Reactive and Safe Co-Navigation with Haptic Guidance

Mela Coffey*, Dawei Zhang*, Roberto Tron, and Alyssa Pierson

***Abstract*— We propose a co-navigation algorithm that enables a human and a robot to work together to navigate to a common goal. In this system, the human is responsible for making high-level steering decisions, and the robot, in turn, provides haptic feedback for collision avoidance and path suggestions while reacting to changes in the environment. Our algorithm uses optimized Rapidly-exploring Random Trees (RRT*) to generate paths to lead the user to the goal, via an attractive force feedback computed using a Control Lyapunov Function (CLF). We simultaneously ensure collision avoidance where necessary using a Control Barrier Function (CBF). We demonstrate our approach using simulations with a virtual pilot, and hardware experiments with a human pilot. Our results show that combining RRT* and CBFs is a promising tool for enabling collaborative human-robot navigation.**

I. INTRODUCTION

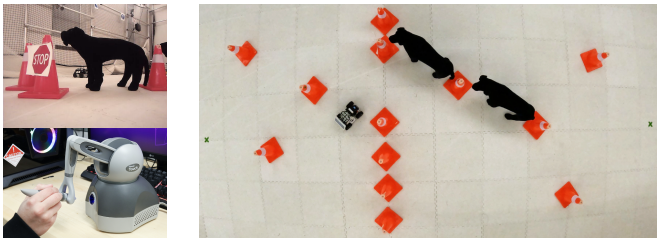
Navigating vehicles through complex, cluttered environments can be extremely difficult, especially when operating large vehicles such as powered wheelchairs. Fully autonomous vehicles are not yet sophisticated enough to be able to navigate our world while maintaining their own safety and the safety of humans. Thus, humans are still necessary for the operation of such vehicles, although robots can bring their own strengths to human-robot systems. While much of existing research explores fully autonomous systems or collision avoidance systems, we propose a collaborative human-robot system, where the human and the robot co-navigate to a common goal. In such a system, the human is ultimately responsible for making decisions about where to steer the robot. However, the robot uses its own information about its environment to provide collision avoidance and path suggestions to the human via haptic feedback. This approach takes advantage of both human and robot autonomy.

We envision two scenarios where the proposed approach could be useful. First, in the operation of personal mobility devices, such as powered wheelchairs, which can be especially challenging for novice users. Wheelchair users encounter cluttered environments (such as restaurants and other public places) every day, and avoiding obstacles can be difficult with a relatively large vehicle. A co-navigation scheme with a haptic joystick and sensors incorporated into powered wheelchairs could significantly ease operation: onboard sensors aid in obstacle detection, and high-level path planners can provide suggestions toward the user's goal.

*Denotes equal contribution.

All authors are with the Department of Mechanical Engineering, Boston University, 110 Cummington Mall, MA 02215, United States [mcoffey, dwzhang, tron, pierson] @bu.edu.

This work supported in part by a Boston University startup grant and the National Science Foundation award FRR-2212051. Their support is gratefully acknowledged.



(a) Operator view. (b) Overhead view.

Fig. 1: Hardware experimental setup with a human user teleoperating a ground robot through a cluttered environment (b), with a first-person view (top of (a)) streamed from the robot to the computer screen. The human operator uses a Geomagic Touch haptic device (bottom of (a)) to control the robot, and force feedback is provided to the human for navigation assistance and collision avoidance.

The second scenario is in the teleoperation of a robot through a remote environment. In situations like these, the human pilot typically has a limited field of view of the environment (see Figure 1a for an example). The limited peripheral vision can easily result in collisions with unforeseen obstacles. Moreover, in repetitive environments (such as a pine forest) the pilot might get disoriented and lose sight of the direction to follow toward their goal. In this case, the robot can use sensor data to create an haptic force feedback that helps avoid collisions and guides the user.

Related Work

Previous related work includes both collaborative teleoperation and safety-critical robot motion planning. In the context of collaborative teleoperation, haptic guidance has been proved as a promising approach to improve robot safety and intention transparency by providing extra feedback to the human operator. Haptic feedback has a variety of applications in robotic manipulation [1]–[3], autonomous vehicles [4]–[7], and powered wheelchairs [8], [9].

The use of haptic feedback in a teleoperation settings has traditionally been used as a collision avoidance mechanism, where the result is a repulsive force field pointing away from obstacles. This has been studied in ground vehicles such as wheelchairs [8], [10], where the repulsive force is scaled based on the distance from obstacles. To improve the safety in the teleoperation of unmanned aerial vehicles (UAVs), researchers have proposed Parametric Risk Fields (PRFs) [4] and Dynamic Kinesthetic Boundaries (DKBs) [5]. More recently, Control Barrier Functions (CBFs) have gained significant attention as an effective approach for generating safe control policies that achieve collision avoidance in robotic systems. The authors' prior work [7], [11] uses CBFs to

generate force feedback that helps the user to issue a command that is safe but as close as possible to their intentions. The results in [7], [11] show that collision avoidance through CBFs significantly improves the user's experience [12].

Closer to this paper, some works have proposed a more collaborative navigation paradigm, in which robots guide the human user to a target while providing collision avoidance. In [9], a haptic feedback algorithm designed for wheelchairs switches between a collision avoidance mode and a mode that suggests short, circular paths to the user. The authors of [13] propose haptic feedback for the teleoperation of UAVs in which a DKB is used for collision avoidance and RRT* for route suggestions. The authors' prior work [14] proposes a similar control scheme for non-holonomic vehicles with sample-based guarantees of collision-free navigation.

Paper contributions

A true co-navigation system enables the human and robot to navigate to a common goal while ensuring the safety of the vehicle. In this paper, we combine the benefits of RRT* for route suggestions [14] with those of CBFs for collision avoidance [7]. The former guides the user (and ultimately the robot) toward the common goal, while the latter ensures collision avoidance for dynamic or unexpected obstacles not captured by the planner. The main features and contributions of our paper are the following:

- We incorporate the sample tree from RRT* into a Control Lyapunov Function (CLF) formulation to create haptic suggestions toward kinematically feasible paths. With respect to previous work that uses CBFs for collision avoidance alone, in our method the user can allow the robot to autonomously move toward the goal.
- Standard planners such as RRT* approximate a (typically static) free configuration space with a finite number of samples. As such, obstacles that are small, highly dynamic, or unknown at planning time may cause collisions when the plan is executed. The use of CBFs provides cues that help the user avoid collisions without the need to resample the RRT* tree.
- The user maintains ultimate control authority and may stop the robot or change to a different path as necessary. At the same time, the user can also voluntarily relinquish control and allow the robot to navigate autonomously by simply releasing the haptic interface.

In the remainder of this paper, we first provide preliminary technical details related to CLFs, CBFs, and RRT* (Section II). We then present the details of our proposed solution for haptic feedback generation (Section III), together with results in both simulations and hardware experiments (Section IV).

II. Preliminaries and Background

In this section, we provide the technical background necessary for this paper. We briefly discuss the theory of Control Lyapunov Functions (CLFs) and Control Barrier Functions (CBFs), and then review the non-holonomic RRT* algorithm used for path suggestions. These are used in our haptic feedback formulation and algorithm in Section III.

A. Notation

We first define the notation used in this paper. The set of real numbers in n dimensions is denoted by $\mathbb{R}^n$. The Lie derivative of a smooth function $h(x)$ along a field $f(x)$ is defined as $L_f h(x) \doteq \frac{\partial h(x)}{\partial x} f(x)$, where x is the robot state. We denote the final goal of the human-robot system as x_{f}. We let the force feedback be denoted by F, and the user steering input by u_{ref}.

B. CLFs and CBFs

CLFs and CBFs have initially been applied to adaptive cruise control [15], [16], but they have been widely applied to various applications where safe control is necessary [17]–[19]. In particular, CBFs can be used to implement a supervisory controller that modifies the user's command in the teleoperation of UAVs [12], [20]. Here we provide a brief technical background on these formulations.

We consider a nonlinear affine control system of the form

$$\dot{x} = f(x) + g(x)u, \tag{1}$$

with $x \in \mathbb{R}^n$, $u \in \mathcal{U} \subseteq \mathbb{R}^m$, $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n \times m}$ locally Lipschitz.

A continuously differentiable function $V : \mathbb{R}^n \to \mathbb{R}$ is a globally and exponentially stabilizing CLF for system (1) if there exist constants $c_1 > 0$, $c_2 > 0$ and $p > 0$ such that

$$\begin{gathered} c_1\|x\|^2 \leq V(x) \leq c_2\|x\|^2, \\ \inf_{u \in U}[L_f V(x) + L_g V(x)u + pV(x) \leq 0], \forall x \in \mathbb{R}^n. \end{gathered} \tag{2}$$

Given a CLF $V(x)$, any Lipschitz controller $u \in \mathcal{U}$ satisfying

$$L_f V(x) + L_g V(x)u + pV(x) \leq 0,$$

exponentially stabilizes the system (1) to the origin [15]. In Section III we use a CLF that is tied to the tree produced by RRT*, and is used to pull the user toward the suggested path.

We implement collision avoidance using CBFs. We define a safe set as the zero superlevel set of a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$ as:

$$\mathcal{C} := \{x \in \mathbb{R}^n : h(x) \geq 0\}.$$

The set $\mathcal{C}$ is *forward invariant* for system (1) if every solution starting from any $x(t_0) \in \mathcal{C}$ satisfies $x(t) \in \mathcal{C}$ for all $t \geq t_0$. Given a safe set $\mathcal{C}$, $h(x)$ is a CBF if there exists a class $\mathcal{K}$ function α such that

$$\inf_{u \in U}[L_f h(x) + L_g h(x) + \alpha(h(x))u \geq 0], \forall x \in \mathcal{C}.$$

Given a CBF $h(x)$ with zero level set $\mathcal{C}$, any Lipschitz controller $u \in \mathcal{U}$ that satisfies

$$L_f h(x) + L_g h(x)u + \alpha(h(x)) \geq 0,$$

makes $\mathcal{C}$ forward invariant for system (1) (i.e., safe) [19].

In practice, CLFs and CBFs work well for controllers based on local information: for instance, in adaptive cruise control [19], a CLF pulls the vehicle toward a desired speed, while a CBF maintains a minimum distance with respect to the preceding vehicle. However, when applied as a global

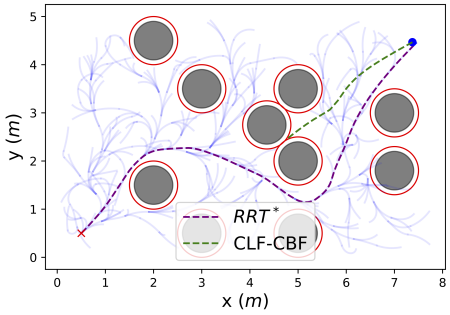


Fig. 2: Comparing CLF-CBF and RRT* as global path planners. The blue dot • represents the starting position, and the red cross × represents the goal x_{f}. Solid blue lines represent the RRT* tree, and dashed lines indicate the robot trajectory under the different planners. Under CLF-CBF the robot gets stuck in a local minimum. Under RRT*, the robot is able to reach the goal.

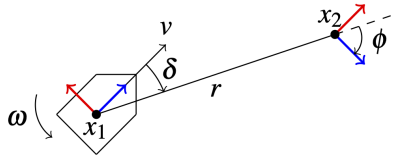


Fig. 3: Egocentric polar coordinates used in the non-holonomic RRT* planner. Here, x_1 is the robot position, and x_2 is an arbitrary goal location; v and ω are the linear and angular velocity of the robot; r and δ are the radial and angular components of the polar coordinates of x_2 with respect to x_1; ϕ is the angle between the line passing through x_1, x_2 and the robot heading at x_2.

planner, e.g., to navigate to a final goal x_{f} trough a cluttered environment, the CLF-CBF method can cause the robot to get trapped in local minima; Figure 2 shows an example of such behavior. This issue motivates the use of RRT* as a global planner, which avoids local minima through sampling; see Figure 2 for the comparison.

C. Non-holonomic RRT*

Sampling-based approaches such as RRT and RRT* have been widely explored and proved to be effective strategies for global path planning [21]–[23]. In particular, in this paper we use the non-holonomic RRT* algorithm from [23] to generate the suggested paths for the user to follow toward the goal location x_{f}. A non-holonomic planner not only eases the teleoperation of ground vehicles by generating smooth paths, but we also anticipate that it will be more intuitive for a human pilot, since prior studies have shown that humans can be modeled as non-holonomic vehicles [24].

The authors of [23] use egocentric polar coordinates (Figure 3) to define a distance between two sampled points x_1 and x_2:

$$\text{Dist}(x_1, x_2) = \sqrt{r^2 + k_\phi^2 \phi^2} + k_\delta |\delta - \delta^*|, \tag{3}$$

where $r > 0$, ϕ, and δ are scalars as in Figure 3, $r, \phi, \delta \in \mathbb{R}$, δ^* is a steering function denoting the target angle at x_2,

$$\delta^*(\phi) = \arctan\left(-k_\phi \phi\right), \tag{4}$$

and k_ϕ and k_δ are positive constants (we set to $k_\phi = 1.2$ and $k_\delta = 3.0$ in all our simulations and experiments). Equation (3) represents the approximate cost for extending the path from node x_1 to node x_2. We use $\mathcal{T}$ and $\mathcal{P}$ to denote, respectively, the tree produced by RRT* and the suggested path.

RRT* is asymptotically complete and optimal; however, in practice, with a finite number of samples it might miss obstacles that are too small, or are not represented well in the map used for planning. Note that RRT* was integrated with CBFs in [25]; while that solution alleviates the problems caused by a finite number of samples, it makes the approach computationally much more intensive. We present a different way to use CBFs and design a force feedback that guides the user to follow the RRT* path while avoiding collisions in Section III.

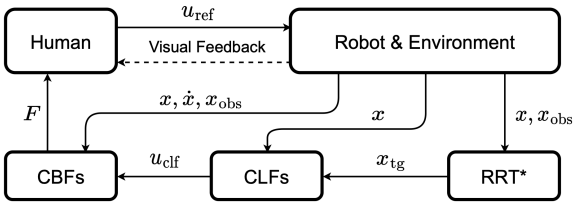


Fig. 4: A flow chart of the proposed haptic guidance and collision avoidance architecture.

III. Haptic Guidance and Collision Avoidance

In this work, we introduce a haptic feedback algorithm that provides a combination of collision avoidance and guidance along a path to the goal of the user. For guidance, we propose a CLF controller centered on a temporary goal x_{tg} that is dynamically computed along the path $\mathcal{P}$ suggested by RRT*. For collision avoidance, we propose a CBF controller (as in [7]) that accounts for obstacles with position x_{obs}. RRT*, CLF, and CBFs are then used to compute a control u_{cbf} from the human input u_{ref}. The difference between u_{cbf} and u_{ref} is used to compute the haptic force F. Figure 4 shows the overall architecture of our system. .

A. CLF-CBF formulation

We use a two-step formulation to generate the control input u_{cbf}. First, we design the CLF $V(x)$ as the Euclidean distance from the robot to the temporary goal x_{tg} on the path $\mathcal{P}$ from RRT* (see also Section III-D for details on the computation of x_{tg}):

$$V(x) = \|x - x_{\text{tg}}\|^2. \tag{5}$$

Applying the CLF theory reviewed above, we compute the minimum-norm control input u_{clf} that satisfies the CLF constraint (2) by solving (in closed form) the Quadratic Program

$$\begin{aligned} u_{\text{clf}} = &\underset{u \in \mathcal{U}}{\text{argmin}} \frac{1}{2}\|u\|^2 \\ &\text{subject to } L_f V(x) + L_g V(x) u + pV(x) \leq 0. \end{aligned} \tag{6}$$

Given (5) and (6), the generated u_{clf} points toward the temporary goal on the suggested path, but might also point toward obstacles.

As second step, we define one CBF $h_i(x)$ for each obstacle i as the Euclidean distance between the robot and the obstacle:

$$h_i(x) = (x - x_{\text{obs,i}})^2 - r_{\text{obs,i}}^2 - d_{\text{safe}}, \tag{7}$$

where $x_{\text{obs,i}} \in \mathbb{R}^2$ is the position of the i-th obstacle, $r_{\text{obs,i}} \in \mathbb{R}$ denotes the radius of the obstacle, and $d_{\text{safe}} \in \mathbb{R}$ is a user-defined safe distance, which can act as a buffer to account for the size of the robot. Given the path-following input u_{clf}, we can compute the safe control input u_{cbf} by

$$\begin{aligned} u_{\text{cbf}} = &\underset{u \in \mathcal{U}}{\operatorname{argmin}} \frac{1}{2}\|u - u_{\text{clf}}\|^2 \\ &\text{subject to } L_f h(x) + L_g h(x) + \alpha(h(x)) \geq 0. \end{aligned} \tag{8}$$

This input ensures that, despite the RRT* path potentially containing collisions, the suggested force feedback will maintain the safety of the robot.

B. Force Feedback Design

As already mentioned, we design the haptic force feedback F as the difference between the safe control input u_{cbf} and the user steering command u_{ref}:

$$F = k_{\text{f}}(u_{\text{cbf}} - u_{\text{ref}}), \tag{9}$$

where $k_{\text{f}} \in \mathbb{R}$ is a positive constant parameter to adjust the magnitude of the feedback. This force feedback reflects the disagreement between the user and the controller. If the user chooses to follow the guidance from the force feedback, the disagreement can be minimized.

Recall that we grant the user full control over the robot. We therefore let the dynamics of the robot be

$$\dot{x} = u_{\text{ref}}.$$

In order to evaluate the efficacy of the force feedback (9), we assume that the pilot can follow the force feedback F while teleoperating the robot.

C. Virtual agreeable pilot

For the purpose of evaluating our approach in simulation (Section IV), we introduce the notion of an *agreeable virtual pilot* who follows the suggested commands through the dynamics

$$\dot{u}_{\text{ref}} = F. \tag{10}$$

Note that we cannot map u_{ref} directly to F, since the latter is computed as a function of the former; instead, F is used to gradually change u_{ref}. We the dynamic (10) in Lemma 1 to show that, assuming the user can follow the force feedback F as in (10), the robot maintains a safe state. Before we can make this claim, we must assume that the constant k_{f} is large enough, such that the feedback F updates faster than or equal to the robot speed, formalized in Assumption 1.

Assumption 1: The force feedback gain k_{f} is large enough that the u_{ref} changes much faster than the robot speed $\dot{x}$.

Lemma 1: Given Assumption 1, let the user be an agreeable human, following dynamics (10). Then, the user input u_{ref} tends to converge to the control suggested by the force feedback F, keeping the robot in a safe state.

Proof: Let the user input evolve according to the dynamics (10), following the force feedback F. Note that (10) has an equilibrium point for $F = 0$, which, from (9), implies that $u_{\text{ref}} = u_{\text{cbf}}$. From Assumption 1, u_{ref} will change much faster than u_{cbf}. Thus, the virtual pilot tends to converge to the control suggested by the autonomy controller. Assuming that the CBF optimization problem (8) is feasible, then (8) will output a safe control input u_{cbf} that will ensure that $h(x) \geq 0$ [19]. In other words, u_{cbf} will ensure that the robot does not collide with an obstacle. ■

Lemma 1 shows that our force feedback (9) used by an agreeable human pilot will keep the robot in a safe state. In the next section, we show that the feedback is not only safe, but it also steers the user on the path to the goal.

While this section introduced a virtual pilot for simulation purposes, in reality, and in our human pilot demonstrations (Section IV-B), the human acts autonomously and can override the force if they wish, maintaining complete control authority.

D. Haptic Feedback Algorithm

Having defined the computation of the force feedback F, we now discuss our complete method to provide the force feedback to the user. Our overall algorithm is outlined in Algorithm 1 and visualized in Figure 4. We first initialize the robot position x, final user goal x_{f}, and obstacle positions and sizes. We then compute and output the force feedback until the robot reaches the goal. At each time step, we retrieve the user input u_{ref} and robot position x. We update the temporary goal x_{tg} using Algorithm 2; the temporary goal, used by the CLF controller, is defined as a point along the path ahead of the robot. We compute the CLF using (5), and then apply (6) to compute the corresponding control input u_{clf}. Since the RRT* path $\mathcal{P}$ is not guaranteed to be collision-free due to finite sampling or dynamic obstacles, we use (8) to compute the safe input u_{cbf} that will be encoded in the force feedback. Before sending the force feedback to the user, we ensure that if the user steering command is zero, such as when the user brakes, then we do not provide any force feedback. Otherwise, output the force feedback (9). This process is repeated until the robot position x is within a distance ϵ from the final goal x_f. We now describe Algorithms 2 and 3 in detail.

1) getTempGoal: In order to compute the temporary goal, we use Algorithm 2. Prior to starting the haptic feedback generation in Algorithm 1, we initialize the RRT* tree $\mathcal{T}$ and optimal path $\mathcal{P}$. The tree is generated from the final goal x_{f} to the initial position $x(t_0)$ to ensure that any path the user decides to take will lead to the goal. Since we use a unicycle robot model, the robot can track the non-holonomic RRT* path in either direction. The initial tree $\mathcal{T}$ is saved and utilized for the entirety of Algorithm 1, i.e. until the robot reaches the goal x_{f}.

We also initialize the positive constant $d_{\max}$, which is a parameter we use to determine whether the human rejects the suggested path $\mathcal{P}$. For example, if the human can see obstacles that the robot cannot detect, then the user may decide to take a different path to the goal. We define $d_{\max}$ as the maximum permissible distance between the robot and

the path $\mathcal{P}$. If the user steers the robot away from the path, and the distance between the robot and path is greater than $d_{\max}$, then we find a new suggested path in the tree $\mathcal{T}$.

We also initialize the constant integer $c > 0$, $c \in \mathbb{R}$, which is a dynamic index used to define the temporary goal. In Algorithm 2, we define the temporary goal as the nearest point on the discretized path $\mathcal{P}$ to the robot with plus c points of lookahead. For instance, let x_{near} be the point on $\mathcal{P}$ closest to the robot. If $c = 50$, then x_{tg} will be 50 points ahead of x_{near} toward the goal x_{f} on $\mathcal{P}$.

We can now summarize the process of determining the temporary goal (Algorithm 2) – required for the computation of u_{clf} in Algorithm 1 – as follows. We first compute the distance from the robot to the current suggested path $\mathcal{P}$. If the user rejects the suggested path, then we redefine $\mathcal{P}$ using Algorithm 3. We then find the nearest point $\mathcal{P}_{i_{\text{near}}}$ with respect to the robot position. Then we define the temporary goal as c points ahead of $\mathcal{P}_{i_{\text{near}}}$. If $i_{\text{near}} + c$ exceeds the length of the path, i.e. $\mathcal{P}_{i_{\text{near}}}$ does not exist, then we set the temporary goal as the last element in $\mathcal{P}$, which is the goal x_{f}.

Under Algorithm 2, the input u_{clf} is guaranteed to drive the robot to the final goal x_{f}, as formalized in Proposition 1 below. We first have to assume, however, the following:

Assumption 2: An optimal collision-free path $\mathcal{P}$ exists from the robot position x to the final goal x_{f}. Furthermore, any path in the tree $\mathcal{T}$ is a path to the goal x_{f}.

Proposition 1: Given Assumptions 1 and 2, the control u_{clf} drives the robot from any point x to the final goal x_{f}.

Proof: Recall that the temporary goal x_{tg} is required to compute the CLF input. Specifically, x_{tg} is used in the CLF (6) such that the input u_{clf} will drive the robot to the temporary goal, i.e. $x \to x_{\text{tg}}$ as $t \to \infty$. Note that, under Algorithm 1, x_{tg} changes at each time step, but x_{tg} is always on the path. Therefore, u_{clf} will always drive x to a point on the path $\mathcal{P}$ and in the direction of the final goal x_{f}. Note also that as the robot nears the end of the path, the temporary goal x_{tg} coincides with the goal location x_{f} (Algorithm 2 Lines 7-9). Therefore, the control input u_{clf} will always guide the user to the goal, i.e. $x \to x_{\text{f}}$ as $t \to \infty$. ■

Remark 1: By (8), the input u_{cbf} is guaranteed to be safe, while being as close as possible to u_{clf}. Therefore, by Lemma 1, an agreeable pilot under the force feedback (9) will execute a safe control input that will guide the user to the final goal x_{f}.

2) getNewPath: We discuss Algorithm 3, which is required by Algorithm 2 to find a new path for the user to follow to the final goal x_{f} when the user rejects the current path. The function takes the robot position, the distance to the current path $\mathcal{P}$, and the saved tree $\mathcal{T}$ as inputs. The algorithm first loops through each node in the tree and gets the non-holonomic path [23] from that node to its parent node, if it has one. The purpose of this for loop is to get the node $i_{\text{near}} \in \mathcal{T}$ corresponding to the nearest branch in $\mathcal{T}$. Once we have i_{near}, we generate the path from this node all the way back to the goal, which is the purpose of the while loop. We can then return this new path $\mathcal{P}$ for use in Algorithm 2. If there are no paths in $\mathcal{T}$ that are closer to the robot than d_{curr}, then the suggested path does not change.

Algorithm 1 Haptic Feedback Generation

Initialize: $x(t_0)$, x_{f}, obstacles
while $\|x - x_{\text{f}}\| \geq \epsilon$ **do**
  Input: user command $u_{\text{ref}}(t)$ and robot position $x(t)$
  Update goal: $x_{\text{tg}} \leftarrow \text{getTempGoal}(x)$
  Define CLF: $V(x) \leftarrow \|x - x_{\text{tg}}\|^2$
  Compute input to path: $u_{\text{clf}}(x, V) \leftarrow$(6)
  Compute safe input $u_{\text{cbf}}(x, u_{\text{clf}}) \leftarrow$(8)
  if $u_{\text{ref}} = 0$ **then**
    No force necessary: $F \leftarrow 0$
  else
    Compute force: $F \leftarrow k_{\text{f}}(u_{\text{cbf}} - u_{\text{ref}})$
  end if
  Output: F
end while

Algorithm 2 $x_{\text{tg}} \leftarrow \text{getTempGoal}(x)$

Initialize: $d_{\max}$, c, and tree $\mathcal{T}$ and optimal path $\mathcal{P}$ from planner [23]
Compute distance to path: $d_{\text{path}} \leftarrow \min \|x - \mathcal{P}\|$
if $d_{\text{path}} > d_{\max}$ **then**
  $\mathcal{P} \leftarrow \text{getNewPath}(x, d_{\text{path}}, \mathcal{T})$
end if
Get the nearest node on $\mathcal{P}$: $i_{\text{near}} \leftarrow \text{argmin} \|x - \mathcal{P}\|$
if $i_{\text{near}} + c >$ length of $\mathcal{P}$ **then**
  return $t_{\text{tg}} \leftarrow x_{\text{f}}$
end if
return $t_{\text{tg}} \leftarrow \mathcal{P}(i_{\text{near}} + c)$

Algorithm 3 $\mathcal{P} \leftarrow \text{getNewPath}(x, d_{\text{curr}}, \mathcal{T})$

Initialize nearest node index $i_{\text{near}} \leftarrow$ None
for each node j in $\mathcal{T}$ **do**
  if j has a parent node in $\mathcal{T}$ **then**
    Get path $\mathcal{P}_{jk}$ from j to parent node k
    Compute distance to the path: $d \leftarrow \min \|x - \mathcal{P}_{jk}\|$
    if $d \leq d_{\text{curr}}$ **then**
      Update smallest distance: $d_{\text{curr}} \leftarrow d$
      Update nearest node: $i_{\text{near}} \leftarrow j$
    end if
  end if
end for
if i_{near} does not exist **then**
  return current path $\mathcal{P}$
end if
Initialize new path: $\mathcal{P} \leftarrow \mathcal{T}(i_{\text{near}})$
Initialize node in path $j \leftarrow i_{\text{near}}$
while node j has a parent k **do**
  Get path $\mathcal{P}_{jk}$ from node j to k
  Append path $\mathcal{P}_{jk}$ to $\mathcal{P}$
  Set the current node as the parent node: $j \leftarrow k$
end while
return new path $\mathcal{P}$

IV. Experimental Validation

Here, we present simulations and hardware experiments to validate our proposed approach. For the simulations, we implement an agreeable virtual pilot to demonstrate the effectiveness of CBFs to achieve collision avoidance when encountering local dynamic obstacles. In our hardware experiments, we implement our haptic feedback with a human pilot, demonstrating the human's ability to follow a path under the force feedback. We also show a human-robot disagreement scenario in which the human must navigate around obstacles that are unknown to the robot.

The optimization problems in (6) and (8) are solved using Gurobi[1] through its Python bindings. We implement the non-holonomic RRT* described in Section II-C as a global planner to get the tree and an initial optimal path. For the implementation of CBF, we define one barrier function $h_i(x)$ for each obstacle i as described in Equation (7). In the CBF constraint, we define the class $\mathcal{K}$ function in (8) as a linear function: $\alpha(x) = qx$, where $q \in \mathbb{R}$ is a constant parameter.

A. Simulations

Here we present simulations with a virtual pilot, which acts as a proxy for the human user. We assume that the pilot can sufficiently follow the path under the guidance of the force feedback in Algorithm 1. We therefore implement the virtual pilot dynamics (10). We initialize the virtual pilot's steering command as $u_{\text{ref}} = 0$, and then adjust the command according (10), allowing the robot to follow dynamics (10).

We conducted 50 trials of different randomized environments with nine static obstacles and one dynamic obstacle. The dynamic obstacle has a periodic motion given by the dynamics

$$\dot{x}_{\text{obs}} = v_{\text{dyn}} \sin \psi t,$$

where $v_{\text{dyn}} \in \mathbb{R}^2$ is a predefined constant velocity and $\psi \in \mathbb{R}$ adjusts the period. During each trial, we randomly generated the start and the final positions on opposite sides of the environment while ensuring they have no collisions with the obstacles. We also randomized the positions (initial position for the dynamic obstacle) of all obstacles to be non-overlapping. At the start of each trial, we generate a new tree from x_{f} to $x(t_0)$ before beginning Algorithm 1, with u_{ref} computed from the virtual pilot (10). We considered two cases in simulation as follows.

1) RRT* *with CLF only:* As a baseline comparison, we consider the case without the CBF. Here, we use the CLF to guide the user along the path generated by RRT* to get u_{clf} as in (6), but we do not compute u_{cbf}. Thus, the force feedback is ultimately

$$F = k_{\text{f}}(u_{\text{clf}} - u_{\text{ref}}). \tag{11}$$

Note that this formulation differs from our proposed feedback (9): here, we use u_{clf} to compute F, whereas in (9), we use u_{cbf}. This case allows us to observe the efficacy of our algorithm without the extra collision avoidance invoked by the CBF.

Figure 5 shows the results of one of our virtual pilot simulations. As we can see in Figure 5a, RRT* generates a collision-free path for the initial static environment. As the robot follows the suggested path, a local dynamic obstacle (represented by the orange dot in the figure) oscillates near the path. Figure 5b shows that a collision happens between the robot and the dynamic obstacle under the feedback (11), i.e. when using only RRT* without CBF.

Of the 50 trials we conducted in this case study, we observed only 33 successful trials, with the other 17 trials containing collisions where the robot was unable to reach the goal. These results indicate that the feedback (11) cannot account for all obstacles, and thus we require additional collision avoidance for obstacles not accounted for by RRT*.

2) RRT* *with CLF and CBF (our proposed approach):* For these simulations, the virtual pilot computes u_{cbf} to ensure collision avoidance (Algorithm 1). Hence, we implement (10) with force feedback (9). Figure 5c shows that, under the same initial conditions as that presented in Figure 5b, the robot can successfully avoid the local dynamic obstacle. Additionally, of the 50 randomized simulations, we observed no collisions under our approach. These results demonstrate the ability of our approach to not only guide the user to a common goal using a CLF with RRT*, but also guarantee collision avoidance using a CBF.

One may notice that in both 5b and 5c, the robot path deviates slightly from the RRT* path, even with an agreeable pilot (10). This feature is a result of the temporary goal used to compute u_{clf}. In particular, the variable c, which defines the lookahead point along the path (Algorithm 2), causes the temporary goal to be slightly in front of the robot and along the path. The force feedback will therefore not attract the robot to travel directly along the path. In fact, for abrupt curves in the RRT* path, the robot will appear to cut corners, since the robot will be attracted to points after the curve.

B. Hardware Experiments

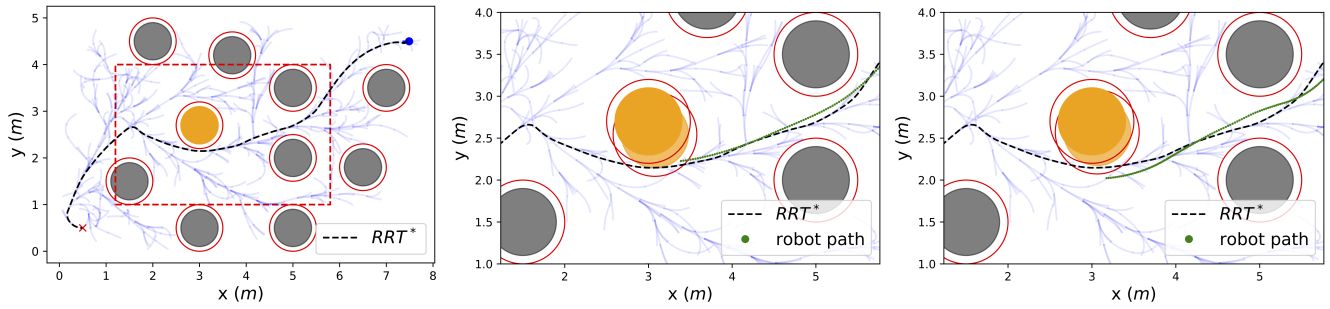
We conducted hardware experiments with one author acting as the human pilot. Algorithm 1 was implemented on a desktop computer (8-core, 32GB RAM, Ubuntu 20.04) using Robot Operating System (ROS). The experimental setup can be seen in Figure 1. We conduct experiments with both known and unknown obstacles – the robot only has knowledge of the cones, but cannot detect other obstacles. The human uses a Geomagic Touch[2] haptic device to steer an AglieX LIMO[3] robot through a remote environment.

The forward (back) position of the Touch's stylus mapped to positive (negative) linear velocity of the robot, and left (right) position of the stylus mapped to positive (negative) angular velocity. These steering commands (u_{ref}) are sent to the robot from the central computer via Wi-Fi, thus the human pilot has complete control over the robot steering. Force feedback was sent to the user through the Geomagic Touch, such that the feedback would force the stylus to a

[1] https://www.gurobi.com/

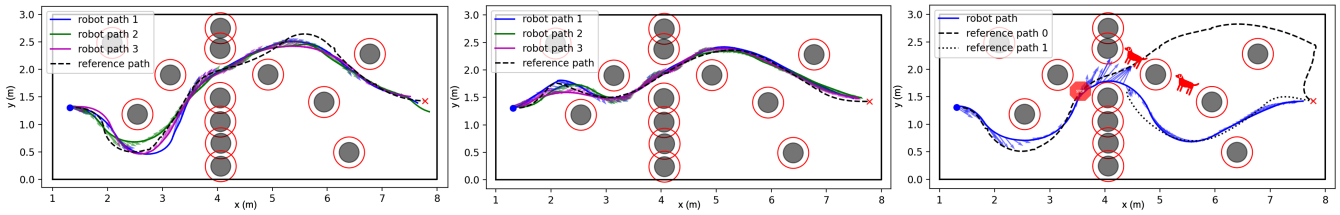
[2] https://www.3dsystems.com/haptics-devices/touch

[3] https://global.agilex.ai/products/limo



(a) $t = 0\,\text{s}$ (b) $t = 16\,\text{s}$, no CBF applied (c) $t = 20\,\text{s}$, CBF applied

Fig. 5: Robot reactions with a local dynamic obstacle. The blue $\cdot$ denotes the start point and the red $\times$ denotes the final goal x_f. The orange circle represents the dynamic obstacle, and the gray circles represent static obstacles. The red ring around each obstacle denotes the safe distance d_safe, and the blue lines are the generated RRT* tree $\mathcal{T}$. (a) shows the suggested path generated by RRT* and starting position of the dynamic obstacle at $t = 0\,\text{s}$. (b) is the result at $t = 16\,\text{s}$ when CBF is not applied. (c) is the result at $t = 20\,\text{s}$ when CBF is applied. Note that (b) and (c) are the zoomed-in plots for the region highlighted by the dashed red box in (a).



(a) Tree 1, agreeable human. (b) Tree 2, agreeable human. (c) Tree 3, unknown obstacles

Fig. 6: Human pilot hardware experiments, where the human is responsible for navigating from the blue $\cdot$ to the red $\times$. The experimental setup is shown in Figure 1. In (a)-(b), the human obeys the haptic command to follow the optimal path generated by RRT*, represented by the dashed black line. Solid lines represent the robot trajectory as navigated by the human. Vectors signify the force F (9) applied to the human along the trajectory. In (c), the human encounters obstacles unknown to the robot. First, the human must stop at the point shown by a stop sign. Then the human has to avoid two dogs blocking the initial suggested path (black dashed line). When the human rejects the optimal path, our algorithm suggests another path (black dotted line) from the RRT* tree to the goal.

position that would steer the robot onto the suggested path or away from obstacles. A real-world example of such force feedback is assisted driving through a haptic steering wheel. We demonstrate this force feedback on the Geomagic Touch in the supplemental video – when the human is navigating the robot through a narrow passage, they can allow the haptic device to take control and ensure collision-free navigation. The force F is converted to linear (F_v) and angular (F_ω) force as follows for appropriate input to the haptic device:

$$F_v = k_v \begin{bmatrix} \cos\theta & \sin\theta \end{bmatrix} F,$$
$$F_\omega = k_\omega \arctan\left(\frac{\begin{bmatrix} -\sin\theta & \cos\theta \end{bmatrix} F}{\begin{bmatrix} \cos\theta & \sin\theta \end{bmatrix} F} \right),$$

where $k_v, k_\omega \in \mathbb{R}$ are positive constants, and θ is the robot heading angle with respect to the global frame.

Note that the human operator does not have any information about the RRT* tree (i.e. the user cannot see the tree or the suggested path), and thus relies on the feedback for navigation assistance. The human has a first-person view of the environment via a camera mounted to the front of the robot, and its video feed was streamed to the central computer (see Figure 1a). We implemented two case studies in our human pilot experiments as follows.

1) Agreeable human: This first case can be considered the ideal scenario, in which the robot can detect all obstacles, and the human is able to follow the suggested path without any obstructions. We tested this case with two different RRT* trees, and thus two different suggested paths. An author teleoperated the robot, without resisting the force feedback, through the environment three times for each of these paths. Results are shown in Figures 6a and 6b, with vectors indicating the direction and relative magnitude of the force provided to the human. We observe that the human is able to follow the suggested path to the goal, without collisions, and without switching to a new path.

2) Human-robot disagreement: We also wish to demonstrate a scenario in which the human pilot disagrees with the planned path and decides to navigate the robot in a different direction. Such a disagreement usually happens when there is a local observation by the human that the RRT* planner does not account for, or the user has a strong preference for a different path. In such a case, the human should be able to override the force feedback, steer the robot in a safe direction, and the robot should be able to suggest a different path to the goal. In our experiments, we demonstrate a scenario in which the human and robot may disagree. Specifically, we present a case study in which some obstacles are unknown

to the robot. In Figure 1b, we show the experimental setup in which unknown obstacles (stuffed dogs) are placed in the environment, and we show our results for this case in Figure 6c. As seen in Figure 6c, the first suggested path, indicated by the dashed line, would require the user to collide with a dog. We see the magnitude of the force increase as the human deviates the robot from the path. Once the robot determines that the user has rejected the path (using Algorithm 2), the robot suggests a new path (dotted line) to the goal using Algorithm 3, after which the human and robot have little disagreement. Videos from these experiments are included in the supplement.

V. Conclusions

In this work, we propose an algorithm to enable collaborative navigation between a human operator and a ground robot. The human pilot uses a haptic device to steer the robot, and the robot provides collision avoidance and path suggestions to the human via force feedback. We implement a non-holonomic RRT* path planner for navigation suggestions, with a CLF to help drive the human toward and along the path. To correct any CLF inputs that would cause collisions with obstacles, such as small or dynamic obstacles, we utilize a CBF to ensure collision-free navigation. Our simulations and hardware experiments demonstrate an effective co-navigation algorithm, guaranteeing collision avoidance while ensuring the robot reaches the goal.

References

[1] A. M. Okamura, "Methods for haptic feedback in teleoperated robot-assisted surgery," *Industrial Robot: An International Journal*, vol. 31, no. 6, pp. 499–508, 2004.

[2] ——, "Haptic feedback in robot-assisted minimally invasive surgery," *Current opinion in urology*, vol. 19, no. 1, p. 102, 2009.

[3] F. Abi-Farraj, C. Pacchierotti, O. Arenz, G. Neumann, and P. R. Giordano, "A haptic shared-control architecture for guided multi-target robotic grasping," *IEEE transactions on haptics*, 2019.

[4] T. M. Lam, H. W. Boschloo, M. Mulder, and M. M. Van Paassen, "Artificial force field for haptic feedback in UAV teleoperation," *IEEE Trans. on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 39, no. 6, pp. 1316–1330, 2009.

[5] X. Hou and R. Mahony, "Dynamic kinesthetic boundary for haptic teleoperation of aerial robotic vehicles," in *IEEE Intl. Conf. on Intelligent Robots and Systems*, 2013, pp. 4549–4950.

[6] C. Masone, M. Mohammadi, P. Robuffo Giordano, and A. Franchi, "Shared planning and control for mobile robots with integral haptic feedback," *The International Journal of Robotics Research*, vol. 37, no. 11, pp. 1395–1420, 2018.

[7] D. Zhang, G. Yang, and R. P. Khurshid, "Haptic teleoperation of uavs through control barrier functions," *IEEE Trans. on Haptics*, vol. 13, no. 1, pp. 109–115, 2020.

[8] Y. Morère, M. A. Hadj Abdelkader, K. Cosnuau, G. Guilmois, and G. Bourhis, "Haptic control for powered wheelchair driving assistance," *IRBM*, vol. 36, no. 5, pp. 293–304, 2015.

[9] E. B. Vander Poorten, E. Demeester, E. Reekmans, J. Philips, A. Hüntemann, and J. De Schutter, "Powered wheelchair navigation assistance through kinematically correct environmental haptic feedback," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 3706–3712.

[10] G. Bourhis and M. Sahnoun, "Assisted Control Mode for a Smart Wheelchair," in *2007 IEEE 10th International Conference on Rehabilitation Robotics*, 2007, pp. 158–163.

[11] D. Zhang and R. Tron, "Stable haptic teleoperation of uavs via small l 2 gain and control barrier functions," in *2021 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2021, pp. 8352–8357.

[12] D. Zhang, R. Tron, and R. P. Khurshid, "Haptic feedback improves human-robot agreement and user satisfaction in shared-autonomy teleoperation," in *2021 IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2021, pp. 3306–3312.

[13] X. Hou, X. Wang, and R. Mahony, "Haptics-aided path planning and virtual fixture based dynamic kinesthetic boundary for bilateral teleoperation of VTOL aerial robots," in *2016 35th Chinese Control Conference (CCC)*, 2016, pp. 4705–4710.

[14] M. Coffey and A. Pierson, "Collaborative Teleoperation with Haptic Feedback for Collision-Free Navigation of Ground Robots," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 8141–8148.

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

[16] W. Xiao and C. Belta, "Control barrier functions for systems with high relative degree," in *Proc. 58th IEEE Conference on Decision and Control*, Nice, France, 2019, pp. 474–479.

[17] 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, 2017.

[18] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[19] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *European Control Conference (ECC)*. IEEE, 2019, pp. 3420–3431.

[20] B. Xu and K. Sreenath, "Safe Teleoperation of Dynamic UAVs Through Control Barrier Functions," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 7848–7855.

[21] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[22] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 1478–1483.

[23] J. J. Park and B. Kuipers, "Feedback motion planning via non-holonomic RRT for mobile robots," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 4035–4040.

[24] G. Arechavaleta, J.-P. Laumond, H. Hicheur, and A. Berthoz, "The nonholonomic nature of human locomotion: a modeling study," in *The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, 2006. BioRob 2006.* IEEE, 2006.

[25] G. Yang, B. Vang, Z. Serlin, C. Belta, and R. Tron, "Sampling-based motion planning via control barrier functions," in *Proceedings of the 2019 3rd International Conference on Automation, Control and Robots*, ser. ICACR 2019, no. 8, New York, NY, USA, 2019, p. 22–29.