Safe Human-Robot Coetaneousness Through Model Predictive Control Barrier Functions and Motion Distributions∗
Mohammadreza Davoodi∗ Joseph M. Cloud∗∗ Asif Iqbal∗
William J. Beksi∗∗ Nicholas R. Gans∗
∗ University of Texas at Arlington Research Institute, Fort Worth, TX, USA. (mdavoodi@uta.edu, asif.iqbal@uta.edu, nick.gans@uta.edu.)
∗∗ Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, TX, USA.
(joe.cloud@mavs.uta.edu, william.beksi@uta.edu.)

Abstract: Future real-world applications will consist of robots and human workers collaborating with each other in a shared environment to increase productivity. In such scenarios, it is necessary to guarantee the safety of humans while maintaining precise control of the robots performing tasks. Probabilistic movement primitives (ProMPs) are a powerful tool for defining a distribution of trajectories for dynamic systems. However, they have been solely used for determining robot trajectories. In this paper, we utilize ProMPs to predict the probabilistic motion of humans in the environment. To achieve this, we propose a combination of model predictive control (MPC) and control barrier functions (CBFs) to guide a robot along a predefined trajectory while guaranteeing it always maintains a desired distance from a human worker motion distribution defined by a ProMP. A case study is provided to demonstrate the efficacy of our methods.

Copyright © 2021 The Authors. This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0/)

Keywords: Probabilistic Movement Primitives; Control Barrier Functions; Model Predictive Control; Human-Robot Coexistence

1 INTRODUCTION

Cooperation between humans and robots has increased in many applications such as industry and in the home (Kanazawa et al. (2019); Fishman et al. (2020)). In these scenarios, robots may perform repetitive and simple jobs, as well as physically demanding tasks outside the abilities of humans. Currently, human workers are typically responsible for tasks that are challenging for robots, such as situation assessments. To have an efficient and united human-robot system, their advantages must be integrated in a collaborative manner. One of the main challenges for realizing a seamless, combined system is to guarantee the safety of human workers in the environment (Kimmel and Hirche (2017)). Two important aspects for accomplishing safe cooperation are the following: (i) modeling human-motion trajectories in a way that captures humans’ relatively low-levels of repeatability in the environment; (ii) synthesizing an appropriate controller for a robot to avoid collisions with human trajectories.

1.1 Human Motion Prediction

Successful human-robot collaboration requires models of human behavior (e.g., motion). This problem has been investigated in a variety of contexts (Bai et al. (2015); Zhou and Wachs (2018)). Human behavior, even when performing repetitive jobs, does not have a repeatable and deterministic pattern. It rather depends on many factors including fatigue, skill level, and adaptation to various tasks. Consequently, it’s more effective to consider probabilistic models such as Gaussian mixture models (Wiest et al. (2012)) which can model the prediction uncertainty. Moreover, from a control theory point of view, it is important to have methods capable of predicting...
human motion over a future time horizon instead of just delivering the current human dynamical states.

In this work, we use ProMPs for modeling human motion in the environment whereby a distribution of trajectories is learned from multiple demonstrations or observations (Calinon and Lee (2017)). In Paraschos et al. (2018a), 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 movements to a force-torque sensor input was developed in Paraschos et al. (2018b). Unlike Paraschos et al. (2018a,b) we do not design a ProMP controller in this work, instead we utilize ProMPs to model human motion and for defining safety barriers.

1.2 Safe Robot Control Approaches

Real-time safety in safety-critical dynamical systems is a paramount issue (Wieand and Allgöwer (2007)). 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 (Sloth et al. (2012)). The concept of barrier certificates/functions was extended for synthesizing real-time safe control laws for dynamical systems via CBFs (Ames et al. (2016)).

CBFs can provide verifiable control laws with safety guarantees. They integrate seamlessly with control Lyapunov functions (CLFs) to offer stability while respecting limits and safe regions of the state space (Ames et al. (2019)). 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. Efforts to automate the definition of CBFs and CLFs include mapping temporal logic statements with respect to performance requirements (Srinivasan and Coogan (2020)) and fitting piecewise-linear barrier functions to trained obstacles or regions (Saveriano and Lee (2020)).

Although CBFs are effective for safe control design in engineering systems, most existing techniques only assure myopic short-term safety constraints for simple nonlinear dynamical systems. On the other hand, MPC is a well-established method for real-time control of dynamical systems subject to system constraints and has a long history in industrial applications (Qin and Badgwell (2003)). While MPC is a useful tool for addressing various types of system constraints, there exists some safety constraints that cannot be adequately handled. However, MPC and CBFs can be unified together into one optimization problem to address such challenges (Zeng et al. (2020)).

1.3 Contributions

This work investigates the concept of human-robot collaboration from the viewpoint of having an efficient level of safe coexistence (Magrini et al. (2020)). For example, this may occur when a robot and human work close to each other in a shared workspace without requiring mutual contact or the coordination of actions and intentions (De Luca and Flacco (2012)). We present a safe human-robot coexistence system for anticipating the surrounding human workers’ motions based on the assumption that there exists a centralized high-level task assignment and observer (HLTAO) module for assigning tasks to and monitoring all humans and robots in the environment.

Our goal is to design effective control strategies for robots such that they are capable of achieving their assigned tasks while avoiding collisions with other robots and humans. To accomplish this objective, the possible motions of the human workers in the environment are modeled as a set of ProMPs. Using HLTAO, each individual robot can receive the current location of the neighboring robots as well as the position and distribution (represented by ProMPs) of the surrounding humans. These human distributions and robot positions are then used to define two sets of CBFs. We propose an MPC/CBF control method that simultaneously guarantees the control performance and the safety of the system.

2 BACKGROUND

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

Notation: Given a matrix $A$, let $A^\top$ denote its transpose. The number of axes of the workspace is represented by $n$. We denote the sample number for a discrete-time state or parameter by subscript, e.g., $x_k$. Given a vector $x \in \mathbb{R}^n$, we indicate its $j$th element at the $k$th time step with a subscript $x_{j,k}$. $\gamma$ is used as a subscript throughout $\gamma_j$. A trajectory can be generated from the ProMP distribution (or human) to exhibit coordinated behavior. To realize movement, which can be either rhythmic or stroke-based.

In a ProMP, a trajectory is represented as a set of Cartesian positions $\mathbf{q}_i = \{x_{i,k}\}$, where $x_{i,k} \in \mathbb{R}$, axis $j \in \{1, \ldots, n\}$, and $k$ is the time step. 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

$$g_{j,k} = \begin{bmatrix} x_{j,k} \\ x_{j,k}^\top \end{bmatrix} = \Phi_k w_i + \xi_{x_i},$$

where $\Phi_k = [\phi_k \phi_k^\top] \in \mathbb{R}^{2 \times L}$ 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, a 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.,
The distribution \( p(\xi, \theta_i) \) defines a hierarchical Bayesian model over the trajectories \( \xi \) (Paraschos et al. (2018a)), and \( p(w_t \mid \theta_i) = \mathcal{N}(w_t \mid \mu_{w_t}, \Sigma_{w_t}) \). In a movement primitive representation, the parameters of a single primitive must be easy to obtain from demonstrations. The distribution of the state \( p(\xi_{i,k} \mid \theta_i) \) is

\[
p(\xi_{i,k} \mid \theta_i) = \mathcal{N}(\xi_{i,k} \mid \Phi_k \mu_{w_i}, \Phi_k \Sigma_{w_i} \Phi_k^\top + \Sigma_{\theta_i}).
\]

(2)

A 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 (or human) movement, which can be either rhythmic or stroke-based. From (2), the mean \( \mu_{i,k} \in \mathbb{R}^2 \) of the ProMP trajectory at \( k \) is \( \Phi_k \mu_{w_i} \) and the covariance \( \Sigma_{i,k} \in \mathbb{R}^{2\times2} \) is \( \Phi_k \Sigma_{w_i} \Phi_k^\top + \Sigma_{\theta_i} \).

Multiple demonstrations are needed to learn a distribution over \( w_i \). We use a combination of radial basis and polynomial basis functions for training. From the demonstrations, the parameters \( \theta_i \) can be estimated using maximum likelihood estimation (Lazaric and Ghavamzadeh (2010)). However, this may result in unstable estimates of the ProMP parameters when there are insufficient demonstrations. Our method uses a regularization to estimate the ProMP distribution similar to Gomez-Gonzalez et al. (2020). We maximize \( \theta_i \) for the posterior distribution over the ProMP using expectation maximization, \( p(\theta_i \mid x_{i,k}) \propto p(\theta_i) p(x_{i,k} \mid \theta_i) \).

For systems with multiple degrees of freedom, where \( n > 1 \), connecting between axes of the workspace enables the robot (or human) to exhibit coordinated behavior. To realize this with ProMPs, we calculate the covariance of the axes (Paraschos et al. (2018b)). In addition, we use a Normal-Inverse-Wishart as a prior distribution \( p(\theta_i) \) to increase stability when training the ProMP parameters (Gomez-Gonzalez et al. (2020)). Assuming two, single-axis ProMPs \((\mu_{i,k}, \Sigma_{i,k}), i \in \{1, 2\} \), the joint distribution is defined as \( (\mu_k, \Sigma_k) \in \mathbb{R}^{2 \times 2} \). Finally, by removing the rows and columns related to the axis velocities, \( (\mu_k, \Sigma_k) \in \mathbb{R}^{2 \times 2} \) are obtained and used as the 2D model of human motion in the workspace as described in Section 3.

2.2 Control Barrier Functions

Consider the following discrete-time nonlinear system

\[
x_{k+1} = f(x_k, u_k),
\]

(3)

where \( x \in \mathcal{X} \subset \mathbb{R}^n \) denotes the state, \( u \in \mathcal{U} \subset \mathbb{R}^p \) is the control input, and \( f : \mathbb{R}^n \rightarrow \mathbb{R}^n \) is a locally Lipschitz vector field. It is assumed that the system in (3) is controllable.

Assumption 1: All system states \( x_k \) can be measured or estimated at each time step \( k \).

Define a set \( C \) for which we wish to verify that \( x(t) \in C, \forall t \). \( C \) then defines a safe set. A smooth function \( h(x) : \mathbb{R}^n \rightarrow \mathbb{R} \) is defined to encode a constraint on the state \( x \) of the system. The constraint is satisfied if \( h(x) \geq 0 \), and violated if \( h(x) < 0 \). Consider the set \( C \) defined by

\[
C = \{ x \in \mathbb{R}^n : h(x) \geq 0 \},
\]

\[
\partial C = \{ x \in \mathbb{R}^n : h(x) = 0 \},
\]

\[
\text{Int}(C) = \{ x \in \mathbb{R}^n : h(x) > 0 \},
\]

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

Existing approaches to define CBFs include exponential CBFs, zeroing CBFs, and reciprocal CBFs (Nguyen and Sreenath (2016); Ames et al. (2016)). These methods have trade-offs in terms of ease of definition, boundedness of velocities, speed of convergence, etc. In this work we investigate discrete-time exponential CBFs.

Definition 1. Given the set \( C \), the smooth function \( h \) is a CBF if there exists a constant \( \gamma \) such that (Agrawal and Sreenath (2017)),

1. \( h(x_0) > 0 \),
2. There exists a control input \( u_k \) such that

\[
\Delta h(x_k, u_k) = -\gamma h(x_k), \quad 0 < \gamma \leq 1,
\]

(5)

where \( \Delta h(x_k, u_k) = h(x_{k+1}) - h(x_k). \) From (5) it is clear that \( h(x_{k+1}) \geq (1 - \gamma) h(x_k). \) Hence, the barrier function \( h(x_k) \) always has a lower bound \((1 - \gamma)^k h(x_0),\) and is an exponential function in \( k \). Due to this fact, it is referred to as a discrete-time exponential CBF.

2.3 Model Predictive Control

The main goal of MPC is to find a sequence of control actions by solving a finite-horizon optimization problem at each sampling instant. Concretely, the MPC problem is formulated as (Grandia et al. (2020); Zeng et al. (2020))

\[
\min_u \quad V(x_{k+N|k}) + \sum_{t=0}^{N-1} L(x_{k+t|k}, u_{k+t|k})
\]

(6a)

subject to

\[
x_{k+t+1|k} = f(x_{k+t|k}, u_{k+t|k}), \quad t = 0, \ldots, N - 1,
\]

(6b)

\[
x_{k+t|k} \in \mathcal{X}, \quad u_{k+t|k} \in \mathcal{U}, \quad t = 0, \ldots, N - 1,
\]

(6c)

\[
x_{k+N|k} \in \mathcal{X}_f,
\]

(6d)

\[
x_{k|k} = x_k,
\]

(6e)

where \( V(\cdot) \) and \( L(\cdot) \) are, respectively, the terminal and stage costs, and \( N \) is the time horizon. The state and input constraints are given by (6c), and the terminal constraint is enforced in (6d). To solve an MPC problem, at each time step \( k \), (6) is solved and a sequence of control actions \( u^* = [u_{k|k}, \ldots, u_{k+N-1|k}] \) is generated. Then, only the first element of \( u^* \) is applied (as a feedback law) to the system dynamics and the next measured state \( x_{k+1} \) is estimated. Finally, the estimated state is considered as a new initial condition for the next time step and the optimization problem is repeated.

3 MPC/CBF FORMULATION

3.1 Problem Formulation

Consider an environment (denoted by \( \mathcal{Q} \subset \mathbb{R}^n \)), with \( t \) tasks designated by Task\( T \), \( o \in \mathcal{O} = \{1, \ldots, t\} \), \( m \) human workers \( \mathcal{H}^o \), \( h \in \mathcal{H} = \{1, \ldots, m\} \), and \( l \) heterogeneous robots \( \mathcal{R}^r \), \( r \in \mathcal{R} = \{1, \ldots, l\} \) that work in close proximity to humans to complete a set of predefined tasks. Similar to (3), each robot is modeled as a nonlinear dynamical system, \( x_i^{t+1} = f(x_i^t, u_i^t), i \in \mathcal{R} \), with state vector \( x_i \in \mathbb{R}^n \), and control input \( u_i \in \mathbb{R}^p \).

Problem: Our primary objective is to control the robots in real-time such that (i) they avoid collisions with human
workers and other neighboring robots (for safety), and (ii) they can reach to their assigned tasks (for accomplishing the control objectives).

To solve this problem, we utilize a HLTAO module that is capable of continuously monitoring (e.g., using cameras) all robots and humans, estimating their current positions. HLTAO assigns tasks to each robot or human, and it can communicate with all robots in the environment. Moreover, we assume that the humans’ motions in the environment are modeled as a set of ProMPs. Transmission of these models by HLTAO to the robots allows for the accurate prediction of human workers’ future behaviors. Based on these predictions, a set of MPC/CBF controllers are designed for the robots such that control and obstacle avoidance objectives can be accomplished simultaneously in a unified framework. The general structure of our proposed system is shown in Fig. 2.

3.2 Centralized HLTAO Module

The coordinated control of a network of robots is achieved by exchanging data such as sensor measurements or information about intended actions. The HLTAO module collects and transmits the appropriate data between different robots and humans in the environment. We consider two types of dynamic obstacles in $Q$. First, a robot’s neighboring obstacles are represented by $\mathcal{N}_R^1 = \{ j \mid \text{distance}(p_R^1, p_R^j) < \epsilon_{sr}, i \neq j, j \in \mathcal{R}\}$, where $\epsilon_{sr} > 0$ is the sensing range of robot and $p_R^i = (p_R^{i,x}, p_R^{i,y}) \in \mathbb{R}^2$, $i \in \mathcal{R}$, is the current position of robot $i$. Second, a robot’s neighboring human workers, represented by $\mathcal{N}_H^1 = \{ j \mid \text{distance}(p_H^1, p_H^j) < \epsilon_{hr}, j \in \mathcal{H}\}$, where $p_H^i = (p_H^{i,x}, p_H^{i,y}), i \in \mathcal{H}$ is the current position of human $i$. We model small static obstacles as non-moving robots.

We assume the HLTAO module can estimate the current position of the robots, $p_R^i$, and humans, $p_H^i$. Based on the current positions of the humans and robots, the module calculates $\mathcal{N}_H^1$ and $\mathcal{N}_R^1$ for each robot. The human motions are modeled by a set of ProMPs. To do this, we define the finite set $S = \{ \text{Entrance}, \text{Task}^1, \ldots, \text{Task}^3 \}$ as the set of points a human worker may traverse in order to perform a set of tasks in $Q$. Then, a set of ProMPs between each pair of points in $S$ is generated and all of these models are saved by the HLTAO module.

Assumption 2: The human worker moves at an approximately constant speed in the environment.

HLTAO continuously transmits the humans’ current position and the ProMP parameters to the neighboring robots. Robots use this data to predict the human motion distribution and the future movement of the humans in the environment. Our high-level strategy (Fig. 3) for coordinating robots in the environment is summarized as follows.

1. At each time step, HLTAO receives the current position of all the humans and robots in the environment and calculates $\mathcal{N}_H^1$ and $\mathcal{N}_R^1$ for $i$th robot.
2. When a new task (represented by a point in $S$) is assigned to a human in $\mathcal{N}_H^1$. HLTAO selects the corresponding ProMP between the current position of the human and the assigned task.
3. At each time step, HLTAO transmits the current positions of the robots in $\mathcal{N}_R^1$ as well as the positions and ProMPs of the humans in $\mathcal{N}_H^1$ to the $i$th robot.
4. For each human in $\mathcal{N}_H^1$, robot $i$ finds the closest ProMP mean position to the current human position and uses it as the initial ProMP location in its finite-horizon optimization.

3.3 MPC/CBF Control Design

In designing an MPC/CBF controller, each robot needs to estimate its states and the states of neighboring humans and robots in a finite-horizon. Although each robot is capable of predicting future human behavior using the humans’ ProMPs, it only has access to the current location (not future data) of the neighboring robots. Therefore, we assume that among a group of neighboring robots only one robot with the highest index number can move while the other robots are stationary at each time step (i.e., they act as static obstacles for the moving robot). Therefore, the following switching control policy (SCP) is implemented on each robot.

$$\mathcal{R}^i \begin{cases} \text{MPC/CBF controller} & \text{if } i > j, j \in \mathcal{N}_R^1, \\ \text{STOP} & \text{if } i < j, j \in \mathcal{N}_R^1. \end{cases}$$

For the $i$th robot, $i \in \mathcal{R}$, two sets of barrier functions $(h_{1,i}^j, j \in \mathcal{N}_H^1, h_{2,i}^j \in \mathcal{N}_R^1)$ are defined to assure safety in the presence of both neighboring humans and
robots. The Mahalanobis distance, as a measure of the distance between the current robot position \( p_r^k \) and the current distribution of a human \( N_k^j = (\mu_k^j, \Sigma_k^j) \), is used to constrain the minimal distance to the human and avoid collisions. It is calculated as

\[
D_M(p_r^k, \mu_k^j, \Sigma_k^j) = (p_r^k - \mu_k^j)\Sigma_k^{-1}(p_r^k - \mu_k^j)^\top.
\]

The Mahalanobis distance is used to define the CBFs \( h_1^j \)

\[
h_1^j = (p_r^k - \mu_k^j)^\top \Sigma_k^{-1}(p_r^k - \mu_k^j) - \epsilon_h^j, \quad j \in \mathcal{M}_H, (7)
\]

where \( \epsilon_h^j \) acts as a safety factor by specifying a bound for the minimal distance to the humans in the workspace. Thus, \( \epsilon_h^j \) can be chosen by the designer (mission planner) based on the safety requirements of the tasks in the environment. For example, if the job is dangerous (e.g., the robot is carrying a hazardous payload as part of its task) larger values can be considered for \( \epsilon_h^j \) to create more distance between the robot and the humans.

Neighboring robots are modeled as circles with centroids \( p_r^{ij} = (p_r^i, p_r^j) \) and fixed radii \( r_i \). Thus, the CBFs \( h_2^j \) that represent the safety constraint between robot \( i \) and its neighboring robots are defined as

\[
h_2^j = (p_r^i - p_r^{ij})^2 + (p_r^j - p_r^{ij})^2 - r_i^2, \quad j \in \mathcal{M}_R. (8)
\]

Based on the results of Grandia et al. (2020) and Zeng et al. (2020), the MPC/CBF problem is formulated as

\[
\begin{align*}
\min_{u^i} & \quad V^i(x_{k+1}^i) + \sum_{t=0}^{N-1} L^i(x_{k+t+1}^i, u_{k+t+1}^i) \\
\text{subject to} & \quad x_{k+t+1}^i = A^i x_{k+t}^i + B^i u_{k+t}^i + B_r^i, \quad t = 0, \ldots, N - 1, \\
& \quad x_{k+t}^i \in X_i, \quad u_{k+t}^i \in U_i, \quad t = 0, \ldots, N - 1, \\
& \quad x_{k+N}^i \in X_i, \\
& \quad \Delta h_1^j(x_{k+t}^i, u_{k+t}^i, N_{j+k+t}^i) \geq 0, \\
& \quad \Delta h_2^j(x_{k+t}^i, u_{k+t}^i, p_r^{ij}) \geq 0, \\
& \quad \Delta h_3^j(x_{k+t}^i, u_{k+t}^i, p_{r}^{ij}) \geq 0,
\end{align*}
\]

where \( x_{k+1}^i = A^i x_k^i + B^i u_k^i \) is the linear discrete-time counterpart to the system in (3), corresponding to \( i \) th robot system dynamics. The nonlinear distance constraints for the safety criteria are given by conditions (9e) and (9f). One option for the stage cost is \( L^i(x_k^i, u_k^i) = (x_k^i - x_{ref}^i)^\top Q^i (x_k^i - x_{ref}^i) + u_k^i \top R^i u_k^i \). In this work, \( x_{ref}^i = [p_{r}^{Task_1}, p_{r}^{Task_2}, 0, \ldots (n-2)] \) such that the \( i \) th robot will reach its corresponding task, Task \( i \). Note that this type of cost function minimizes the system input similar to \( u_k^i \top H(x_k^i) u_k^i \) in a typical CLF/CBF control design.

In the MPC/CBF control design problem, the terminal cost \( V^i \) and the terminal set \( X^i \subseteq X_i \) can be utilized to guarantee the stability of the system along the closed-loop trajectory. Instead of using the CLF constraints in the conventional CLF/CBF-based QP optimization (Ames et al. (2019)), in our formulation the terminal cost minimizes the CLF.

The following simple procedure can be used to guarantee stability using the terminal cost (Christofides et al. (2013)). We assume a linear stabilizing feedback control law \( u_k^i = K^i x_k^i \) exists for the unconstrained case, i.e., \( A^i + B^i K^i \) is stable. Such a controller can be computed using the solution of an infinite-horizon linear quadratic regulator problem with the same weights \( Q^i \) and \( R^i \) used in the MPC optimization problem. Then, letting \( P^i \) be the solution of the Lyapunov equation

\[
(A^i + B^i K^i)^\top P^i (A^i + B^i K^i) - P^i = -(Q^i + K^i \top R^i K^i),
\]

the terminal cost and set are chosen to be \( V^i(x_N^i) = x_N^i \top P^i x_N^i \) and

\[
X^i_{f} = \{ x^i | x^i \top P^i x^i \leq c \},
\]

where \( c \) is a small positive value chosen such that \( u_k^i = K^i x_k^i \in U^i \) for any \( x^i \in X^i_{f} \). More details on the stability analysis can be found in Borrelli et al. (2017).

**Remark 1.** Based on the considered stage cost in (9), it is guaranteed that each robot safely achieves its final position \( x^{ref}_i \) from its current position. However, it might be practically interesting to have scenarios in which robots need to follow some predefined distributions while avoiding collisions with humans. The proposed method can be modified in such a way that it forces robots to move inside predefined distributions, delivered from pre-trained ProMPs. To this end, the cost function in (9) should behave as a ProMP-based controller for each robot. In this case, the stage cost of the MPC is replaced with

\[
L^i(x_k^i, u_k^i) = (x_k^i - \mu_k^i)^\top \Sigma_k^{-1} (x_k^i - \mu_k^i) + u_k^i \top R_k u_k^i,
\]

where \( \mu_k^i \) and \( \Sigma_k \) come from pre-trained ProMPs for the \( i \) th robot. Moreover, to avoid collisions between robot and human distributions, the Mahalanobis distance should be replaced with the Bhattacharyya distance. The Bhattacharyya distance, \( D_B \), measures the similarity of two probability distributions. For two normal distributions \( N^1 = (\mu^1, \Sigma^1) \) and \( N^2 = (\mu^2, \Sigma^2) \), the distance is defined as

\[
D_B = \frac{1}{8} (\mu^i - \mu^j)^\top \Sigma^{-1} (\mu^i - \mu^j) + \frac{1}{2} \ln \left( \frac{\det \Sigma}{\det \Sigma^i \det \Sigma^j} \right),
\]

where \( \Sigma = \Sigma^i + \Sigma^j / 2 \).

**4 SIMULATION RESULTS**

In this section a proof of concept example is provided to highlight the effectiveness of our methodology for a safe human–robot coexistence system. The system models and proposed real-time controller were simulated using MATLAB 2019a. All computations were done on a Dell OptiPlex 7050 machine with an Intel Core i7-7700X CPU and 8 GB of memory.

Consider an environment \( Q \) with two robots \( (R^1, R^2) \), one human \( (H^1) \), and three tasks \( (\text{Task}_1, \text{Task}_2, \text{Task}_3) \). The positions of the tasks in the environment are, respectively, \((0.3270, 0.2535), (-0.3813, 1.6545), (-0.15, 1.8)\). The initial positions of the robots are \( R^1 = (-0.1, 1.6), R^2 = (0.4, 0.1) \). The human is initially at \( \text{Task}_1 \). The HLTAO module has assigned Task \( 2 \) to the human and Task \( 3 \) to the second robot. It’s assumed that the first robot does not move during the mission and acts as a static obstacle to the second robot. The goal is to control the second robot \( (R^2) \) to move from its initial position to Task \( 3 \) without colliding with the human \( (H^1) \) and the other robot \( (R^1) \).

The human motion between Task \( 1 \) and Task \( 2 \) is modeled as a ProMP. We generated 49 different trajectories that arrive...
in a neighborhood of Task^2 with a fixed starting position. Using this dataset, we trained the ProMP with Algorithm 1 from Gomez-Gonzalez et al. (2020). We used L = 5 basis functions consisting of five radial basis parameters. The results of the ProMP training are presented in Fig. 4 along with the mean of the human demonstration shown in green, and the mean ± variance bounds shown with dotted black lines. We assume the human moves with a constant speed and it takes 8 seconds for the human to reach Task^2 from Task^1.

The robots are modeled as a linear discrete-time 2D double integrator system \( x_{k+1} = A^i x_k + B^i u_k \), where

\[
A^i = \begin{bmatrix}
1 & 0 & dt \\
0 & 1 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0
\end{bmatrix}, \\
B^i = \begin{bmatrix}
0.5dt^2 & 0 \\
0 & 0.5dt^2 \\
0 & dt \\
0 & 0
\end{bmatrix}.
\] 

(10)

The sampling time is set to \( dt = 0.05 \). The weighting parameters of the MPC/CBF controller are selected as \( Q^i = 0.1I_4 \), \( R^i = I_2 \) and \( P^i = 100I_4 \). The CBF parameters \( \gamma_1 \) and \( \gamma_2 \) are initialized to \( \gamma_1 = \gamma_2 = 0.4 \). The fixed radius in the second CBF is set to \( r^i = 0.05 \). Additionally, we assume the robot model (10) is subject to the following state and input constraints

\[
N^i = \{ x_k^i \in \mathbb{R}^n, -5I_{4 \times 1} \leq x_k^i \leq 5I_{4 \times 1} \}; \\
U^i = \{ u_k^i \in \mathbb{R}^n, -0.5I_{2 \times 1} \leq u_k^i \leq 0.5I_{2 \times 1} \}.
\] 

(11) (12)

IPOPT, an open-source software package for large-scale nonlinear optimization (Wächter (2009)), is used as the solver for the proposed optimization problem.

Three different scenarios are considered. In scenario 1, the safety factor \( \epsilon_{th} = 0.1 \) and prediction horizon \( N = 4 \). For scenario 2, \( \epsilon_{th} = 0.85 \) and \( N = 4 \). In scenario 3, \( \epsilon_{th} = 0.1 \) and \( N = 8 \). The results are presented in Figs. 4, 5, 6. By comparing scenario 1 and scenario 2 in Fig. 4, it is clear that increasing the safety factor \( \epsilon_{th} \) increases the distance of the robot to the human distribution. It can be seen that a larger horizon \( N \) (scenario 3) causes the robot to avoid obstacles (human \( H^1 \) and robot \( R^2 \)) earlier. Specifically, among all scenarios, the robot trajectory in scenario 3 is farthest from the obstacles. Moreover, the designed controller in scenario 3 is faster than the controllers in the other scenarios and the system starts to avoid the obstacle earlier. This indicates that the robot is operating within a smaller safe set, i.e., the system tends to be safer. Furthermore, we can see that with a larger horizon \( N \), the system has noticeable obstacle avoidance behavior when it is near obstacles. As can be seen from Fig. 6, in scenario 3 robot \( R^2 \) reaches the static obstacle (robot \( R^1 \)) after approximately 4.15 seconds (note that \( h_3 \approx 0 \)), which is sooner than the other scenarios, with \( h_3 \approx 0 \) at about 9 seconds. Consequently, having a larger horizon time (scenario 3) is more efficient for cases where the robot should reach its final task (goal position) in a limited amount of time.

In the simulations, the main computational cost with respect to the time of our controller comes from the fact that it has to solve a nonlinear optimization problem using IPOPT at every time step. To visualize the effect of the prediction horizon on the computational complexity of our proposed method, we compared the computational times in scenario 2 (Sc2) and scenario 3 (Sc3). The average required time (\( T_{ave} \)), maximum time (\( T_{max} \)), and the standard deviation (std) for solving the optimization problem of these scenarios are as follows: \( T_{ave}^{Sc2} = 0.0107 \), \( T_{ave}^{Sc2} = 0.0071 \), std = 0.0036, \( T_{ave}^{Sc3} = 0.0112 \), \( T_{max}^{Sc3} = 0.1571 \), std = 0.0074, where the units are seconds. Based on these results, it can be concluded that a larger prediction horizon in the proposed MPC/CBF controller leads to higher computational time. Furthermore, it is clear that the expected execution time of the optimization problems is very small (in the range of 10 ms). The large maximum times are each a single outlier. Hence, the controller is appropriate for a real-time implementation.

5 CONCLUSIONS

In this paper, we investigated the design of a distributed, real-time controller for safe human-robot coexistence. We assumed there exists a HLTAO module capable of monitoring and assigning tasks to all humans and robots in the environment. Furthermore, the extent of possible human motions as they move from one task to another were mod-
Fig. 6. The second CBF represents the safety constraint between the robot ($R^2$) and neighboring robot ($R^1$). Note that the minimal distance between robots never crosses zero thus assuring collision avoidance.

eled as a set of ProMPs. By transmitting these ProMPs and the current location of the humans to the robots through HLTAO, an MPC/CBF controller was designed for each robot. The controller can guide the robots to reach their assigned tasks and simultaneously avoid collisions with other robots and humans in the workspace.

REFERENCES


