IEEE Control Systems Letters paper presented at
2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

Safe Returning FaSTrack with Robust Control Lyapunov-Value Functions

Zheng Gong*, Boyang Li* and Sylvia Herbert

***Abstract*— Real-time navigation in *a priori* unknown environment remains a challenging task, especially when an unexpected (unmodeled) disturbance occurs. In this paper, we propose the framework Safe Returning Fast and Safe Tracking (SR-F) that merges concepts from 1) Robust Control Lyapunov-Value Functions (R-CLVF) [1], and 2) the Fast and Safe Tracking (FaSTrack) framework [2]. The SR-F computes an R-CLVF offline between a model of the true system and a simplified planning model. Online, a planning algorithm is used to generate a trajectory in the simplified planning space, and the R-CLVF is used to provide a tracking controller that exponentially stabilizes to the planning model. When an unexpected disturbance occurs, the proposed SR-F algorithm provides a means for the true system to recover to the planning model. We take advantage of this mechanism to induce an artificial disturbance by "jumping" the planning model in open environments, forcing faster navigation. Therefore, this algorithm can both reject unexpected true disturbances and accelerate navigation speed. We validate our framework using a 10D quadrotor system and show that SR-F is empirically 20% faster than the existing works while maintaining safety.**

I. Introduction

Safe control for autonomous systems is a challenging task, particularly for dynamic systems navigating through *a priori* unknown environments. For computational efficiency, many algorithms use a simplified (often kinematic) model of the system to generate a path around obstacles to a goal. A more complex model representing the true robot is then used to track this path. Popular path planning algorithms include Dijkstra's [3], A* [4], Rapidly Exploring Random Trees (RRT) [5] and heuristic-based methods [6], [7]. The tracking controller can be generated using, for example, model predictive control (MPC) [8], [9], or control Lyapunov functions (CLFs) [10], [11]. For safety, control barrier functions (CBFs) [12] or Hamilton Jacobi (HJ) reachability analysis [13], [14] can generate safety filters for the controller.

Since planning is typically done with a simplified model of the system, the path might not be feasible and safe for the actual robot to track. To address this issue, [15], [16] directly adds a CBF and CLF as a constraint in the path planning algorithm. [17] uses a reference governor control design wherein a robot with a specific form of dynamics can safely stabilize to a moving equilibrium point. Integrated planning and control (IPC) uses the notion of a safe flight corridor (SFC) as a safety constraint in nonlinear MPC to guarantee safe navigation [18], [19], and shows empirical robustness to sudden disturbances.

This research is supported by ONR YIP (#N00014-22-1-2292) and the UCSD JSOE Early Career Faculty Award. *Both authors contributed equally to this work. All authors are in Mechanical and Aerospace Engineering at UC San Diego {zhgong,bol025, sherbert}@ucsd.edu.

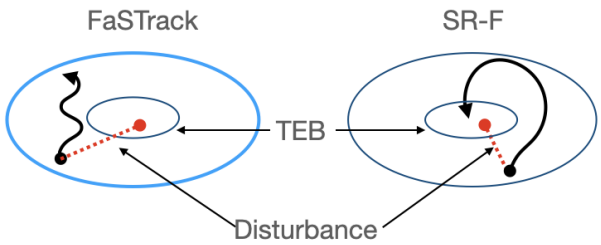

Fig. 1: Comparison of the relative trajectory using the original FaSTrack framework (left) and our proposed SR-F (right). The red dotted line denotes an unexpected disturbance that causes the relative state to leave the minimum tracking error bound (TEB). FaSTrack can only guarantee the relative state stays in the larger error bound, while the SR-F can stabilize the relative state back to the TEB.

Fast and Safe Tracking (FaSTrack) [2] is a modular framework that separates the navigation task into independent planning and tracking tasks (with corresponding planner and tracker models of the autonomous system). Offline, HJ reachability is used to precompute a tracking error bound (TEB) on the maximum deviation that the true tracker model may take from the planner model (Fig. 1). This is paired with an optimal tracking controller that maintains this error bound regardless of the planning algorithm used by the planner. Online, the obstacles are augmented with TEB and the planning algorithm provides a path in the low-dimensional planning space around the augmented obstacles. The tracking controller guarantees the distance between the tracker and the path is contained in the TEB, preserving safety.

While several of the above approaches can handle predefined disturbance bounds, they are not designed to maintain safety when experiencing a *sudden disturbance beyond the expected bounds*. In this paper, we modify the FaSTrack framework and propose the novel Safe Returning FaSTrack (SR-F) framework. The main contributions are as follows:

1) We introduce the SR-F, where a CLF-like function in the relative space between the tracker and planner is computed offline, and a new *safe returning* mechanism is used to accommodate unexpected disturbances. We prove (under mild assumptions) that the SR-F can maintain safety under unexpected disturbances.
2) We take advantage of this robustness to sudden disturbances by methodically introducing an artificial sudden disturbance by "jumping" the planner towards the goal, forcing the autonomous system to speed up in open environments while maintaining safety.

3) We compare SR-F with existing work [19], [20] on 8D and 10D quadrotor navigation tasks that are subjected to sudden high wind gusts. We show the SR-F can maintain safety when unexpected disturbance happens, and outperforms existing methods on navigation speed.

II. Background

We consider three models: (1) a tracker model that represents the true robot, (2) a planner model that is designed by the user for path planning, and (3) a relative model used to guarantee safety.

1) Tracker Model: The tracker model is given by the following nonlinear ordinary differential equation:

$$\frac{dx}{ds} = \dot{x} = f(x, u, d), \quad x(t) = x_0, \quad s \in [t, 0], \tag{1}$$

where s is the time, $x \in \mathcal{X} \subseteq \mathbb{R}^n$ is the tracker state, $u \in \mathcal{U}_s \subseteq \mathbb{R}^m$ is the control input, and $d \in \mathcal{D} \subseteq \mathbb{R}^d$ is the disturbance. Assume the dynamics $f : \mathcal{X} \times \mathcal{U}_s \times \mathcal{D} \mapsto \mathcal{X}$ is Lipschitz continuous in x for fixed u, d. Assume the control and disturbance signal $u(\cdot)$, $d(\cdot)$ are measurable functions:

$$u(\cdot) \in \mathbb{U}_s := \{u : [t, 0] \mapsto \mathcal{U}_s, u(\cdot) \text{ is measurable}\},$$
$$d(\cdot) \in \mathbb{D} := \{d : [t, 0] \mapsto \mathcal{D}, d(\cdot) \text{ is measurable}\},$$

where $\mathcal{U}_s$ and $\mathcal{D}$ are compact sets. Under these assumptions, we can solve for a unique solution of (1), denoted as $\xi_f(s; t, x, u(\cdot), d(\cdot))$. Denote $\mathcal{G} \subset \mathcal{X}$ the goal set, and $\mathcal{C} \subset \mathcal{X}$ the constraint set, i.e., the set of states that we want to avoid.

2) Planner Model: The planner model is given by:

$$\frac{dp}{ds} = \dot{p} = h(p, u_p), \quad p(t) = p_0,$$

where $p \in \mathcal{P} \subseteq \mathbb{R}^p$ is the planner state, $u_p \in \mathcal{U}_p$ is the planner control. Further, assume that $\mathcal{P}$ is a subspace of $\mathcal{X}$ and we make analogous assumptions on the planner model dynamics as for (1) to guarantee a unique solution.

The goal and constraint sets in the planner space are denoted as $\mathcal{G}_p \subset \mathcal{P}$, $\mathcal{C}_p \subset \mathcal{P}$ respectively.

3) Relative Dynamics: Define the relative state

$$r = \Phi(x, p)(x - Qp), \tag{2}$$

where $r \in \mathcal{R} \in \mathbb{R}^n$, Q augments the planner state and Φ is a linear map so that the dynamics can be written as

$$\dot{r} = g(r, u, u_p, d). \tag{3}$$

The existence of Q and Φ are justified in [2]. From the assumption of the tracker and planner model, the relative dynamics also admits unique solution $\xi(s; t, r, u(\cdot), u_p(\cdot), d(\cdot))$. Denote the error states between tracker and planner as e and the rest as η, i.e., $r = [e, \eta]$.

A. HJ Reachability and Fastrack

The FaSTrack framework contains two parts: offline computation and online execution. The offline part uses the HJ reachability to generate the TEB, which is a robust control invariant set. Online, it senses the environment, augments the obstacles with the TEB, and then plans and tracks a path around the augmented obstacles.

1) HJ reachability (Offline): HJ reachability can be formulated and solved as an optimal control problem. Specifically, the cost function $\ell : \mathcal{R} \mapsto \mathbb{R}^+$ is designed to measure distance (via the Euclidean norm) in the relative state space. The tracker control u tries its best to track the planner and minimize this cost, whereas the disturbance d and planner control u_p try to escape the tracker as far as possible by maximizing this cost. Because the environment and planning algorithm are not necessarily known *a priori*, we assume the worst-case scenario, i.e. that the u_p, d can act optimally to u. We define their strategies as mappings $\lambda_p : \mathcal{U}_s \mapsto \mathcal{U}_p$, $\lambda_d : \mathcal{U}_s \mapsto \mathcal{D}$. We further restrict them to be non-anticipative $\lambda_p \in \Lambda_p$, $\lambda_d \in \Lambda_d$ [14]. The value function is given by

$$V(r, t) = \max_{\lambda_p \in \Lambda_p, \lambda_d \in \Lambda_d} \min_{u \in \mathbb{U}_s} \{ \max_{s \in [t, 0]} \ell(\xi(s; t, r, u(\cdot), \lambda_p(\cdot), \lambda_d(\cdot)))\}.$$

This value function captures the worst-case tracking error when the tracker is acting optimally and the disturbance and planner are acting adversarially. We assume the following limit exits on a compact set, i.e., it converges:

$$V^\infty(r) = \lim_{t \to -\infty} V(r, t). \tag{4}$$

The minimal value of (4) is denoted $\underline{V}^\infty$, whose level set provides the TEB in the relative state space. For planning, we can project this bound into the error state space:

$$\mathcal{B}_e := \{e : \exists \eta \text{ s.t. } V^\infty(e, \eta) \leq \underline{V}^\infty\}.$$

If $\mathcal{C}$ is known in advance, we could compute the *inevitable backward reachable tube* of the tracker to $\mathcal{C}$, i.e., the set of states such that the collision must happen [14].

2) Online Execution: The FaSTrack augments sensed obstacles by $\mathcal{B}_e$ and employs any planning algorithm to output the next state of the planner model. The gradients of the precomputed value function $V^\infty(r)$ inform a linear program to compute optimal control for the tracker model to pursue the planner model. This process is repeated until the tracker model reaches the goal. Safety is guaranteed as long as disturbances fall within the expected bounds [2].

Remark 1. The value function is computed with a prespecified disturbance bound $\mathcal{D}$. A larger $\mathcal{D}$ corresponds to a larger TEB, which causes the augmented environment to be denser, impacting performance. However, this also makes the system more robust to the disturbance. On the other hand, a smaller $\mathcal{D}$ results in a smaller TEB, and therefore a sparser augmented environment, and better average performance, but is less robust to disturbances.

B. R-CLVF

Recently, [1] proposed the robust control Lyapunov value function (R-CLVF), defined as:

Definition 1. R-CLVF $V_\gamma^\infty : D_\gamma \mapsto \mathbb{R}$ of (3) is

$$V_\gamma^\infty(r) = \lim_{t \to -\infty} \max_{\lambda_p \in \Lambda_p, \lambda_d \in \Lambda_d} \min_{u_s \in \mathbb{U}_s} \{ \max_{s \in [t, 0]} e^{\gamma(s-t)} \ell(\xi(s))\}.$$

Here, $D_\gamma \subseteq \mathbb{R}^n$ is the domain, γ is a user-specified parameter that represents the desired decay rate, and $\ell(x) = ||x|| - \underline{V}^\infty$.

When $\gamma = 0$, the R-CLVF is equivalent to the infinite-time HJ value function (4). Prop. 3 in [1] shows that for all $\gamma \geq 0$, the R-CLVFs have the same zero-level set. In other words, for all $\gamma \geq 0$, the zero-level set of the R-CLVFs is the TEB.

The R-CLVF value of r captures the largest exponentially amplified deviation of a trajectory starting from r to the TEB, under worst-case disturbance. If this value is finite, it means r can be exponentially stabilized to the TEB (Lem. 7 of [1]).

Theorem 1. The relative state can be exponentially stabilized to the TEB from $\mathcal{D}_\gamma \setminus \mathcal{B}$, if the R-CLVF exists in $\mathcal{D}_\gamma$.

$$\min_{a \in \partial\mathcal{B}} ||\xi(s) - a|| \leq k e^{-\gamma(s-t)} \min_{a \in \partial\mathcal{B}} ||r - a||, \tag{5}$$

where $k > 0$ and $t \leq s \leq 0$.

The R-CLVF can be computed by solving the following R-CLVF-VI until convergence

$$\begin{aligned} 0 = \max\{&\ell(r) - V_\gamma^\infty(r), \\ &\min_{u \in \mathcal{U}_s} \max_{u_p \in \mathcal{U}_p, d \in \mathcal{D}} \frac{dV_\gamma^\infty}{dr} \cdot g(r, u, u_p, d) + \gamma V_\gamma^\infty\}. \end{aligned}$$

The R-CLVF optimal controller is

$$u^* = \arg\min_{u \in \mathcal{U}_s} \max_{u_p \in \mathcal{U}_p, d \in \mathcal{D}} \frac{dV_\gamma^\infty}{dr} \cdot g(r, u, u_p, d). \tag{6}$$

III. Safe Returning with Unexpected Disturbance

FaSTrack is robust to bounded pre-specified disturbances. However, unexpected and infrequent short-duration disturbances can happen because of communication delays, sudden external forces (e.g. a strong wind), or model mismatch. After a sudden *unexpected* disturbance event that causes the tracker to leave the TEB, the FaSTrack framework only guarantees that the tracker will not exit the *current* level set of the relative value function. This is visualized in Fig. 1, left. The corresponding error bound that must be used to augment obstacles is shown in blue, resulting in conservative plans.

We propose using the R-CLVF to guarantee that the relative states stabilize back to the TEB at the desired rate γ. Alternatively, one can try to find a CBF that represents the TEB, and design a controller to stabilize the relative states to this TEB, which is hard for high-dimensional systems with disturbances and input constraints. We present the SR-F framework and highlight two important implications

1) After an unexpected disturbance event, the relative state will converge back to the TEB at an exponential rate γ.
2) We can take advantage of this convergence property by introducing an *artificial disturbance* that "jumps" the planner forward towards the goal when safe to do so, speeding up navigation process.

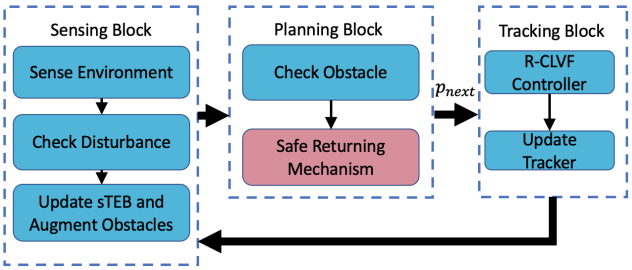

Fig. 2: Online flowchart for SR-F. The online algorithm contains three main blocks: the sensing block, the planning block, and the tracking block. The sensing block senses the environment, determines the sTEB and augments the obstacle, and checks if unexpected disturbances happen. The planning block checks if the sensed environment is free of obstacles, and use the safe returning function to determine the next plan state p_{next} (and raw path). The tracking block takes in p_{next}, determines the optimal controller, and updates the tracker state.

A. SR-F Algorithm

The overall algorithm is shown in Alg. 1, with a flowchart shown in Fig. 2. We begin by explaining this algorithm at a high level. First the "sensing block" senses the environment and any unexpected disturbances, then augments obstacles by the *maximum safe resetting region* (sTEB).

Next the "planning block" by default employs a planning algorithm to generate a path through the sensed environment that obeys the dynamics of the planner model. This planning block has modifications for two scenarios: 1) if a sudden disturbance has occurred, the planner may be moved in a way to ensure that the tracker will not hit an obstacle as it converges back to the TEB, 2) if there is an opportunity to do so safely, the planner will "jump" ahead towards the goal, forcing the tracker to converge back towards it at the rate γ.

Finally, there is a "tracking block," which updates the current relative state between the tracker and planner, and applies the pre-computed optimal controller to the tracker that minimizes the distance between itself and the planner.

B. Sensing Block

Initialization. Every iteration starts with checking if the tracker has experienced an unexpected disturbance, which we assume does not cause failure immediately.

Environment Sensing. The robot senses the environment, updates the constraint $\mathcal{C}_{\text{sensed}}$ (also in the planner space $\mathcal{C}_{\text{p,sensed}}$), and finds the distance from the tracker to the nearest obstacle within sensing range. This distance is given by

$$dst(x; \mathcal{C}_{\text{sensed}}) = \begin{cases} \mathcal{R} & \text{no obstacle} \\ \min_{a \in \partial\mathcal{C}_{\text{sensed}}} ||x - a|| & \text{otherwise} \end{cases}. \tag{7}$$

If a new obstacle is sensed, we assign 1 to ReplanFlag (RF).

Computation of the Max Safe Resetting Region, sTEB. Since the planner model is a virtual model with no physical realization, the framework can reset the planner state arbitrarily if needed to ensure that the tracker does not collide with

Algorithm 1: SR-FaSTrack

Require: V_γ^∞, $\mathcal{B}$, sense range $\mathcal{R}$, initial states x_0, p_0.

Initialization:
$x \leftarrow x_0$, $x_{\text{old}} \leftarrow x$, $p \leftarrow p_0$, $t \leftarrow 0$, sTEB $\leftarrow \mathcal{B}$, JF $\leftarrow 0$, RF $\leftarrow 1$
while Goal not reached **do**
  Sensing Block
  If unexpected disturbance happens ($x \neq x_{\text{old}}$), update relative state: $r \leftarrow \Phi(x,p)(x - Qp)$
  Sense environment, update $\mathcal{C}_{\text{p, sense}}$, and update distance from the tracker to the obstacle using (7)
  RF $\leftarrow 1$ if new obstacle sensed
  Find safe resetting region $\mathcal{S}$ using (8), augment obstacle with $\mathcal{S}_e$ and update $\mathcal{C}_{\text{p, aug}}$
  Planning Block
  if $V_\gamma^\infty(r) > 0$ **then** JF $\leftarrow 1$
  else if $V_\gamma^\infty(r) \leq 0$ **then**
    if $\mathcal{C}_{\text{p, sense}}$ is obstacle free **then** JF $\leftarrow 1$
    else if Not obstacle free **then** JF $\leftarrow 0$
    end if
  end if
  JF, RF, p_{next}, $p_{\text{raw}} \leftarrow$ SafeReturn($x, \mathcal{U}_p$, JF, RF, p_{raw}, $\mathcal{C}_{\text{p, aug}}$)
  Tracking Block
  $p \leftarrow p_{\text{next}}$, $r \leftarrow \Phi(x,p)(x - Qp)$, $u \leftarrow u^*$ using (6)
  Update tracker state: $x \leftarrow \text{nextTrack}(x, u)$
  $r, r_{\text{old}} \leftarrow \Phi(x,p)(x - Qp)$, $s \leftarrow s + \Delta s$
end while

Algorithm 2: Safe Returning Function

Require: x, $\mathcal{U}_p$, JF, RF, p_{raw}, $\mathcal{C}_{\text{p,aug}}$

Output: Next plan state p_{next}, p_{raw}, JF, RF
if JF = 1 **then**
  p_{next}, $p_{\text{raw}} \leftarrow$ the closest point to the target s.t. $\Phi(x, p_{\text{next}})(x - Qp) \in$ sTEB and $p \notin \mathcal{C}_{\text{p,aug}}$
  RF $\leftarrow 1$
else if JF = 0 **then**
  if RF = 1 **then**
    $p_{\text{raw}} \leftarrow \text{PathPlanningAlgo}(p, \mathcal{C}_{\text{p,aug}})$
  end if
  $p_{\text{next}} \leftarrow \text{nextPlan}(p_{\text{raw}}, \mathcal{U}_p)$
  remove p_{next} from p_{raw} if $p_{\text{next}} \in p_{\text{raw}}$, otherwise $p_{\text{raw}} \leftarrow p_{\text{raw}}$
  RF $\leftarrow 0$
end if
JF $\leftarrow 0$
Return p_{next}, p_{raw}, JF, RF

obstacles as it converges back to the planner. We provide a method to find the sTEB, which is denoted as $\mathcal{S}$. Consider a hyperball in the relative state space with radius $dst(x)/2$ and centered at the origin: $B(0, dst(x)/2)$. If the TEB $\mathcal{B}$ is contained in this ball $B(0, dst(x)/2)$, the sTEB is the largest sub-level set of the R-CLVF contained in $B(0, dst(x)/2)$. Otherwise, the sTEB is the TEB:

$$\mathcal{S} = \begin{cases} \mathcal{B} & \mathcal{B} \nsubseteq B(0, dst/2) \\ \text{largest sub-level set} & \mathcal{B} \subseteq B(0, dst/2) \end{cases}. \quad (8)$$

The sTEB in the planner space is given by

$$\mathcal{S}_e := \{e : \exists \eta \text{ s.t. } [e, \eta] \in \mathcal{S}\}. \quad (9)$$

Augmentation of Obstacles. $\mathcal{S}_e$ is used to augment the obstacles and update the augmented constraint set $\mathcal{C}_{\text{p,aug}}$. The outputs of the sensing block are the sensed and augmented obstacle map $\mathcal{C}_{\text{p,sense}}$, $\mathcal{C}_{\text{p,aug}}$, sTEB, and the RF.

Remark 2. To guarantee safety, the consideration of the hyperball $B(0, dst(x)/2)$ is necessary, and its radius must be at least $dst(x)/2$. The reason is that though exponential convergence to the TEB is guaranteed using R-CLVF, it is not necessary that for the next immediate time step, the norm of relative state decreases. This is because of the constant amplifier k in (5). We illustrate this issue in Fig. 1, right. With the hyperball $B(0, dst(x)/2)$, we guarantee that the distance between the planner and tracker is always smaller than the distance between the planner and the obstacle.

C. Planning Block and the Safe Returning Function

Jump Evaluation. The planning block begins by evaluating whether the planner should "jump" from its current state. This occurs under two conditions. The first condition occurs when the relative state indicates that it is outside of the TEB (i.e. $V_\gamma^\infty(r) > 0$). In this case the planner must jump to ensure that the tracker does not collide with an obstacle as it converges back to the TEB. The second condition is when there are no obstacles within the sensing radius. In this case, the planner creates an artificial disturbance by intentionally "jumping" to a further point on its path, increasing the relative state r and forcing it to leave the TEB. This accelerates the navigation as the tracker works to converge back at an exponential rate while obeying its control bounds. If either of these conditions for jumping occurs, the JumpFlag (JF) is set to 1.

Safe Returning Function. If the JF = 1, the safe returning function sets p_{next} as the state that is closest to the target, free of the augmented obstacles, and guarantee the relative state is in the sTEB (i.e., $\Phi(x, p_{\text{next}})(x - Qp_{\text{next}}) \in \mathcal{S}$). We assign 1 to the RF, indicating that the planning algorithm should plan a new path from p_{next}. The JF is reset to 0.

Replan. If the ReplanFlag has been activated, either from a jump or a new obstacle detected, the path planning algorithm is used to generate a new path for the planner. This path is processed by the function *nextPlan*, which converts the path into a trajectory that obeys the dynamics and control bounds of the planner. We then reset the RF to 0.

D. Tracking Block

We update the planner state using p_{next}, and update the relative state r using (2). The tracking controller u is determined by (6), which is then sent to the tracker model

and updates the tracker state. Note that we keep track of r_{old}, which is used to check if disturbance happens in the next iteration (lines 18-20 of Algorithm 1).

Theorem 2. Safety is guaranteed using SR-F if the disturbance does not push the tracker in its *inevitable backward reachable tube* of $\mathcal{C}$ (as defined in [14]).

Proof. Assume the JF=0 for some time step, the SR-F works just like the FaSTrack, and safety is guaranteed [2].

Assume JF≠0 at some time step. After resetting the planner state and before tracking, denote planner, tracker, and relative states as p_{next}, x_1 and r_1. From line 3 of Algorithm 2, p_{next} is chosen such that $p_{\text{next}} \notin \mathcal{C}_{\text{p,aug}}$, which means the sTEB centered at x_1 is obstacle free. After applying controller (6), denote the new tracker and relative states as x_2 and r_2. r_2 must be contained in a strict subset of the sTEB (by Theorem 1), which is also obstacle-free. This suggests that x_2 is free of obstacles, and safety is guaranteed for the next time step. The overall navigation process is a combination of JF = 1 and JF = 0, and for both cases, immediate safety is guaranteed. We conclude that the whole navigation process is safe concerning modeled and unexpected disturbances. □

Remark 3. We provide two benefits compared with the FaSTrack. 1) SR-F is robust to unexpected disturbances, 2) in the obstacle-free region, we mimic a "beneficial disturbance" to make the planner jump, accelerating the navigation.

IV. Experiments

We demonstrate that SR-F can provide safety guarantees given unexpected disturbances, and accelerate the navigation process. We consider two examples: 1) an 8D quadrotor model tracking a 2D integrator planner model with the A* planner and 2) a 10D near-hover quadrotor tracking a 3D integrator planner model with the RRT planner. We compare our method with FaSTrack, Meta-FaSTrack (M-F) [20], and IPC [19]. All simulations are conducted in MATLAB. Code can be found at https://github.com/UCSD-SASLab/Safe-Returning-FT.

A. Offline computation

1) 10D − 3D: The system dynamics of the 10D quadrotor (tracker) and the 3D integrator (planner) are from Example B in [2]. The tracker states (x, y, z) denote the position, (v_x, v_y, v_z) denote the velocity, (θ_x, θ_y) denote the pitch and roll, (ω_x, ω_y) denote the pitch and roll rates. The tracker has controls (u_x, u_y, u_z), representing the desired pitch and roll angle and the vertical thrust. The planner has controls $(\hat{v}_x, \hat{v}_y, \hat{v}_z)$, representing the velocity in each positional dimension. The system parameters are set to be $d_0 = 10, d_1 = 8, n_0 = 10, k_T = 0.91, g = 9.81$, $|u_x|, |u_y| \leq \pi/9$, $u_z \in [0, 1.5g]$, $|\hat{v}_x|, |\hat{v}_y|, |\hat{v}_z| \leq 0.5$, $d_x = d_y = d_z = 0$.

The relative dynamics can be obtained as

$$\begin{aligned}
&\dot{x}_r = v_x - \hat{v}_x + d_x, \quad \dot{v}_x = g\tan\theta_x, \quad \dot{\theta}_x = -d_1\theta_x + \omega_x, \\
&\dot{\omega}_x = -d_0\theta_x + n_0 u_x, \quad \dot{y}_r = v_y - \hat{v}_y + d_y, \quad \dot{v}_y = g\tan\theta_y, \\
&\dot{\theta}_y = -d_1\theta_y + \omega_y, \quad \dot{\omega}_y = -d_0\theta_y + n_0 u_y, \\
&\dot{z}_r = v_z - \hat{v}_z + d_z, \quad \dot{v}_z = k_T u_z - g. \qquad (10)
\end{aligned}$$

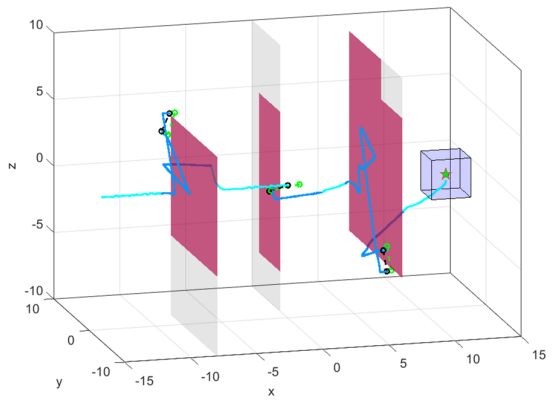

Fig. 3: 10D-3D simulation using SR-F. The tracker tracks a RRT path when not obstacle-free (blue), and jumps ahead on the path (cyan) when obstacle-free. The planner's position is the green star in the translucent blue box (representing sTEB). Both systems start on the left and navigate to a goal on the right. The three light grey rectangles are obstacles, and once sensed by the quadrotor they turn red. When the quadrotor is passing near an obstacle, it experiences an unexpected disturbance to its position (black dashed line), mimicking a sudden wind gust. The green dashed line shows the change of the planner's position after replanning.

This is decomposed into three independent subsystems $(x_r, v_x, \theta_x, \omega_x)$, $(y_r, v_y, \theta_y, \omega_y)$, (z_r, v_z) [21], allowing us to solve for the R-CLVF more tractably.

2) 8D − 2D: The relative dynamics of the 8D tracker and the 2D planner are the x, y subsystems above.

B. Online Planning and Navigation

1) 10D − 3D: we compare SR-F with FaSTrack and M-F. The result is shown in Fig. 3 and Table I. We design three experiments with different disturbance settings: a) no disturbance, b) unexpected disturbance to the position states pushing the tracker to the obstacle (like a sudden wind), and c) unexpected disturbance to the position and velocity states that act in the worst-case. When no disturbance exists, safety is guaranteed for all three frameworks. When unexpected disturbances exist, both the FaSTrack and M-F collide for more than 80% of runs. Since the positional disturbances push the tracker to the obstacle, M-F and FaSTrack are prone to crash. However, the SR-F can survive these disturbances, showcasing the safe-returning property. When unexpected disturbances are generated by uniformly distributed noise, M-F and FaSTrack collide in less than 10% of experiments.

We highlight that SR-F guarantees safety under unexpected disturbances, though it takes more time to reach the goal. This is because FaSTrack and M-F do not consider the unexpected disturbance, and do not spend time to replan.

TABLE I: Comparison of FaSTrack, M-F, and SR-F for the 10D-3D system. Each row is averaged across 40 runs.

Types of Disturbance	No Dist			Pos Dist			Pos + Vel Dist		
Metrics	FaSTrack	M-F	SR-F	FaSTrack	M-F	SR-F	FaSTrack	M-F	SR-F
Reach Goal (%)	100	100	100	15	12	100	11	10	100
Obstacle Collision (%)	0	0	0	85	88	0	89	90	0
Navigation Time (s)	96	77	81	102	70	115	101	73	121

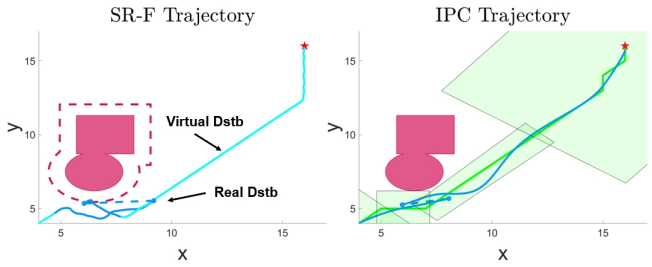

Fig. 4: Online simulations for an 8D quadrotor tracking a 2D planning model paired with an A* planner. Results using SR-F (ours) and IPC are shown. Left: the entire trajectory using SR-F. The quadrotor starts at $(4, 4)$ and navigates to the goal at $(16, 16)$ (red star). The obstacle (red) is augmented by the TEB. The system's trajectory is shown in cyan (when jumping) and blue (when tracking). A position disturbance (labeled "Real Dstb") is applied to the quadrotor, pushing it (blue dashed line) close to the obstacle. Right: the trajectory (blue) using IPC, with the same start and goal as SR-F. The SFC is shown as the light green region, and the path given by A* is shown in green.

However, it is preferable to sacrifice the navigation speed for the safety guarantee in most real-world applications. Also, note that the navigation speed is affected by the planning algorithm used and the environment.

2)8D − 2D: we compare SR-F with IPC. We construct two scenarios: a) a relatively larger disturbance to the position states pushing the tracker to the obstacle (see Fig. 4 and column 1 of Table II), and b) uniformly distributed smaller position disturbances ($\Delta x, \Delta y \in [-0.2, 0.2]$) that are randomly added (Table II). The simulation time is 40s with a 0.1s time step. In all simulations, SR-F safely navigates the quadrotor to the goal. When the large disturbance is applied, IPC collides in 45% of runs. When 3 and 5 smaller random disturbances are applied, IPC fails to reach the goal within the simulation time for 20% and 75% of runs, though no collisions occur. When only one disturbance happens, the IPC is faster than SR-F, but as the occurrence of the disturbances increases, the navigation speed of IPC decreases significantly, and the SR-F outperforms it. The reason is that the IPC's controller does not consider disturbance and the safety is guaranteed using the safe flight corridor (which works for the case where disturbance does not push the system out of it.). The presence of disturbances greatly impacts the performance of the MPC controller.

TABLE II: Comparison of SR-F and IPC frameworks for 8D-2D system. Each row is averaged across 40 runs.

Occurance of Disturbance	1		3		5	
Metrics	SR-F	IPC	SR-F	IPC	SR-F	IPC
Reach Goal (%)	100	55	100	80	100	25
Obstacle Collision (%)	0	45	0	0	0	0
Navigation Time (s)	35	23	31	35	31	38

V. Conclusion

In this paper, we introduced the SR-FaSTrack framework, which can be used to reject unexpected disturbances during navigation in *a priori* unknown environments. It also accelerates navigation by intentionally making the planner "jump" as a virtual disturbance in open environments.

Future work includes extending to multi-agent systems and dealing with moving obstacles, modifying the R-CLVF to get differently shaped TEBs, combining with deep neural networks to better accommodate different environments, and implementing hardware demonstrations.

References

[1] Z. Gong and S. Herbert, "Robust control lyapunov-value functions for nonlinear disturbed systems," https://arxiv.org/abs/2403.03455, 2024.

[2] M. Chen*, S. Herbert*, H. Hu, Y. Pu, J. Fisac, S. Bansal, S. Han, and C. Tomlin, "Fastrack: a modular framework for real-time motion planning and guaranteed safe tracking," in *Trans. on Automatic Control*. IEEE, 2020.

[3] E. W. Dijkstra, "A note on two problems in connection with graphs," in *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, 2022.

[4] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Trans. on Systems Science and Cybernetics*, 1968.

[5] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *International Conference on Intelligent Robots and Systems*. IEEE, 2002.

[6] A. Stentz *et al.*, "The focused D^* algorithm for real-time replanning," in *International Joint Conferences on Artificial Intelligence*, 1995.

[7] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic a*: An anytime, replanning algorithm," in *International Conference on Automated Planning and Scheduling*, 2005.

[8] P. Bouffard, "On-board model predictive control of a quadrotor helicopter: Design, implementation, and experiments," Master's thesis, EECS Department, University of California, Berkeley, Dec 2012.

[9] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[10] Z. Artstein, "Stabilization with relaxed controls," *Nonlinear Analysis: Theory, Methods & Applications*, 1983.

[11] E. D. Sontag, "A 'universal' construction of Artstein's theorem on nonlinear stabilization," *Systems & control letters*, 1989.

[12] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *European Control Conf.* IEEE, 2019.

[13] L. C. Evans and P. E. Souganidis, "Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations," *Indiana University Mathematics Journal*, 1984.

[14] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: A brief overview and recent advances," in *Conference on Decision and Control*. IEEE, 2017.

[15] F. S. Barbosa, L. Lindemann, D. V. Dimarogonas, and J. Tumova, "Provably safe control of lagrangian systems in obstacle-scattered environments," in *Conference on Decision and Control*. IEEE, 2020.

[16] A. Manjunath and Q. Nguyen, "Safe and robust motion planning for dynamic robotics via control barrier functions," in *Conference on Decision and Control*. IEEE, 2021.

[17] Z. Li and N. Atanasov, "Governor-parameterized barrier function for safe output tracking with locally sensed constraints," *Automatica*, 2023.

[18] Y. Wu, Z. Ding, C. Xu, and F. Gao, "External forces resilient safe motion planning for quadrotor," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8506–8513, 2021.

[19] W. Liu, Y. Ren, and F. Zhang, "Integrated planning and control for quadrotor navigation in presence of suddenly appearing objects and disturbances," *IEEE Robotics and Automation Letters*, 2023.

[20] D. Fridovich-Keil, S. L. Herbert, J. F. Fisac, S. Deglurkar, and C. J. Tomlin, "Planning, fast and slow: A framework for adaptive real-time safe trajectory planning," in *International Conference on Robotics and Automation*. IEEE, 2018.

[21] C. He, Z. Gong, M. Chen, and S. Herbert, "Efficient and guaranteed hamilton–jacobi reachability via self-contained subsystem decomposition and admissible control sets," *Control Systems Letters*, 2023.