm \sigma_2$ is shown as a filled "tube". The control results in different trajectories for distinct values of the weight $p_{sc2}$, but all trajectories remain safe.

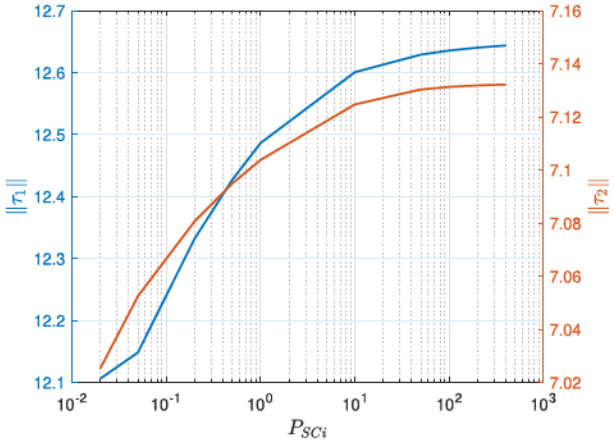



Fig. 9. Control effort with respect to different values of $P_{SCi}$.

For the simulations, the values of these variables are selected as $m_1 = 1$, $m_2 = 1$, $l_1 = 1$, $l_2 = 1$, and $g = 9.8$.

We generated 50 trajectories that achieve a goal position from various starting positions while avoiding obstacles. Using this dataset, we train a ProMP with Alg. 1 from [29]. We use $L = 2$ basis functions consisting of five radial basis parameters. This result originally appeared in [26], and results for both joints can be seen there. We present the results for a single joint here. In Fig. 7 the training trajectories are shown in red, the ProMP mean joint trajectories ($\mu_2$) are shown in dark green, and in a light-green fill is $\mu_2 \pm \sigma_2$.

Three sets of simulations were conducted. In each simulation, the CLF parameters are selected as $\epsilon_i = 0.1$ and $c_{3i} = 0.5$. In the *first scenario*, greater priority is given to the CLF than CBF by choosing a high gain, i.e., $p_{sci} = 200$. Moreover, the CBF design parameters are set to $a_{E11} = a_{E12} = 20.1$, $a_{E21} = a_{E22} = 20$, $b_{E11} = b_{E12} = 1$, $b_{E21} = b_{E22} = 0.9$, $\gamma_1 = 10.1$, and $\gamma_2 = 9$. In the *second scenario*, $p_{sci}$ is chosen as $p_{sc1} = p_{sc2} = 0.02$, which implies less priority to the CLF in comparison with the CBF. Moreover, the CBF parameters in this scenario are similar to the first scenario. To show the effects of changing the CBF parameters $a_{Eij}$, $b_{Eij}$, and $\gamma_i$, we consider another scenario. In the *third scenario*, $a_{E11} = a_{E12} = 1.1$, $a_{E21} = a_{E22} = 1.1$, $b_{E11} = b_{E12} = 0.4$, $b_{E21} = b_{E22} = 0.5$, $\gamma_1 = 1.3$, and $\gamma_2 = 1.51$, with $p_{sci} = 0.02$ as in the second scenario. Consequently, the effects of changing the CBF parameters can be concluded by comparing the second and third scenarios.

The simulation results are exhibited in Fig. 8. In the *first scenario*, by choosing a large value for $p_{sci}$ (more priority to the CLF than CBF), the system output remains close to the mean trajectory. However, in the *second scenario*, by considering a small value for $p_{sci}$ (more priority to the CBF than CLF), the system remains safely inside the distribution but does not necessarily stay close to the mean. In the *third scenario*, it can be seen that by choosing smaller values for $a_{Eij}$, $b_{Eij}$, and $\gamma_i$, the system output can have more deviation from the constraint surfaces, i.e., be closer to the mean trajectory. In short, our method provides a valuable option to the designer that grants differing levels of trajectory flexibility while ensuring safety.

Moreover, our technique allows for prioritizing between strict tracking of the ProMP mean, and "loose" but safe tracking of mean trajectory which is not possible in the native ProMP control design nor with only a CLF controller. This enables us to design safe controllers with different desired performance attributes from the viewpoint of control effort, length of trajectories, etc. For example, the results of simulation for different values of $P_{SCi}$ are shown in Fig. 9. From Fig. 9, it is clear that by decreasing $P_{SCi}$ we can have trajectories that require less control effort.

### B. Case Study 2: Workspace Control of Two-Link Robot

To demonstrate the capabilities of our methodology in safe workspace control design we consider a two-link robot with the same dynamic model represented in Section V-A. Moreover, the forward kinematics equations for the 2-DOF robotic manipulator have been derived as [36]

$$\begin{aligned} p_1 &= l_1 \cos(q_1) + l_2 \cos(q_1 + q_2), \\ p_2 &= l_1 \sin(q_1) + l_2 \sin(q_1 + q_2), \end{aligned} \tag{35}$$

where $p_1$ and $p_2$ are, respectively, the $x$ and $y$ position of the robot's end-effector. The Jacobian matrix $J(q)$ can be used to obtain the relationship between the end-effector velocity, $\dot{p}_i$, and the joint velocities, $\dot{q}_i$, as well as to design the nonlinear feedback linearization controller (19). This matrix is

$$J = \begin{bmatrix} -l_1 \sin(q_1) - l_2 \sin(q_1 + q_2) & -l_2 \sin(q_1 + q_2) \\ l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) & l_2 \cos(q_1 + q_2) \end{bmatrix}. \tag{36}$$

Consider the two regions A and B in the robot workspace as demonstrated with blue ellipsoids in Fig. 10 (top). The goal is to control the robot to move from A to B, and then move back to region A from B. Moreover, we consider that the robot encounters a temporary obstacle, such as a human coworker, when it reaches region B. Hence, it should stop (stay in region B) until the path is cleared to avoid a collision. The task is modeled via a FSM with the following three states: "Go to Region A", "Go to Region B", "Obstacle Avoidance".

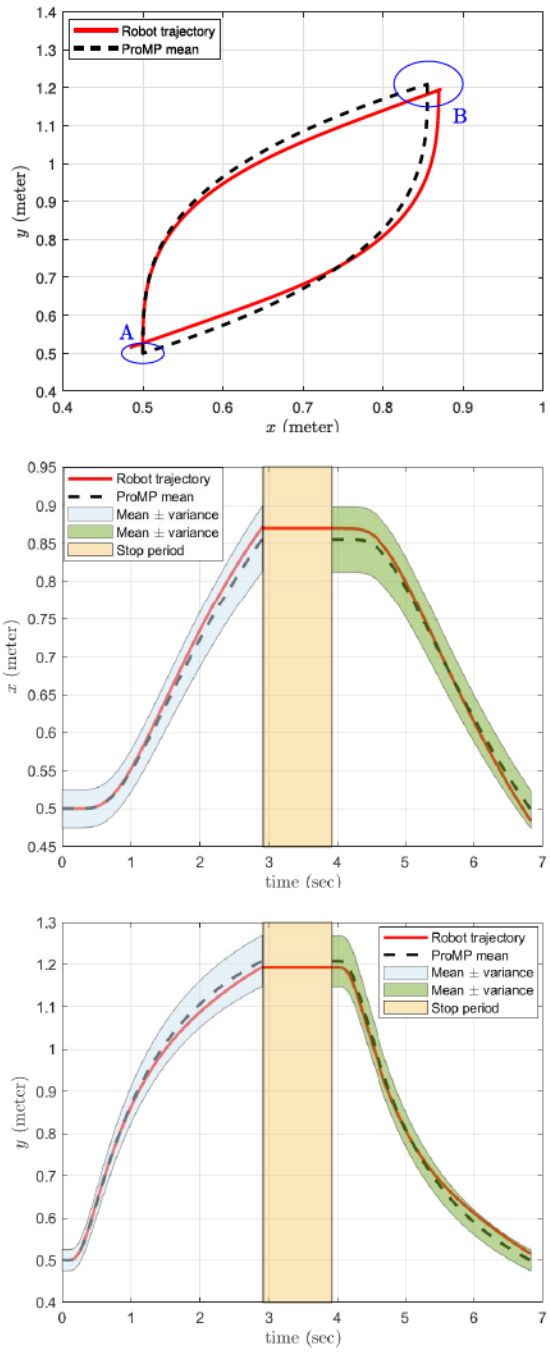



Fig. 10. Results of the workspace ProMP controller for the $x-y$ position (top), the $x$ position (middle), and the $y$ position (bottom).

The robot motion between two regions A and B is modeled via ProMPs. In Fig. 10 (middle) and (bottom), the mean of the ProMP distributions are shown with dashed black lines and the mean $\pm$ variance bounds for the ProMPs *from A to B* and *from B to A*, respectively, are shown as filled tubes with light blue and green.

The CLF parameters are selected as $\epsilon_i = 0.1$ and $c_{3i} = 0.6$. The weighting variable $p_{sci}$ is selected as $p_{sci} = 0.2$. Moreover, the CBF design parameters are set to $a_{E11} = a_{E12} = 1.1$, $a_{E21} = a_{E22} = 1.2$, $b_{E11} = b_{E12} = 0.5$, $b_{E21} = b_{E22} = 0.55$, $\gamma_1 = 1.51$, and $\gamma_2 = 10.51$. The

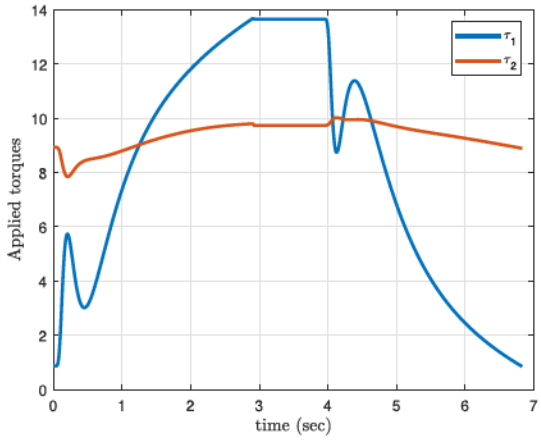



Fig. 11. Robot torque inputs.

robot trajectory generated using the proposed workspace safe controller is depicted by a solid red line in Fig. 10. The shaded orange area in Fig. 10 (middle and bottom) indicates the occurrence of an obstacle. It can be seen that the system states ($x$, $y$ positions of the robot), at the time of switching between two ProMPs, are at the intersection of both ProMPs. This satisfies the required condition of the rule-based safe control design in Section IV. C, i.e., the condition for switching between different CBFs. As is clear from the figure, the robot can successfully move to region B from A, stop for one second to avoid a collision, and then move back to region A through the provided distributions. The control input signals are depicted in Fig. 11. As shown in this figure, switching between the different safe sets, or between the QPs, does not lead to a discontinuity or sudden jump in the control inputs. The constant value of control inputs, during the one second "Stop" period, corresponds to the torque necessary to cancel gravity.

### *C. Case Study 3: Universal Robots UR5 6-Link Robot*

The equation of motion of the UR5 robot can be written in the form of (1) with the following parameters [37]

$$D(q) = \left[\sum_{i=1}^{6} m_i J_{v_i}^\top J_{v_i} + J_{w_i}^\top R_i I_{m_i} R_i^\top J_{w_i}\right], \tag{37}$$

where $m_i \in \mathbb{R}$ is the mass of the $i$th link, $J_{v_i} \in \mathbb{R}^{3\times 6}$ and $J_{w_i} \in \mathbb{R}^{3\times 6}$ are the linear and angular parts of the Jacobian matrix $J_i$, respectively. $R_i \in \mathbb{R}^{3\times 3}$ is the rotation matrix and $I_{m_i} \in \mathbb{R}^{3\times 3}$ is the inertia tensor. The elements of $C(q,\dot{q})$ are obtained from the inertia matrix as

$$c_{ij} = \sum_{k=1}^{6} \frac{1}{2}\left(\frac{\partial m_{ij}}{\partial q_k} + \frac{\partial m_{ik}}{\partial q_j} - \frac{\partial m_{kj}}{\partial q_i}\right)\dot{q}_k, \tag{38}$$

where $m_{ij}$ are the entries of the inertia matrix. The elements of the gravity vector are obtained from

$$K_i(q) = \frac{\partial \mathscr{P}}{\partial q_i}, \tag{39}$$

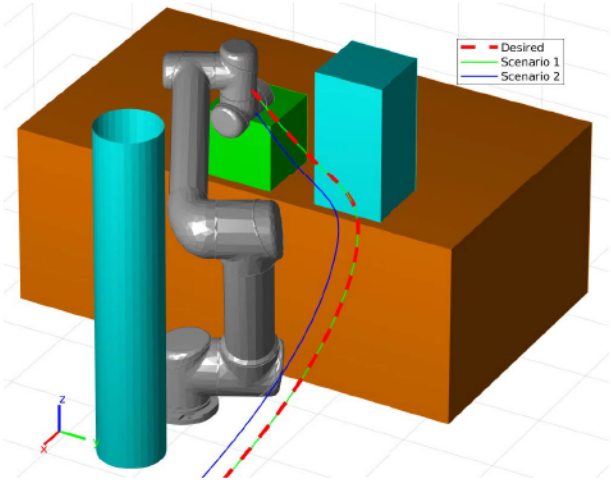



Fig. 12. Results of the proposed ProMP controller for the UR5.

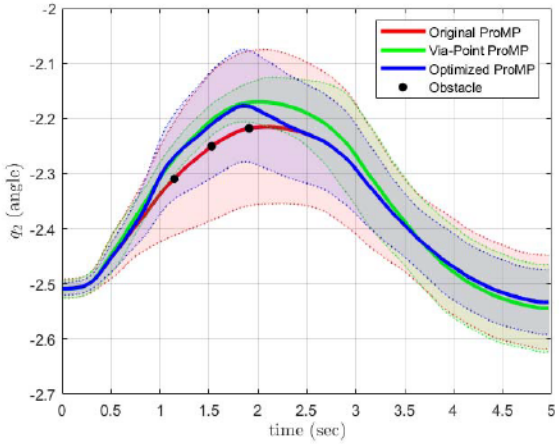



Fig. 13. Results of the modified ProMP trajectory for the second joint provided to the robot.

where $\mathscr{P}$ is the total potential energy of the robot. Additional information on these equations can be found in [38]. We generate 90 joint space trajectories with defined goals, obstacles, and starting positions. The 90 UR5 trajectories are then used to train a joint space ProMP using the same parameters as in the two-link robot case study. The following set of CLF and CBF parameters were chosen: $\epsilon_1 = \epsilon_2 = \epsilon_3 = \epsilon_4 = \epsilon_5 = 0.1$, $\epsilon_6 = 0.01$, and $c_{3i} = 1.1$, $a_{Eij} = 20.1$, $b_{Eij} = 1$, and $\gamma_i = 10.1$, $i \in \{1, \ldots, 6\}$, $j \in \{1, 2\}$. The simulation environment is depicted in Figs. 1 and 12.

We consider two different scenarios. In the *first scenario* $p_{sci} = 200$, which gives higher importance to the CLF. In the *second scenario* $p_{sci} = 0.001$, implying that the design interest and priority is on the CBF. As is clear from Fig. 12, in both scenarios the robot can effectively track the mean of the ProMP and simultaneously avoid colliding with obstacles.

## VI. Experimental Results

In this section, we provide an experimental evaluation of our CLF/CBF-based ProMP controller using a Universal Robots UR5e six-link manipulator with a Robotiq two-finger gripper. We selected three fixed locations to represent the *start*, *pick*, and *place* locations introduced as part of the FSM. The work environment included two static obstacles: the table that the robot was mounted on and a stack of books on the table.

The experiments were carried out as follows. First, a teacher demonstrates a pick-and-place procedure to the robot. Then, the robot must move from a *start* location to a *pick* location where a bin with small objects is located. From there, the robot travels to a *place* location while avoiding any static obstacles, i.e., the stack of books. Finally, once the robot deposits the objects in the place location bin, it returns to the start location.

We installed a gravity compensation controller on the robot to enable the teacher to directly manipulate the robot's joints via kinesthetic teaching. Using this directed learning from demonstration approach allows us to bypass the correspondence problem [4]. Additionally, kinesthetic teaching retains parity between the demonstration, learning, and execution space of the trajectories.

In all, ten demonstrations of the task were conducted for each of the three FSM states. Figs. 14 and 15 depict the demonstration operation as the operator moves the UR5e robot during the pick-and-place task, respectively, without and with obstacles. This grants sufficient variation in trajectories to allow obstacle avoidance. From these demonstrations, we collected the joint angles, velocities, and the associated time steps. It is worth mentioning that care must be taken to collect data that is roughly Gaussian, or at least unimodal (e.g., moving around an obstacle in the same direction each time). Methods to handle non-Gaussian or multimodal data is an issue to be addressed in future work.

Using the demonstration data, we trained 6 joint space ProMPs with $L = 55$ basis functions for a 6-link UR5 robot. We implemented the CLF/CBF-based ProMP controller with $p_{sci} = 0.5$. The other parameters were selected to be the same as the previous simulations in Section V-C. Figs. 16, 18, 17, and 19 show the UR5e robot executing the CLF/CBF-based ProMP controller trajectory for the pick-and-place task. Fig. 17 shows that following the ProMP mean trajectory will cause a collision when obstacles are present.

As discussed in Section III-B2, we demonstrated obstacle avoidance based on both the via-point and optimization-based processes. To do this, we modified the second joint (shoulder) trajectory to avoid obstacles during the experiments. In both the via-point and optimization-based procedures, the physical obstacle was represented using three point obstacles $\mathcal{O}_k$ along the trajectory at times $k = 1.14s, 1.53s$, and $1.9s$. Please note that in the current setup, the manipulator cannot identify the presence of obstacles. We hard-coded the position of the obstacles and generate the trajectories offline. As part of future work, we plan to utilize external sensing to perform monitoring of the environment and identify the presence of obstacles.

In the via-point modification, $\hat{q}_2(k)$ was specified to go through interest points $\{-2.27, -2.2, -2.17\}$ at times $k = 1.14$s, 1.53s, and 1.9s, which corresponds to the values of $q_2(k) + 0.03$. The resultant trajectory is shown in green in Fig. 13. Our optimization-based obstacle avoidance scheme was implemented using CVXPY [39], [40]. In the

Fig. 14. Operator demonstrating a pick-and-place task to a UR5e robot without obstacles.

Fig. 15. Operator demonstrating a pick-and-place task to a UR5e robot with an obstacle.

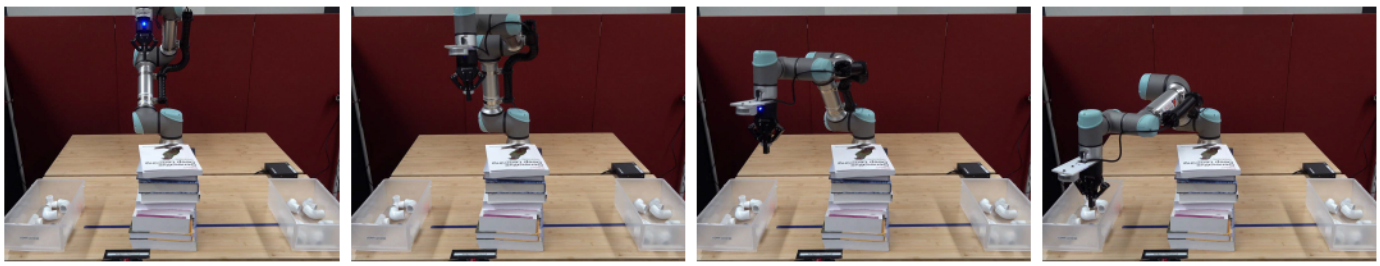


Fig. 16. Robot performing the initial pick action of the FSM.

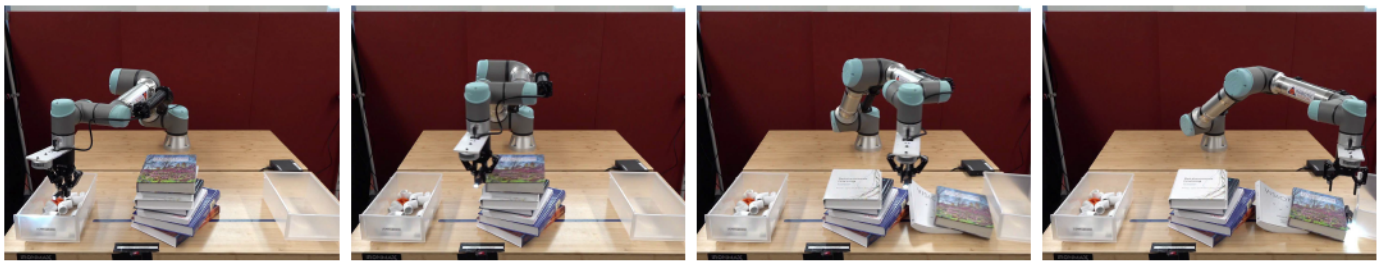


Fig. 17. Robot executing the mean ProMP, without any obstacle avoidance, thus causing it to hit the obstacle.

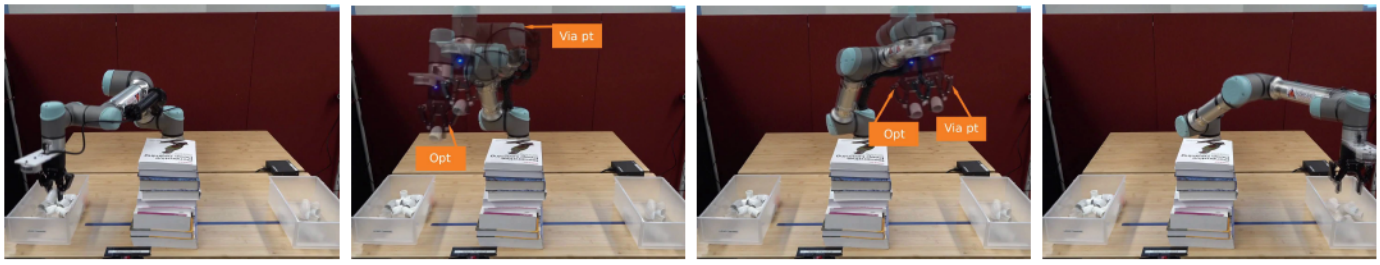



Fig. 18. Overlay showing the robot performing the pick to drop action of the FSM using both the optimization and via-point based trajectories.

optimization-based process, we specified three constraints to avoid the point obstacles and maintain a distance threshold, $D = 0.03$, from the ProMP trajectory while staying within the ProMP distribution. We chose $\lambda = diag([8, 1])$ to give more priority to the position of the trajectory over velocity. The subsequent trajectory is shown in blue in Fig. 13. A video of these experiments has been submitted as metadata and can be viewed from the publisher.

The computation time is arguably the major distinguishing factor between these approaches. Although the algorithmic

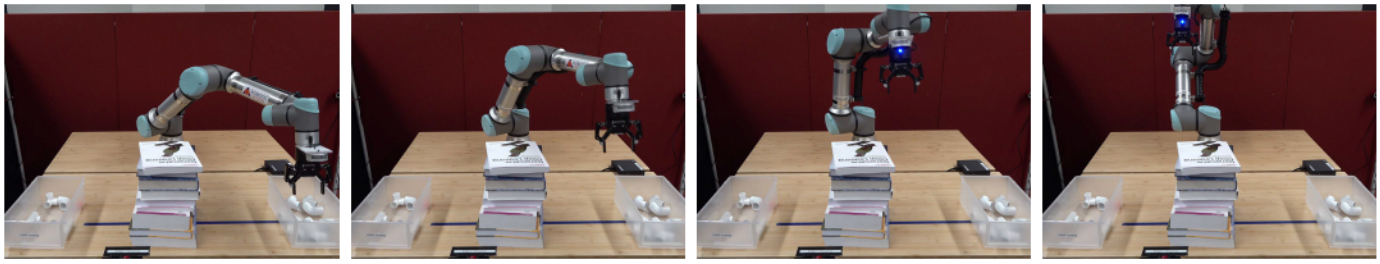


Fig. 19. Robot performing the drop to start action of the FSM.

time complexity of both the via-point and optimization-based modifications is $\mathcal{O}(n^2)$, the iterative-based CVX modification ran notably slower than the via-point method. We recorded the time to compute the via-point and optimization-based trajectory adjustments over multiple simulations. This included running both methods with 3, 6, and 10 different points for 100 times. The mean computation times for the via-point trials were 0.35ms, 0.56ms, and 0.98ms, respectively. The optimization process was completed in 0.7s, 1.12s, and 1.78s, respectively. These tests demonstrate that the via-point approach can run fast enough for use in real-time (assuming a 1 kHz update rate). On the other hand, the optimization-based method can be accommodated with a brief pause of the system. The degree to which the new distributions differ from the original is a second distinguishing factor. Compared to the via-points method, the optimization-based approach generally produces mean trajectories that deviate from the original mean later and return to it sooner. The optimization-based method also produces new variances that are closer to the original variance than the via-points procedure, which tends to produce a smaller variance than the original. It is worth noting that the optimization-based approach can generate trajectories similar to the via-point modification by adding more constraints to the optimization process, therefore making it a much more flexible technique.

In summary, both methods of obstacle avoidance can be applied to various robotics tasks depending on the system's time requirements and user preferences. The via-point should be used for higher-speed dynamic systems that must modify the trajectory in real-time such as autonomous vehicles, powered prosthetics, or high-speed/throughput manufacturing. Users will have to balance this with the knowledge that the trajectory generated may be less predictable, though still guaranteed to be safe. Systems that can safely pause such as delivery robots/vehicles not in traffic, rehabilitation robots, and low-throughput pick-and-place robots, can employ the optimization-based method to ensure greater control over the generated trajectory.

## VII. Ongoing and Future Work

A version of this work was a finalist for the Best Paper Award at the 2021 IEEE Conference on Automation Science and Engineering, leading to an invitation to this special issue of the Transactions on Automation Science and Engineering. There are a number of topics we are currently exploring or have identified for future efforts.

### A. Obstacle Avoidance Using CBFs

We have presented an approach for modifying a ProMP during runtime to avoid novel static obstacles. Dynamic obstacles will likely need to be avoided using the control law. One approach is to add a CBF for obstacle avoidance [24], [41]. Since obstacle avoidance could cause the state to leave the ProMP, we must determine how to ensure safety in unexplored areas and an eventual return to the ProMP. Additionally, dynamic constraints to velocity or position could be added to the QP to slow down or stop the robot in the presence of obstacles, but the effects on stability and convergence will need to be studied as will the solvability of the QP.

### B. Different Choices of CBFs

In this work, we focused on reciprocal CBFs for their relative simplicity and fast convergence. However, unbounded function values at boundaries may be undesirable when real-time/embedded implementations are considered. One possible solution will be zeroing barrier functions, where the barrier function vanishes on the set boundary. We plan to explore a mixture of CBFs for different tasks, such as zeroing CBFs for remaining in a safe trajectory and exponential CBFs for obstacle avoidance.

### C. Multidimensional and Non-Simply-Connected ProMPs

We designed the ProMP and CBF/CLF controller using $n_q$ (or $n_p$) independent trajectory distributions. However, the proposed approach can be extended to the case where we encode the coupling between the joints (or Cartesian elements) by using the $2n_q \times 2n_q$ (or $2n_p \times 2n_p$) covariance matrix. We are also interested in how to generate and work with ProMPs that are not simply connected. This can arise if there is an obstacle and trainers take different paths around it. The mean could pass through a non-safe space and thus the ProMP must either accommodate a hole or be split.

### D. Defining Unsafe Regions With ProMPs

We used a ProMP to define the safe region for the robot and CBFs to ensure that the robot remains in the distribution. We recently proposed a variation in which *unsafe* regions are defined by ProMPs, such as by monitoring where humans move through an area [42]. This gives an indication of where people are likely to be and the CBFs are defined to keep the robot out. We will investigate the interplay between safe-region ProMPs and unsafe-region ProMPs in regions where they intersect.

### E. Closed-Loop Object Grasping

Our work uses open-loop controllers to actuate the gripper based on the FSM state. Closed-loop approaches would increase the robustness and success rate for reaching and picking up objects of interest, including reattempts during a grasp failure, by considering gripper orientation and finger position.

## VIII. Conclusion

In this work, we developed a safe, rule-based control design approach for accomplishing complex manipulation scenarios modeled as multiple simpler subtasks and a FSM that switched between them. The subtasks that require robot movements were modeled by ProMPs. A ProMP-based robot guidance problem was solved using a CLF/CBF-based controller that can be designed in either the joint space or workspace. In each subtask, our proposed controller stabilizes the robot and guarantees that the system output is always inside the distribution generated by a ProMP. Moreover, the required condition for switching between different CLF/CBF-based ProMP controllers (i.e., different subtasks) was stated. Simulation and experimental studies on a 2-link and 6-link robot confirm the viability of the proposed method. Lastly, avenues of current and future work were discussed.

## References


[1] F. Vicentini, "Collaborative robotics: A survey," *J. Mech. Design*, vol. 143, no. 4, Apr. 2021, Art. no. 040802.

[2] D. Kragic, J. Gustafson, H. Karaoguz, P. Jensfelt, and R. Krug, "Interactive, collaborative robots: Challenges and opportunities," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 18–25.

[3] M. G. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robot. Autom. Syst.*, vol. 100, no. 2018, pp. 171–185, Feb. 2018.

[4] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auto. Syst.*, vol. 57, no. 5, pp. 469–483, 2009.

[5] S. Chernova and A. L. Thomaz, "Robot learning from human teachers," *Synth. Lectures Artif. Intell. Mach. Learn.*, vol. 8, no. 3, pp. 1–121, Apr. 2014.

[6] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics," *Auto. Robots*, vol. 42, no. 3, pp. 529–551, Mar. 2018.

[7] A. Paraschos, E. Rueckert, J. Peters, and G. Neumann, "Probabilistic movement primitives under unknown system dynamics," *Adv. Robot.*, vol. 32, no. 6, pp. 297–310, Mar. 2018.

[8] S. Calinon and D. Lee, "Learning control," in *Humanoid Robotics: A Reference*. Dordrecht, The Netherlands: Springer, 2017, pp. 1261–1312. [Online]. Available: https://link.springer.com/referencework/10.1007/978-94-007-7194-9

[9] S. Manschitz, M. Gienger, J. Kober, and J. Peters, "Probabilistic decomposition of sequential force interaction tasks into movement primitives," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 3920–3927.

[10] S. Niekum, S. Osentoski, G. Konidaris, and A. G. Barto, "Learning and generalization of complex tasks from unstructured demonstrations," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 5239–5246.

[11] S. Niekum, S. Osentoski, G. Konidaris, S. Chitta, B. Marthi, and A. G. Barto, "Learning grounded finite-state representations from unstructured demonstrations," *Int. J. Robot. Res.*, vol. 34, no. 2, pp. 131–157, 2015.

[12] S. Manschitz, J. Kober, M. Gienger, and J. Peters, "Learning to sequence movement primitives from demonstrations," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 4414–4421.

[13] S. Manschitz, J. Kober, M. Gienger, and J. Peters, "Learning movement primitive attractor goals and sequential skills from kinesthetic demonstrations," *Robot. Auton. Syst.*, vol. 74, pp. 97–107, Dec. 2015.

[14] X. Zhou, H. Wu, J. Rojas, Z. Xu, and S. Li, "Incremental learning robot task representation and identification," in *Nonparametric Bayesian Learning for Collaborative Robot Multimodal Introspection*. Cham, Switzerland: Springer, 2020, pp. 29–49.

[15] P. Wieland and F. Allgöwer, "Constructive safety using control barrier functions," *IFAC Proc. Volumes*, vol. 40, no. 12, pp. 462–467, 2007.

[16] C. Sloth, R. Wisniewski, and G. J. Pappas, "On the existence of compositional barrier certificates," in *Proc. IEEE 51st IEEE Conf. Decis. Control (CDC)*, Dec. 2012, pp. 4580–4585.

[17] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 3861–3876, Aug. 2017.

[18] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *Proc. 18th Eur. Control Conf. (ECC)*, Jun. 2019, pp. 3420–3431.

[19] W. S. Cortez, D. Oetomo, C. Manzie, and P. Choong, "Control barrier functions for mechanical systems: Theory and application to robotic grasping," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 2, pp. 530–545, Mar. 2021.

[20] B. T. Lopez, J.-J.-E. Slotine, and J. P. How, "Robust adaptive control barrier functions: An adaptive and data-driven approach to safety," *IEEE Control Syst. Lett.*, vol. 5, no. 3, pp. 1031–1036, Jul. 2021.

[21] M. Srinivasan and S. Coogan, "Control of mobile robots using barrier functions under temporal logic specifications," *IEEE Trans. Robot.*, vol. 37, no. 2, pp. 363–374, Apr. 2021.

[22] M. Saveriano and D. Lee, "Learning barrier functions for constrained motion planning with dynamical systems," 2020, *arXiv:2003.11500*.

[23] M. Rauscher, M. Kimmel, and S. Hirche, "Constrained robot control using control barrier functions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 279–285.

[24] F. Ferraguti et al., "Safety barrier functions and multi-camera tracking for human–robot shared environment," *Robot. Auto. Syst.*, vol. 124, Feb. 2020, Art. no. 103388.

[25] M. Srinivasan, C. Santoyo, and S. Coogan, "Continuous reachability task transition using control barrier functions," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9696–9701, 2020.

[26] M. Davoodi, A. Iqbal, J. M. Cloud, W. J. Beksi, and N. R. Gans, "Probabilistic movement primitive control via control barrier functions," in *Proc. IEEE 17th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2021, pp. 697–703.

[27] S. He, J. Zeng, B. Zhang, and K. Sreenath, "Rule-based safety-critical control design using control barrier functions with application to autonomous lane change," 2021, *arXiv:2103.12382*.

[28] A. Lazaric and M. Ghavamzadeh, "Bayesian multi-task reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 599–606.

[29] S. Gomez-Gonzalez, G. Neumann, B. Scholkopf, and J. Peters, "Adaptation and robust learning of probabilistic movement primitives," *IEEE Trans. Robot.*, vol. 36, no. 2, pp. 366–379, Apr. 2020.

[30] A. Colome and C. Torras, "Demonstration-free contextualized probabilistic movement primitives, further enhanced with obstacle avoidance," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 3190–3195.

[31] D. Koert, J. Pajarinen, A. Schotschneider, S. Trick, C. Rothkopf, and J. Peters, "Learning intention aware online adaptation of movement primitives," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3719–3726, Oct. 2019.

[32] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *Proc. 53rd IEEE Conf. Decis. Control*, Dec. 2014, pp. 6271–6278.

[33] S.-C. Hsu, X. Xu, and A. D. Ames, "Control barrier function based quadratic programs with application to bipedal robotic walking," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2015, pp. 4542–4548.

[34] H. Zhao, S. Kolathaya, and A. D. Ames, "Quadratic programming and impedance control for transfemoral prosthesis," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 1341–1347.

[35] J. P. Kolhe, M. Shaheed, T. S. Chandar, and S. E. Talole, "Robust control of robot manipulators based on uncertainty and disturbance estimation," *Int. J. Robust Nonlinear Control*, vol. 23, no. 1, pp. 104–122, Oct. 2011.

[36] J. A. Shah and S. Rattan, "Dynamic analysis of two link robot manipulator for control design using PID computed torque control," *Int. J. Robot. Autom.*, vol. 5, no. 4, pp. 277–283, 2016.

[37] M. W. Spong and M. Vidyasagar, *Robot Dynamics and Control*. Hoboken, NJ, USA: Wiley, 2008.

[38] K. Katharina, "Force estimation in robotic manipulators: Modeling, simulation and experiments," M.S. thesis, Dept. Eng. Cybern., Norwegian Univ. Sci. Technol., Trondheim, Norway, 2014. [Online]. Available: https://scholar.google.com/citations?user=c9lfwEwAAAAJ

[39] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *J. Mach. Learn. Res.*, vol. 17, no. 83, pp. 1–5, Apr. 2016.

[40] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd, "A rewriting system for convex optimization problems," *J. Control Decis.*, vol. 5, no. 1, pp. 42–60, 2018.

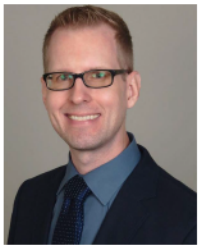
[41] T. Wei and C. Liu, "Safe control algorithms using energy functions: A uni ed framework, benchmark, and new directions," in *Proc. IEEE 58th Conf. Decis. Control (CDC)*, Dec. 2019, pp. 238–243.

[42] M. Davoodi, J. M. Cloud, A. Iqbal, W. J. Beksi, and N. R. Gans, "Safe human-robot coetaneousness through model predictive control barrier functions and motion distributions," in *Proc. Modeling, Estimation, Control Conf.*, 2021, pp. 271–277.

**Mohammadreza Davoodi** received the M.S. and Ph.D. degrees in electrical engineering from Tarbiat Modares University (Iran) in 2008 and 2012, respectively. He is currently an Assistant Professor at the Department of Electrical and Computer Engineering, The University of Memphis, where he also leads the Autonomous and Complex Systems Laboratory. His research interests include autonomous and control systems, robotics, multi-agent systems, and cyber-physical systems.

**Asif Iqbal** received the M.S. and Ph.D. degrees in electrical and electronics engineering from the University of Texas at Dallas in 2019. He is a Research Scientist at The University of Texas at Arlington Research Institute. His research interests are in the fields of robotics, machine vision, and autonomous systems.

**Joseph M. Cloud** (Student Member, IEEE) received the B.S. degree *(summa cum laude)* in computer engineering from the Honors College, The University of Texas at Arlington, in 2019, where he is currently pursuing the Ph.D. degree. His research interests lie in machine learning and computer vision applied to space robotics, robotic manipulation, and human–robot interaction.



**William J. Beksi** (Member, IEEE) received the B.S. degree in mathematics and computer science from the Stevens Institute of Technology in 2002, and the M.S. and Ph.D. degrees in computer science from the University of Minnesota in 2016 and 2018, respectively. He is currently an Assistant Professor at the Department of Computer Science and Engineering, The University of Arlington at Texas, where he also leads the Robotic Vision Laboratory. His research interests include robotics, computer vision, and cyber-physical systems.

**Nicholas R. Gans** (Senior Member, IEEE) received the B.S. degree in electrical engineering from Case Western Reserve University in 1999, the M.S. degree in electrical and computer engineering in 2002, and the Ph.D. degree in systems and entrepreneurial engineering from the University of Illinois Urbana–Champaign in 2005. He is the Division Head of autonomy and intelligent systems at The University of Texas at Arlington Research Institute. His research interests are in the fields of robotics, nonlinear and adaptive control, and machine vision.