# Probabilistic Movement Primitive Control via Control Barrier Functions

Mohammadreza Davoodi[1], Asif Iqbal[1], Joseph M. Cloud[2], William J. Beksi[2], and Nicholas R. Gans[1]

***Abstract*— In this paper, we introduce a novel means of control design for probabilistic movement primitives (ProMPs). ProMPs are a powerful tool to define a distribution of trajectories for robots or other dynamic systems. However, existing control methods to execute desired motions suffer from a number of drawbacks such as a reliance on linear control approaches and sensitivity to initial parameters. We propose the use of feedback linearization, quadratic programming, and multiple control barrier functions to guide a 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 firm stability and known bounds on the system state. Furthermore, we highlight how the proposed method may allow a designer to emphasize certain objectives that are more important than the others. A series of simulations demonstrate the efficacy of our approach and show it can run in real time.**

## I. Introduction

Research in robot motion planning has provided a variety of successful frameworks to generate trajectories [1]. However, these frameworks can be difficult to implement in environments without a predefined static map. Furthermore, motion planning typically requires the selection of algorithms, the design of cost functions, and other tasks that are outside the expertise of nontechnical users. Learning from demonstration is a paradigm that has played an important role in addressing the issue of scaling up robot learning [2], [3]. It can avert the drawbacks of traditional robot motion planning by relying on the presence of a human teacher.

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. MPs are modeled through a compact representation of implicitly continuous and high-dimensional trajectories. For example, dynamic motion primitives (DMPs) use demonstrations to learn a model of the control effort necessary to produce a desired trajectory for a stabilized dynamic system [4], [5]. However, DMPs can be inflexible in that they can only deliver the demonstrated trajectory. Capturing the natural variation in a human demonstration of a task can help a robot overcome uncertainty and deviation between the training regime and actual task execution [6].

[1]The authors are with the University of Texas at Arlington Research Institute, Fort Worth, TX, USA. Emails: mdavoodi@uta.edu, asif.iqbal@uta.edu, nick.gans@uta.edu.

[2]The authors are with the Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, TX, USA. Emails: joe.cloud@mavs.uta.edu, william.beksi@uta.edu.

This material is based upon work supported by the National Science Foundation (NSF) under grant No. 1728057 and the University of Texas at Arlington Research Institute. Joseph M. Cloud was supported by an NSF Graduate Research Fellowships Program grant No. 1746052.

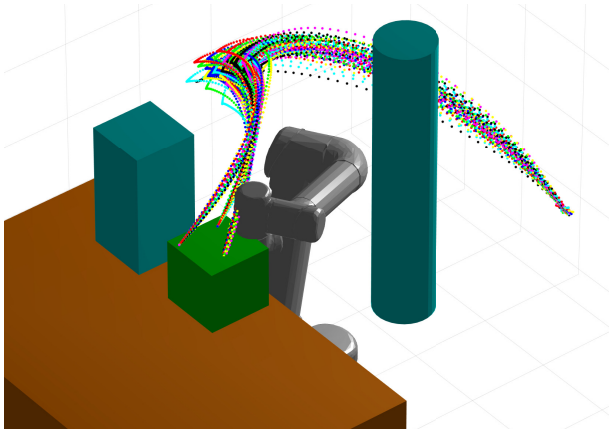

Fig. 1: A set of robot trajectories generated by our demonstration and control method. The controller guarantees the system never leaves a neighborhood defined by the training set and provides a simple way to define trajectories that enforce safety constraints.

Probabilistic movement primitives (ProMPs) are a concept whereby a *distribution* of trajectories is learned from multiple demonstrations. In [7], 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 [8]. In [9], the authors designed a model predictive control-based ProMP controller for a linear discrete time system model.

Nevertheless, ProMP methods suffer from some 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 such as robotics and autonomous vehicles. In addition, ProMPs and their resulting controllers are often difficult to implement, sensitive to noise, and highly dependent upon design parameters and initial conditions. This reduces their usability for non-experts. Finally, since ProMPs are by definition stochastic, the distribution of trajectories may have a large support and result in trajectories that deviate from the training set.

Real-time safety in safety-critical dynamical systems is an important problem that has received attention in recent years [10]. This problem 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 [11]. The concept of barrier certificates/functions was extended for synthesizing real-time safe control laws for dynamical systems via control barrier functions (CBFs) [12].

CBFs integrate seamlessly with control Lyapunov functions (CLFs) to offer stability while respecting limits and

safe regions of the state space [13]–[15]. Additionally, 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. One downside to CBFs and CLFs is the complexity in defining the barriers and trajectories. Recent efforts to automate the definition of CBFs and CLFs include mapping temporal logic statements with respect to performance requirements [16] and fitting piecewise-linear barrier functions to trained obstacles or regions [17].

In this work, we propose a novel control method that uses the distribution delivered by a ProMP to define a set of CLFs and CBFs. This idea addresses each of the previously mentioned issues with ProMPs, CLFs, and CBFs. CLF and CBF controllers are fundamentally based on the nonlinear kinematic/dynamic equations of the system and overcome the inherent linearity of ProMP controllers. 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. Moreover, our approach provides a simple way to define trajectories for CLFs and barriers for CBFs using the mean, standard deviation, or other moments of the distribution. We provide a stability analysis for the proposed control laws, and we demonstrate their effectiveness and computational efficiency through simulations of a 2-link and a 6-link robot. Examples of generated control trajectories for a Universal Robots UR5 are shown in Fig. 1.

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

## II. Background

In this section we provide essential background information on ProMPs, CBFs, and CLFs.

**Notation:** Given a matrix $A$, let $A^\top$ denote its transpose. Let the identity and zero matrices, with appropriate dimensions, be denoted by $I$ and 0, respectively. We denote $\star$ as the symmetric entries of a matrix. For a vector field $f_i(x)$ and vector of vector fields $F(x) = [f_1(x), ..., f_n(x)]$, 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 $f_i(x)$ : $L_F = [L_{f_1}, ..., L_{f_n}]$. Zero-mean i.i.d. Gaussian distribution with mean $m$ and (co)variance $\Sigma$ is denoted $\mathcal{N}(m, \Sigma)$, and the number of joints for a robot arm is represented by $n$.

### A. Probabilistic Movement Primitives

ProMPs provide a parametric representation of trajectories that can be executed in multiple ways through the use of a probability distribution. Basis functions are used to reduce 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 or task spaces). We consider joint space trajectories.

Within a ProMP, the execution of a trajectory is modeled as the set of robot positions, $\zeta_i = \{q_i(k)\}$, where $k = 0, \dots, K$, $q_i(k) \in \mathbb{R}$ is the state variable sampled at time $k$, and $i \in \{1, \dots, n\}$ is the joint index of a robot. Let $w_i \in \mathbb{R}^{1\times L}$ be a weight matrix with $L$ terms. A linear basis function model is then given by

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

where $\Phi(k) = \begin{bmatrix} \phi(k) & \dot{\phi}(k) \end{bmatrix}^\top \in \mathbb{R}^{2\times L}$ is the time-dependent basis function matrix, and $L$ is the number of basis functions. Gaussian noise is described 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

$$p(\zeta_i, \theta_i) = \int p(\zeta_i \,|\, w_i) p(w_i; \theta_i) dw_i. \tag{1}$$

Here, the distribution $p(\zeta_i, \theta_i)$ defines a hierarchical Bayesian model over the trajectories $\zeta_i$ [7] and $p(w_i \,|\, \theta_i) = \mathcal{N}(w_i \,|\, \mu_{w_i}, \Sigma_{w_i})$. In an MP representation, the parameters of a single primitive must be easy to obtain from demonstrations. The distribution of the state $p(x_i(k)\,;\, \theta_i)$ is

$$p(x_i(k); \theta_i) = \mathcal{N}(x_i(k) \,|\, \Phi(k)\mu_{w_i}, \Phi(k)\Sigma_{w_i}\Phi(k)^\top + \Sigma_{x_i}). \tag{2}$$

The trajectory can be generated from the ProMP distribution using $w_i$, the basis function $\Phi(k)$, and (2). The basis function is chosen based on the type of robot movement which can be either rhythmic or stroke-based. From (2), the mean $\tilde{\mu}_i(k) \in \mathbb{R}^2$ of the ProMP trajectory at $k$ is $\Phi(k)\mu_{w_i}$ and the covariance $\Sigma_i(k)$ is $\Phi(k)\Sigma_{w_i}\Phi(k)^\top + \Sigma_{x_i}$.

Multiple demonstrations are needed to learn a distribution over $w_i$. We use a combination of radial basis and polynomial basis functions for training the ProMP. From the demonstrations, the parameters $\theta_i$ can be estimated using maximum likelihood estimation [18]. However, this may result in unstable estimates of the ProMP parameters when there are insufficient demonstrations. Our method uses regularization to estimate the ProMP distribution similar to the work by Gomez-Gonzalez *et al.* [6]. We maximize $\theta_i$ for the posterior distribution over the ProMP using expectation maximization,

$$p(\theta_i \,|\, x_i(k)) \propto p(\theta_i) p(x_i(k) \,|\, \theta_i). \tag{3}$$

In addition, we use a Normal-Inverse-Wishart as a prior distribution $p(\theta_i)$ to increase stability when training the ProMP parameters [6].

### B. Control Barrier and Control Lyapunov Functions

Consider the affine, nonlinear system

$$\dot{x} = f(x) + \tilde{G}(x)u, \tag{4}$$

where $x \in \mathbb{R}^n$ denotes the state, $u \in \mathbb{R}^m$ is the control input, $\tilde{G} = [g_1, ..., g_m]$, 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 (4) is controllable.

*1) Control Barrier Functions:* A smooth function $h(x) : \mathbb{R}^n \to \mathbb{R}$ is defined to encode a constraint on the state $x$ of system. The constraint is satisfied if $h(x) > 0$ and violated if $h(x) \leq 0$. Consider the set $C$ defined by

$$\begin{aligned} C &= \{\eta : h(\eta) \geq 0\}, \\ \partial C &= \{\eta : h(\eta) = 0\}, \\ \text{Int}(C) &= \{\eta : h(\eta) > 0\}, \end{aligned} \tag{5}$$

where $h : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function.

Existing approaches to define CBFs include exponential CBFs, zeroing CBFs, and reciprocal CBFs [12], [19]. However, these methods have trade-offs in terms of ease of definition, boundedness of velocities, speed of convergence, etc. We investigate the use of a reciprocal CBF. This type of CBF has a small value when the states are far from the constraints and becomes unbounded when the states approach the constraint.

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

$$\begin{aligned} \frac{1}{\alpha_1(h(x))} \leq B(x) \leq \frac{1}{\alpha_2(h(x))}, \\ L_f B(x) + L_{\tilde{G}} B(x) u - \frac{\gamma}{B(x)} \leq 0. \end{aligned} \tag{6}$$

*2) Control Lyapunov Functions:* A CLF is a Lyapunov function for a dynamical system with inputs, and it can be used to design the control inputs to ensure objectives such as stability, convergence to the origin (or other set point), or convergence to a desired trajectory. In order to have similar constructions as CBFs, we will consider exponentially stabilizing CLFs [20].

**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

$$\begin{aligned} c_1 ||x||^2 \leq V(x) \leq c_2 ||x||^2, \\ L_f V(x) + L_{\tilde{G}} V(x) u + c_3 V(x) \leq 0. \end{aligned} \tag{7}$$

Having established a CBF (to accomplish safety) and CLF (to achieve control performance objectives), they can be combined through a QP. Consequently, safe control laws may be computed using the QP to solve the constrained optimization problems at each point in time [12].

## III. Problem Formulation

### A. System Modeling

We consider the equations of motion for a robot given in the general form by the Euler-Lagrange equations

$$D(q)\ddot{q} + H(q, \dot{q}) = Eu, \tag{8}$$

where $q \in Q$ is a generalized coordinate position. Suppose that $Q \subset \mathbb{R}^n$ is the configuration space of the robot, $D(q)$ 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 $E$ is the actuation matrix that determines the way in which the inputs, $u$, actuate the system. Then, the system description in (8) can be converted to an ODE of the form in (4) where $x = [q, \dot{q}]^\top$ and

$$f(x) = \begin{bmatrix} \dot{q} \\ -D^{-1}(q)H(q, \dot{q}) \end{bmatrix}, \; g(x) = \begin{bmatrix} 0 \\ D^{-1}(q)E \end{bmatrix}. \tag{9}$$

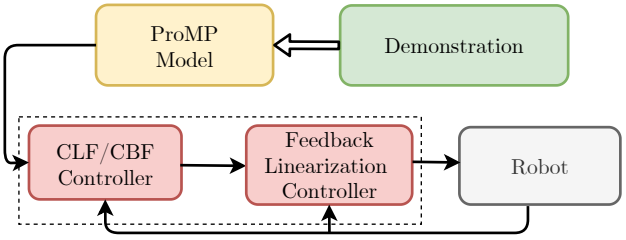

Fig. 2: A block diagram illustrating the general structure of the proposed system.

### B. Problem Definition

Our primary goal is to design a controller such that the system output tracks a trajectory within the distribution generated by a ProMP. To this end, we first construct a nonlinear inner-loop control law founded on the feedback linearization of (8). Then, an outer-loop controller based on a CLF-CBF is designed using the distribution parameters, $\tilde{\mu}_i$ and $\Sigma_i$. This is summarized as the following problem objectives.

1) Design a feedback linearization controller to obtain a linear and decoupled input-output closed-loop relationship for the error signal.
2) Use the demonstrated trajectories of a robot to train and estimate a ProMP distribution. The ProMP provides the time-varying mean and variance of a trajectory.
3) Design a CLF to stabilize the system such that $\forall i,\ q_i \to \mu_i$, where $\mu_i$ is the first element of $\tilde{\mu}_i$.
4) Design a CBF to guarantee that the error $e_i = q_i - \mu_i$ satisfies the safety constraint $\forall i, |e_i| < \sigma_i$, where $\sigma_i$ is the $(1, 1)$ element of $\Sigma_i$.

The general structure of our proposed system is shown in Fig. 2.

## IV. Control Construction

We define the error and trajectory vectors as $e = [e_1, \ldots, e_n]^\top$ and $\mu = [\mu_1, \ldots, \mu_n]^\top$, respectively. Using (4) and taking the derivative $r$ times along $f(x)$ and $g(x)$, we obtain

$$e^{(r)}(x) = L_f^{(r)} e(x) + \underbrace{L_g L_f^{(r-1)} e(x)}_{\Gamma} u - \mu^{(r)}. \tag{10}$$

Assume the decoupling matrix, $\Gamma$, is well-defined and has full rank. This implies that the system in (4) is feedback linearizable and we can prescribe the control law

$$u(x) = \Gamma^{-1} \left( -L_f^{(r)} e(x) + \mu^{(r)} + v \right), \tag{11}$$

where $v$ is an auxiliary feedback control value. This yields the $r$th order linear system from input $v$ to output $e$,

$$e^{(r)} = v. \tag{12}$$

In this work, we limit ourselves to relative degree-two systems (i.e., $r = 2$). By defining $\eta = [e,\ \dot{e}]^\top$, (12) can be written as a linear time invariant system

$$\dot{\eta} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \eta + \begin{bmatrix} 0 \\ I \end{bmatrix} v. \tag{13}$$

From there, $n$ decoupled systems can be obtained from (13),

$$\dot{\eta}_i = F\eta_i + Gv_i, \quad i = 1, \ldots, n, \tag{14}$$

where $\eta_i = [e_i,\ \dot{e}_i]^\top$, $F = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$, and $G = [0\ 1]^\top$.

To accomplish problem objective 3, it is sufficient to ensure $e_i \to 0$. This is accomplished through designing an appropriate CLF. To satisfy problem objective 4, 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 (14). Moreover, the $i$th controller for each system in (14) is designed by combining the corresponding CLFs and CBFs through a QP problem.

### A. Control Lyapunov Function

Consider the following rapidly exponentially stabilizing-CLF (RES-CLF) [21],

$$V_{\epsilon_i}(\eta_i) = \eta_i^\top \begin{bmatrix} 1/\epsilon_i I & 0 \\ 0 & I \end{bmatrix} P \begin{bmatrix} 1/\epsilon_i I & 0 \\ 0 & I \end{bmatrix} \eta_i, \tag{15}$$

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

$$F^\top P + PF - PGG^\top P + I = 0. \tag{16}$$

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

$$\dot{V}_{\epsilon_i}(\eta_i) = L_F V_{\epsilon_i}(\eta_i) + L_G V_{\epsilon_i}(\eta_i) v_i \leq -\frac{c_{3i}}{\epsilon_i} V_{\epsilon_i}(\eta_i), \tag{17}$$

where $c_{3i}$ is a positive constant value. To guarantee a feasible solution for the QP, the CLF constraint can be relaxed by $\delta_i > 0$ [20] resulting in

$$L_F V_{\epsilon_i}(\eta_i) + L_G V_{\epsilon_i}(\eta_i) v_i + \frac{c_{3i}}{\epsilon_i} V_{\epsilon_i}(\eta_i) \leq \delta_i. \tag{18}$$

This relaxation parameter will be minimized in the QP cost function. It is worth mentioning that by providing a weighting factor on the relaxation parameter $\delta_i$, the QP can mediate a trade-off among performance and safety objectives in a way that safety is always satisfied.

### B. Control Barrier Functions

We propose two safety constraints for each system in (14). More specifically, each system should satisfy $-\sigma_i < e_i < \sigma_i$. Consequently, we have the following two safety constraints

$$\begin{aligned} h_{i1} &= e_i + \sigma_i, \\ h_{i2} &= -e_i + \sigma_i. \end{aligned} \tag{19}$$

From (19), it is obvious that we have multiple time-varying constraints that should be satisfied simultaneously. Moreover, it is easy to verify that $L_G e_i = 0$ and $L_F L_G e_i \neq 0$, thus the safety constraint has a relative degree of 2. For the relative degree-two constraints, the reciprocal CBF is defined as [22],

$$B_j(\eta_i) = -\ln\left(\frac{h_{ij}(\eta_i)}{1 + h_{ij}(\eta_i)}\right) + a_{Eij} \frac{b_{Eij} \dot{h}_{ij}(\eta_i)^2}{1 + b_{Eij} \dot{h}_{ij}(\eta_i)^2}, \tag{20}$$

where $j \in \{1, 2\}$ and $a_{Eij}$, $b_{Eij}$ are positive scalars. Note that by choosing small values for $a_{Eij}$ and $b_{Eij}$, the system will stop far from the constraint surfaces. By choosing large parameters, the system will stop close to the constraints. In some cases, especially in the presence of uncertainties, choosing $a_{Eij}$ and $b_{Eij}$ to be too large may cause constraint violations (i.e., no solution exists to the QP problem). As a result, based on the given application, a compromise needs be considered for choosing these parameters. The following control barrier condition should be satisfied for time varying constraints (which leads to time varying CBFs),

$$L_F B_j(\eta_i) + L_G B_j(\eta_i) v_i + \frac{\partial B_j(\eta_i)}{\partial t} - \frac{\gamma_i}{B_j(\eta_i)} \leq 0. \tag{21}$$

### C. Quadratic Program (QP)

As shown in (19), two safety constraints need to be satisfied simultaneously for each linearized, decoupled system. Due to this fact, a single controller can be obtained in such a way that guarantees adherence to both constraints [23]. In this subsection, $n$ QPs are proposed to unify RES-CLF and CBFs for each system in (14) into a single controller. The $n$ QPs for $i \in \{1, \ldots, n\}$ are defined as

$$\min_{\mathbf{v}_i = [v_i,\, \delta_i]^\top \in \mathbb{R}^2} \mathbf{v}_i^\top H_i \mathbf{v}_i,$$

subject to

$$\begin{aligned} &L_F V_{\epsilon_i}(\eta_i) + L_G V_{\epsilon_i}(\eta_i) v_i + \frac{c_{3i}}{\epsilon_i} V_{\epsilon_i}(\eta_i) \leq \delta_i \quad \text{(CLF)} \\ &L_F B_j(\eta_i) + L_G B_j(\eta_i) v_i + \frac{\partial B_j(\eta_i)}{\partial t} \leq \frac{\gamma_i}{B_j(\eta_i)} \ \text{(CBFs)} \end{aligned} \tag{22}$$

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.

**Remark 1.** We design the ProMP generation and controller formulation such that the safety constraints are defined on the independent variance of each joint trajectory distribution. However, the proposed approach can be easily extended to the case where we want to encode the coupling between the joints for ProMP training. Thus, the safety constraints should be defined based on the covariance matrix. To guarantee safety in the presence of covariance, we can impose the following confidence support

$$e^\top \Sigma^{-1} e \leq k_p, \quad \text{with probability } 1 - p. \tag{23}$$

The covariance matrix $\Sigma$ is a symmetric positive definite matrix. Consequently, it is possible to decompose it into $\Sigma = \Psi^\top \Lambda \Psi$, where $\Lambda$ is the diagonal eigenvalue matrix and $\Psi$ is the orthogonal eigenvector matrix. Due to this fact, from (23) we get

$$\Psi^\top \Lambda^{-1} \Psi = \sum_i \nu_i^\top \lambda_i^{-1} \nu_i \geq \nu_i^\top \lambda_i^{-1} \nu_i \quad \forall i = 1, \ldots, n,$$

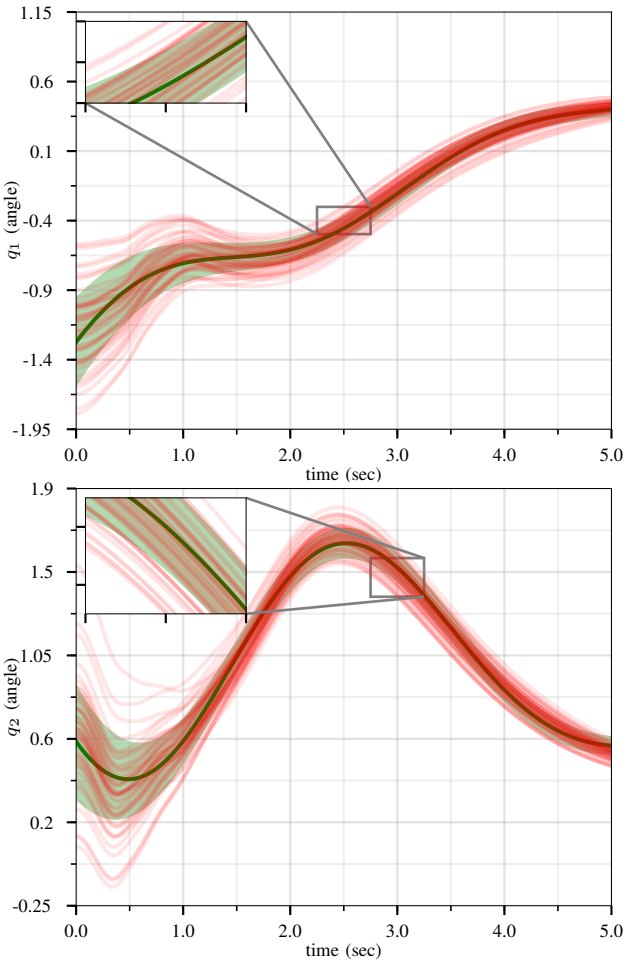


Fig. 3: Training of the ProMPs for the first joint (top) and second joint (bottom). The 50 input trajectories are shown in red and the ProMP mean joint trajectories ($\mu_i$) are shown in dark green. A light-green fill shows $\mu_i \pm \sigma_i$.

where $\nu_i$ and $\lambda_i$ denote, respectively, the eigenvectors and the eigenvalues of the $\Sigma$. This leads to the polytopic bounds

$$-\sqrt{k_p \lambda_i} \leq \nu_i^\top e \leq \sqrt{k_p \lambda_i}, \quad \forall i = 1, \ldots, n. \tag{24}$$

These conditions can be represented in the form of linear constraints for the QP. As a result, by solving the QP problem in the presence of these constraints, safety is assured with probability of at least $1 - p$.

## V. Simulations

In this section we demonstrate different aspects and capabilities of our methodology for the ProMP control of a robotic system. The system models and proposed real-time controller are simulated using a MATLAB 2019a environment. 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: Two-Link Robot

We consider a rigid, two-link robot with the dynamic model of (8) and the following parameters [24]

$$D(q) = \begin{bmatrix} m_1 l_1^2 + m_2(l_1^2 + l_2^2 + 2 l_1 l_2 \cos(q_2)) & * \\ m_2(l_2^2 + l_1 l_2 \cos(q_2)) & m_2 l_2^2 \end{bmatrix},$$

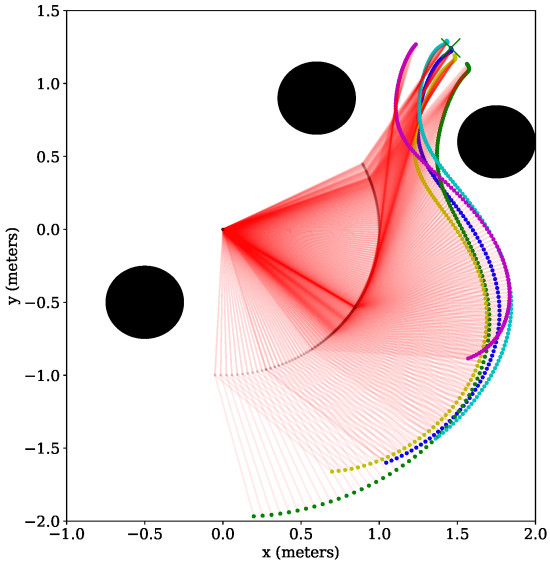


Fig. 4: A visualization of the 2-link ProMP in the robot workspace. The robot link positions over time are shown in red, while the end-effector trajectory following the mean ProMP joint trajectories is shown in green. The four trajectories of the end effector with joint combinations $\mu_i \pm \sigma_i$, $i \in \{1, 2\}$ are illustrated in cyan, blue, yellow and magenta. The black circles correspond to obstacles.

$$C(q, \dot{q}) = \begin{bmatrix} -m_2 l_1 l_2 \sin(q_2) \dot{q}_2 (2\dot{q}_1 + \dot{q}_2) \\ m_2 l_1 l_2 \dot{q}_1^2 \sin(q_2) \end{bmatrix},$$

$$K(q) = \begin{bmatrix} (m_1 + m_2) g l_1 \sin(q_1) + m_2 g l_2 \sin(q_1 + q_2) \\ m_2 g l_2 \sin(q_1 + q_2) \end{bmatrix},$$

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. 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 three obstacles. Using this dataset, we train a ProMP with Algorithm 1 from [6]. We use $L = 2$ basis functions consisting of five radial basis parameters. The results of the ProMP training are presented in Fig. 4 where the black circles indicate the location of obstacles and the 50 input trajectories shown in red. The ProMP mean joint trajectories, $\mu_i$, are shown in dark green, and in a light-green fill we show $\mu_i \pm \sigma_i$. A visualization of the ProMP in the workspace, based on [25], is displayed in Fig. 4. The robot link positions over time are highlighted in red with different colors representing key end-effector trajectories from the ProMP. The green trajectory is the mean of the ProMP distribution. The other four trajectories result from combinations of $\mu_i \pm \sigma_i$, $i \in \{1, 2\}$.

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

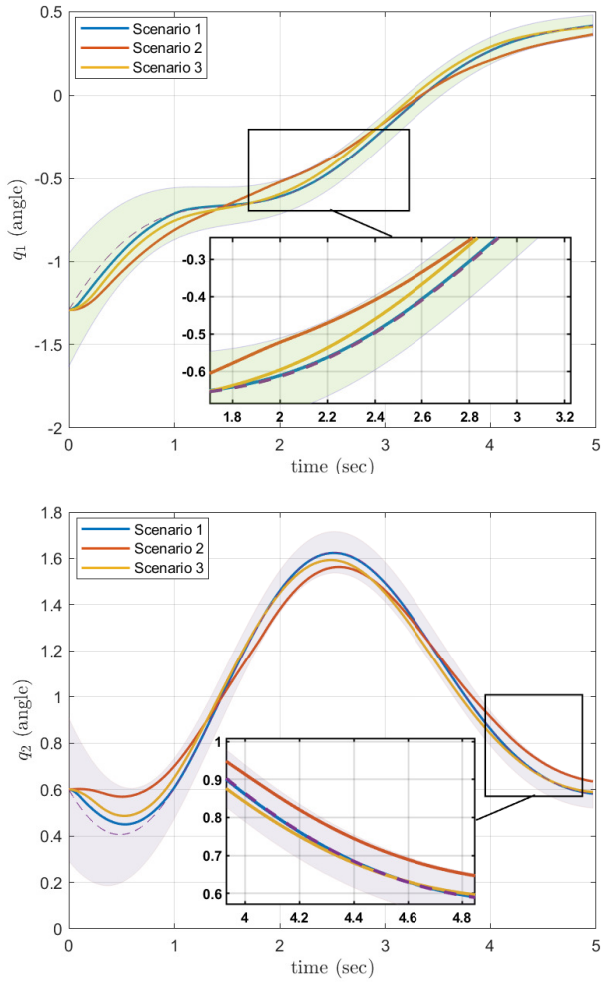


Fig. 5: The results of the CLF/CBF-based ProMP controller for the first joint (top) and second joint (bottom). The safe region of $\mu_i \pm \sigma_i,\ i \in \{1, 2\}$ is shown as a filled "tube". The control results in different trajectories for distinct values of the weight $p_{sci}$, but all trajectories remain safe.

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. 5. 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 proposed method provides a valuable option to the designer that grants levels of flexibility of the trajectory while ensuring safety.

The main computational cost with respect to time of our controller comes from the fact that it has to solve a set of QPs at every time step. In the performed simulations, two QP problems are solved in real time (one for each link). The average required time ($T_{\text{ave}}$), maximum time ($T_{\text{max}}$), and the standard deviation (std) for solving the QP problems are $T_{\text{ave}} = [0.0015,\ 0.0011]$, $T_{\text{max}} = [0.1148,\ 0.0119]$, std $= [0.0053,\ 0.0006]$ where the units are seconds. From these results, it is clear that the expected execution time of the QP problems is very small (e.g., in the range of 1 millisecond). The large maximum times are each a single outlier. Hence, the controller is appropriate for a real-time implementation.

### *B. Case Study 2: Universal Robots UR5 6-Link Robot*

The equation of motion of the UR5 robot can be written in the form of (8) with the following parameters [26]

$$D(q) = \left[\sum_{i=1}^{6} m_i J_{v_i}^\top J_{v_i} + J_{w_i}^\top R_i I_{m_i} R_i^\top J_{w_i}\right], \tag{25}$$

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 part 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

$$c_{ij} = \sum_{k=1}^{6} \frac{1}{2}\left(\frac{\partial m_{ij}}{\partial q_k} + \frac{\partial m_{ik}}{\partial q_j} - \frac{\partial m_{kj}}{\partial q_i}\right)\dot{q}_k, \tag{26}$$

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

$$K_i(q) = \frac{\partial \mathcal{P}}{\partial q_i}, \tag{27}$$

where $\mathcal{P}$ is the total potential energy of the robot. Additional information on these equations can be found in [27].

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 are 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 $\gamma_i = 10.1$, $i \in \{1, ..., 6\}, j \in \{1, 2\}$. The simulation environment is depicted in Figs. 1 and 6.

We consider two different scenarios. In the *first scenario* $p_{sci} = 200$, which gives higher importance to the CLF. In the *second scenario* $p_{sci} = 0.001$, which implies the design interest and priority is on the CBF. As is clear from Fig. 6, in both scenarios the robot can effectively track the mean of the ProMP and simultaneously avoid colliding with environmental obstacles. The running time statistics for solving the QP problems are $T_{\text{ave}} = [0.0014,\ 0.0011,\ 0.0010,\ 0.0010,\ 0.0011,\ 0.0010]$, $T_{\text{max}} = [0.1193,\ 0.0120,\ 0.0128,\ 0.0058,\ 0.0314,\ 0.0028]$, std $= [0.0053,\ 0.0005,\ 0.0005,\ 0.0003,\ 0.0014,\ 0.0002]$. Again, the large maximum values are for a single outlier, thus the expected operation time is well within the needs of a robotic system. The cause of outlier run times is an avenue of future research to offer improved performance guarantees.

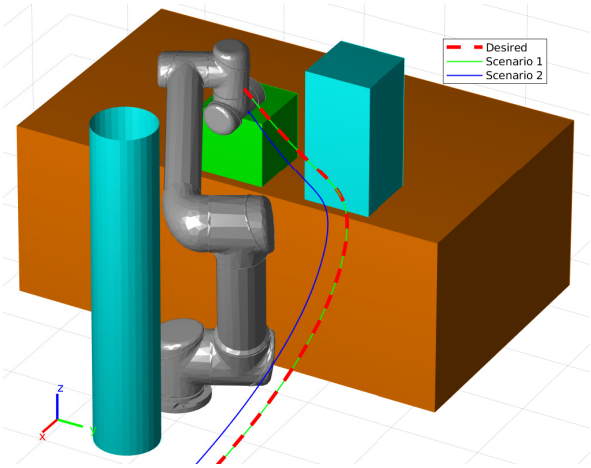

Fig. 6: The results of the proposed ProMP controller for the UR5.

## VI. Conclusion and Future Work

In this work, a ProMP robot guidance problem was solved using a CLF/CBF-based controller. The proposed approach stabilizes the robot and guarantees the system output is always inside the distribution generated by a ProMP. The time-varying nature of the ProMP ensures the robot is guided along the distribution at the desired rate. Moreover, using our proposed method, it is possible to provide a trade-off between trajectory performance and safety objectives. These safety guarantees and performance tuning options are not available in the native ProMP control design. Our simulation studies on a 2-link and 6-link robot confirm the viability of the proposed method for designing the controller.

There are several topics of ongoing work. We are investigating the trade-offs of various different CBFs (e.g., zeroing versus reciprocal), other choices of cost functions, and constraints in the QP. Additionally, we are seeking novel methods that automatically define additional barriers to ensure the safe movement of a co-robot around dynamic obstacles (e.g., humans). This includes the exploration of an active learning framework for ProMPs, whereby a co-robot has the capability to detect when it is outside of its safe trajectory zone and thus requires assistance from a human. Finally, experimental validation of the proposed methodology will be carried out.

## References

[1] M. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robotics and Autonomous Systems*, vol. 100, pp. 171–185, 2018.

[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] S. Chernova and A. L. Thomaz, "Robot learning from human teachers," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 8, no. 3, pp. 1–121, 2014.

[4] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[5] A. Dahlin and Y. Karayiannidis, "Adaptive trajectory generation under velocity constraints using dynamical movement primitives," *IEEE Control Systems Letters*, vol. 4, no. 2, pp. 438–443, 2019.

[6] S. Gomez-Gonzalez, G. Neumann, B. Schölkopf, and J. Peters, "Adaptation and robust learning of probabilistic movement primitives," *IEEE Transactions on Robotics*, vol. 36, no. 2, pp. 366–379, 2020.

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

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

[9] S. Calinon and D. Lee, "Learning control," *Humanoid Robotics: A Reference*, pp. 1261–1312, 2017.

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

[11] C. Sloth, R. Wisniewski, and G. J. Pappas, "On the existence of compositional barrier certificates," in *Proceedings of the IEEE Conference on Decision and Control*, 2012, pp. 4580–4585.

[12] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[13] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *Proceedings of the European Control Conference*, 2019, pp. 3420–3431.

[14] B. T. Lopez, J.-J. E. Slotine, and J. P. How, "Robust adaptive control barrier functions: An adaptive and data-driven approach to safety," *Control Systems Letters*, vol. 5, no. 3, pp. 1031–1036, 2020.

[15] W. S. Cortez, D. Oetomo, C. Manzie, and P. Choong, "Control barrier functions for mechanical systems: Theory and application to robotic grasping," *IEEE Transactions on Control Systems Technology*, 2019.

[16] M. Srinivasan and S. Coogan, "Control of mobile robots using barrier functions under temporal logic specifications," *IEEE Transactions on Robotics*, 2020.

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

[18] A. Lazaric and M. Ghavamzadeh, "Bayesian multi-task reinforcement learning," in *Proceedings of the International Conference on Machine Learning*, 2010, pp. 599–606.

[19] Q. Nguyen and K. Sreenath, "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints," in *Proceedings of the American Control Conference*, 2016, pp. 322–328.

[20] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *Proceedings of the IEEE Conference on Decision and Control*, 2014, pp. 6271–6278.

[21] H. Zhao, S. Kolathaya, and A. D. Ames, "Quadratic programming and impedance control for transfemoral prosthesis," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2014, pp. 1341–1347.

[22] S.-C. Hsu, X. Xu, and A. D. Ames, "Control barrier function based quadratic programs with application to bipedal robotic walking," in *Proceedings of the American Control Conference*, 2015, pp. 4542–4548.

[23] M. Rauscher, M. Kimmel, and S. Hirche, "Constrained robot control using control barrier functions," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 279–285.

[24] J. P. Kolhe, M. Shaheed, T. Chandar, and S. Talole, "Robust control of robot manipulators based on uncertainty and disturbance estimation," *International Journal of Robust and Nonlinear Control*, vol. 23, no. 1, pp. 104–122, 2013.

[25] A. Sakai, D. Ingram, J. Dinius, K. Chawla, A. Raffin, and A. Paques, "Pythonrobotics: a python code collection of robotics algorithms," *arXiv preprint arXiv:1808.10703*, 2018.

[26] M. W. Spong and M. Vidyasagar, *Robot Dynamics and Control*. John Wiley & Sons, 2008.

[27] K. Katharina, "Force estimation in robotic manipulators: Modeling, simulation and experiments," Master's thesis, Norwegian University of Science and Technology, 2014.