TinySDP: Real Time Semidefinite Optimization for Certifiable and Agile Edge Robotics

Ishaan Mahajan[1], Jon Arrizabalaga[2,‡], Andrea Grillo[3,4,‡], Fausto Vega[2,‡],
James Anderson[1,†], Zachary Manchester[2,†], and Brian Plancher[3]
[1]Columbia University, USA [2]Massachusetts Institute of Technology, USA [3]Dartmouth College, USA
[4]Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland
[‡]Equal contribution [†]Equal advising

***Abstract*—Semidefinite programming (SDP) provides a principled framework for convex relaxations of nonconvex geometric constraints in motion planning, yet existing solvers are too computationally expensive for real-time control, particularly on resource-constrained embedded systems. To address this gap, we introduce TinySDP, the first semidefinite programming solver designed for embedded systems, enabling real-time model-predictive control (MPC) on microcontrollers for problems with nonconvex obstacle constraints. Our approach integrates positive-semidefinite cone projections into a cached-Riccati-based ADMM solver, leveraging computational structure for embedded tractability. We pair this solver with an *a posteriori* rank-1 certificate that converts relaxed solutions into explicit geometric guarantees at each timestep. On challenging benchmarks, e.g., cul-de-sac and dynamic obstacle avoidance scenarios that induce failures in local methods, TinySDP achieves collision-free navigation with up to 73% shorter paths than state-of-the-art baselines. We validate our approach on a Crazyflie quadrotor, demonstrating that semidefinite constraints can be enforced at real-time rates for agile embedded robotics.**

I. Introduction

Safe motion planning and control in dynamic environments remains a challenge in robotics. Model-predictive control (MPC) is attractive for this setting due to its ability to reason about system dynamics and future constraints, and MPC-based obstacle avoidance has been demonstrated on a wide range of robotic problems across numerous tasks and scenarios [21, 8, 13, 49, 36, 37, 38, 6, 2, 25, 28, 19].

Recent advances in embedded optimization have significantly expanded the scope of MPC on resource-constrained platforms [14, 24, 20], including the development of popular packages like OSQP [46], CVXGEN [35], ECOS [10], and SCS [39]. In particular, TinyMPC [38] demonstrated that constrained quadratic MPC problems can be solved at high rates on microcontrollers by exploiting the Riccati structure through a combination of offline caching and first-order methods, for efficient online operation. Building on this foundation, subsequent work [32, 31] improved the robustness of the solver and added support for second-order cone constraints, such as friction limits and thrust bounds.

Despite this progress, obstacle avoidance with formal guarantees remains an open problem in embedded MPC, particularly because obstacle constraints are inherently nonconvex. As such, existing real-time MPC implementations rely on conservative over-approximations or local convexifications that fail to capture global geometric structure. Examples of these approximations include linearized distance constraints and tangent half-spaces [44, 45, 29, 38], which, while computationally efficient, are fragile in practice. This is because solutions that satisfy such approximations can still tunnel through obstacles, requiring extensive heuristic tuning of scenario-specific safety margins, and often still fail in the presence of narrow passages, cul-de-sacs, or dynamic obstacles [53, 47].

Similarly, while barrier function based methods are also widely used for obstacle avoidance and collision avoidance, crafting effective barrier functions also relies on significant tuning, also making them fragile [1, 52, 51]. And, while Hamilton Jacobi (HJ) reachability tools can be used to construct barrier functions without requiring tuning, these tools do not scale well to high-dimensional systems [48, 7]. More recently, learning-based barrier functions leveraging neural network have been developed to overcome the dual scalability and tuning challenges. However, deploying the resulting moderate-sized neural networks for real-time applications is often infeasible on embedded platforms [9, 27].

In contrast, semidefinite programming (SDP) provides a principled way to pose nonconvex geometric constraints as convex by lifting quadratic or polynomial constraints into higher-dimensional positive semidefinite cones, and has been successfully applied in offline motion planning and trajectory optimization [22, 42, 18, 33, 34, 11, 41], globally optimal state estimation and localization [12, 17], and sum-of-squares optimization using mature toolchains such as SOSTOOLS [40]. However, as with many of the prior techniques, general-purpose SDP solvers are computationally expensive and memory-intensive. As such, they are generally considered to be poorly suited for real-time receding horizon control, particularly for embedded applications [50, 34].

In this work, we show that it is possible to overcome these challenges through careful exploitation of problem structure. As such, we introduce TinySDP, the first semidefinite programming solver designed for real-time obstacle avoidance on embedded systems. Our approach uses a per-stage, convex, semidefinite relaxation of nonconvex, quadratic, disk-based obstacle avoidance constraints. These relaxations are small, structured, and compatible with Riccati-based MPC solvers, including real-time, cached variants [38]. Crucially, we pair this relaxation with a simple rank-1 certificate that converts the

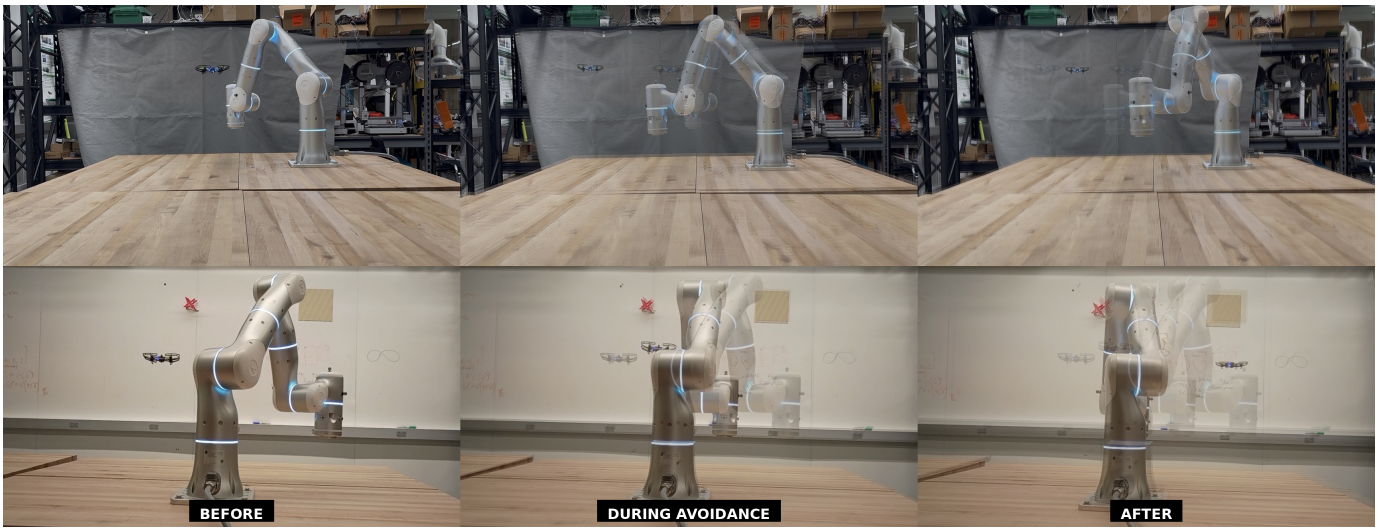

Fig. 1: Front (top) and side (bottom) views of the Crazyflie maneuvering to avoid the moving arm. From left to right: **Before**, **During**, and **After** avoidance.

relaxed solution into an explicit geometric safety guarantee at every timestep. When the certificate holds, it provably certifies that the true robot position avoids all obstacles, despite the use of an efficient convex relaxation.

Our approach builds on prior observations that lifted convex relaxations are empirically tight in structured control problems [3]. In particular, recent work has shown that the lifted semidefinite formulations of linear-quadratic control problems recover low-rank solutions, providing exact certificates of optimality [4, 30].

We evaluate TinySDP on two challenging benchmark tasks that are designed to expose common failure modes for standard formulations: 1) a static U-shaped cul-de-sac and, 2) multiple dynamic moving-gap scenarios. Through systematic comparisons, we show that our method achieves less conservative trajectories, with path lengths up to 73% shorter than baseline approaches. Importantly, we also remain certifiably collision-free at each timestep through the use of our a posteriori rank-1 safety certificate. Finally, we implement the full solver within the TinyMPC framework [38], and validate our algorithm through real-world hardware deployments on a Crazyflie 2.1 Brushless quadrotor at 25 Hz control rates, demonstrating that semidefinite constraints can be enforced at real-time rates for embedded robotic tasks.

II. Background

This section reviews prior work on embedded MPC, with emphasis on cached Riccati-based solvers and operator-splitting methods, as well as convex relaxations for nonconvex quadratic constraints. These ideas form the foundation for the method proposed in Section III.

A. Linear, Cached, and Riccati-Based MPC Solvers

We begin by writing the generic finite-horizon MPC problem and explicitly highlighting the components that prevent a closed-form solution.

Consider a discrete-time linear time-invariant system with state $x_k \in \mathbb{R}^n$ and control input $u_k \in \mathbb{R}^m$ at timestep k,

$$x_{k+1} = Ax_k + Bu_k,$$

where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ define the system dynamics.

Given a horizon length N, and assuming quadratic stage and terminal costs, for $Q \succeq 0$, $R \succ 0$, and $Q_f \succeq 0$, define

$$\ell(x_k, u_k) = x_k^\top Q x_k + u_k^\top R u_k, \quad \ell_f(x_N) = x_N^\top Q_f x_N,$$

and admissible convex state and control sets, $\mathcal{X}$,$\mathcal{U}$, the finite-horizon MPC problem is:

$$\begin{aligned} \min_{\{x_k,u_k\}} \quad & \ell_f(x_N) + \sum_{k=0}^{N-1} \ell(x_k, u_k) && \text{(1a)} \\ \text{s.t.} \quad & x_{k+1} = Ax_k + Bu_k, && \text{(1b)} \\ & u_k \in \mathcal{U}, \quad x_k \in \mathcal{X}. && \text{(1c)} \end{aligned}$$

In the absence of the constraints (1c), the problem reduces to an unconstrained linear-quadratic regulator (LQR), which admits a closed-form solution via the discrete-time Riccati recursion [26]. The optimal control law is affine,

$$u_k = -K_k x_k - d_k, \tag{2}$$

where the gain K_k and feedforward d_k are found via the backward Riccati recursion initialized with $P_N = Q_f$ and $p_N = 0$, and evaluated for $k = N-1, N-2, \ldots, 0$:

$$\begin{aligned} K_k &= (R + B^\top P_{k+1} B)^{-1} B^\top P_{k+1} A, \\ d_k &= (R + B^\top P_{k+1} B)^{-1} (B^\top p_{k+1} + r_k), \\ P_k &= Q + A^\top P_{k+1} A - A^\top P_{k+1} B K_k, \\ p_k &= q_k + A^\top p_{k+1} - A^\top P_{k+1} B d_k. \end{aligned} \tag{3}$$

However, the presence of the constraints destroys this structure and thus necessitates a numerical approach. To overcome this challenge, recent approaches have leveraged the alternating direction method of multipliers (ADMM) [5], to separate

the inequality constraints from the remainder of the problem, enabling the reintroduction of the efficient Riccati recursion. This is done by first introducing auxiliary variables, z_k, into the problem.

For clarity, and without loss of generality, we assume that the only inequality constraints in our MPC problem are on the controls, u. We can then introduce $\mathcal{I}_{\mathcal{U}}$, the indicator function of the admissible input set $\mathcal{U}$, and transform (1) into:

$$\min_{\{u_k, z_k\}} \quad \ell_f(x_N) + \sum_{k=0}^{N-1} \ell(x_k, u_k) + \mathcal{I}_{\mathcal{U}}(z_k) \tag{4a}$$

$$\text{s.t.} \quad x_{k+1} = Ax_k + Bu_k, \tag{4b}$$

$$u_k - z_k = 0. \tag{4c}$$

If we then introduce y_k as scaled dual variables and $\rho > 0$ as a penalty parameter, the augmented Lagrangian associated with the consensus constraint $u_k = z_k$ becomes:

$$\underbrace{\ell_f(x_N) + \sum_{k=0}^{N-1} \left(\ell(x_k, u_k) + \mathcal{I}_{\mathcal{U}}(z_k) + \frac{\rho}{2} \|u_k - z_k + y_k\|_2^2 \right)}_{\mathcal{L}_\rho}.$$

We then perform alternating minimization on $\mathcal{L}_\rho$ with respect to x, u, z, arriving at the three-step ADMM [5] iteration,

$$\textbf{primal update} : x^+, u^+ = \arg\min_{x,u} \mathcal{L}_P(x, u, z, y), \tag{5a}$$

$$\textbf{slack update} : z^+ = \arg\min_{z} \mathcal{L}_P(x^+, u^+, z, y), \tag{5b}$$

$$\textbf{dual update} : y^+ = y + \rho(u^+ - z^+), \tag{5c}$$

where the last step is a gradient-ascent update on the duals. These steps can be iterated until a desired convergence tolerance is achieved. Importantly, the primal problem 5a now reduces to the Riccati Recursion, the slack update is a projection, and the dual update is simply a vector addition.

The key observation that makes this whole framework efficient is that many common convex constraint sets admit simple, closed-form projection operators to solve (5b).

Cached-based solvers, like TinyMPC [38], go one step further and fix and cache key values offline that reduce the computational complexity of the primal Riccati recursion even further. These computational savings both reduce the latency of the solver as well as its memory requirements, enabling high-rate MPC on resource-constrained platforms.

In particular, for sufficiently long horizons, the Riccati recursion converges to the infinite-horizon LQR solution [26], where the time-varying matrices K_k and P_k are approximated by steady-state quantities K_∞ and P_∞. These satisfy the discrete-time algebraic Riccati equations:

$$\begin{aligned} K_\infty &= (R + B^\top P_\infty B)^{-1} B^\top P_\infty A, \\ P_\infty &= Q + A^\top P_\infty A - A^\top P_\infty B K_\infty. \end{aligned}$$

Combined with assumptions of fixed A, B, Q, R, Q_f, this drastically simplifies (3) to:

$$\begin{aligned} d_k &= C_1(B^\top p_{k+1} + r_k), \\ p_k &= q_k + C_2 p_{k+1} - K_\infty^\top r_k, \end{aligned} \tag{6}$$

where, $C_1 = (R + B^\top P_\infty B)^{-1}, C_2 = (A - BK_\infty)^\top$.

B. Semidefinite Relaxations of Quadratic Constraints

Many geometric safety constraints encountered in robotics are quadratic and nonconvex. For clarity, we first consider planar obstacle avoidance, and let $p_k \in \mathbb{R}^2$ denote the position component of the state $x_k \in \mathbb{R}^n$. Disk obstacle avoidance constraints take the form,

$$\|p_k - c_j\|_2^2 \geq r_j^2, \tag{7}$$

where $c_j \in \mathbb{R}^2$ is the center and $r_j > 0$ the radius of obstacle j. Note that Equation (7) is nonconvex. The same lifting idea extends directly to a d-dimensional geometric subspace $p_k \in \mathbb{R}^d$, with disks replaced by Euclidean balls and the lifted moment block growing accordingly.

In this paper, we adopt the standard semidefinite lifting of the geometric state. Specifically, we introduce the lifted matrix,

$$P_k = p_k p_k^\top, \tag{8}$$

which allows quadratic terms in p_k to be expressed linearly. Consistency between p_k and P_k is relaxed via the positive semidefinite constraint,

$$\begin{bmatrix} 1 & p_k^\top \\ p_k & P_k \end{bmatrix} \succeq 0, \tag{9}$$

which equivalent to (8) when (9) is rank-1. Such semidefinite relaxations are widely used in offline trajectory optimization, sum-of-squares programming, and safety verification [50, 42, 18, 33, 34, 11, 41]. However, general-purpose semidefinite programming (SDP) solvers scale poorly with problem dimension and horizon length, making them unsuitable for real-time control and embedded deployment.

III. TinySDP

Despite their success in embedded settings, Riccati-based MPC solvers are typically limited to constraint classes that preserve stage-wise structure and admit efficient proximal/projection steps (e.g., linear and second-order cone constraints). While semidefinite programming is also a form of conic optimization, it is considered incompatible with real-time MPC as general-purpose SDP methods require repeated large matrix factorizations with per-iteration costs that scale cubically in the semidefinite matrix variable dimensions.

We show that, by introducing a structured lifting that preserves the computational structure needed for efficient Riccati-based MPC, these tractability barriers can be bypassed, enabling real-time, certifiable safety for embedded robotics.

We consider the problem as defined in (1) where the inequality constraints are as follows. First, inputs are subject to box constraints:

$$u_{\min} \leq u_k \leq u_{\max}. \tag{10}$$

Second, safety is enforced via obstacle avoidance constraints on a selected geometric subspace of the state. In our deployed setting, we specialize to the planar position and let $p_k \in \mathbb{R}^2$ denote the planar position component of the state, i.e., $p_k =$

$C_p x_k$ for a known selection matrix $C_p \in \mathbb{R}^{2\times n_x}$. We model obstacles as unions of disks,

$$\mathcal{O}_{k,j} := \{p \in \mathbb{R}^2 : \|p - c_{k,j}\|_2^2 \le r_{k,j}^2\}, \tag{11}$$

and enforce safety via per-timestep keep-out constraints using (7). Such constraints are inherently nonconvex and present a central challenge for real-time MPC. Our goal is therefore to design a receding-horizon MPC controller that enforces (7) with explicit safety guarantees, while remaining compatible with embedded solvers based on Riccati recursion.

A. PSD-Relaxed Lifted MPC

1) Lifted Dynamics and Costs: Applying the semidefinite lifting of (8) to (9), we introduce auxiliary lifted second-order moment variables for the state and input outer products. Specifically, we define,

$$X_k = x_k x_k^\top,\ \ XU_k = x_k u_k^\top,\ \ UX_k = u_k x_k^\top,\ \ UU_k = u_k u_k^\top,$$

which we lift to SDP constraints analogously to (9). We define the lifted state, input, and initial condition as,

$$_k := \begin{bmatrix} x_k \\ \mathrm{vec}(X_k) \end{bmatrix},\ \bar{u}_k := \begin{bmatrix} u_k \\ \mathrm{vec}(XU_k) \\ \mathrm{vec}(UX_k) \\ \mathrm{vec}(UU_k) \end{bmatrix},\ \bar{x}_0 = \begin{bmatrix} x_0 \\ \mathrm{vec}(x_0 x_0^\top) \end{bmatrix},$$

noting that the lifted initial state is rank-consistent at initialization (as required by the rank-1 certificate introduced later).

Using the identities,

$$\mathrm{vec}(Axx^\top A^\top) = (A \otimes A)\,\mathrm{vec}(xx^\top),$$
$$\mathrm{vec}(Axu^\top B^\top) = (B \otimes A)\,\mathrm{vec}(xu^\top),$$

and expanding,

$$x_{k+1}x_{k+1}^\top = (Ax_k + Bu_k)(Ax_k + Bu_k)^\top,$$

the lifted dynamics induced by $x_{k+1} = Ax_k + Bu_k$ remain linear in the lifted variables:

$$\bar{x}_{k+1} = \bar{A}\,\bar{x}_k + \bar{B}\,\bar{u}_k, \tag{12}$$

$$\bar{A} = \begin{bmatrix} A & 0 \\ 0 & A \otimes A \end{bmatrix},\ \bar{B} = \begin{bmatrix} B & 0 & 0 & 0 \\ 0 & B \otimes A & A \otimes B & B \otimes B \end{bmatrix},$$

where $\otimes$ denotes the Kronecker product. We then define the projection matrices,

$$C_x := \begin{bmatrix} I_{n_x} & 0 \end{bmatrix} \ \text{ and } \ C_u := \begin{bmatrix} I_{n_u} & 0 & 0 & 0 \end{bmatrix},$$

enabling us to relate x_k and u_k to $\bar{x}_k$ and $\bar{u}_k$ via $x_k = C_x \bar{x}_k$ and $u_k = C_u \bar{u}_k$ respectively. The quadratic stage cost in (1) is then applied to the physical variables extracted from the lifted decision variables (with $\bar{Q} := C_x^\top Q C_x$ and $\bar{R} := C_u^\top R C_u$),

$$\ell(x_k, u_k) = \bar{x}_k^\top \bar{Q}\,\bar{x}_k + \bar{u}_k^\top \bar{R}\,\bar{u}_k. \tag{13}$$

2) Lifted Geometric Constraints: As $p_k = C_p x_k \in \mathbb{R}^2$, we can define the corresponding block of the lifted second moment matrix as follows, noting that in the exact case where $X_k = x_k x_k^\top$ this reduces to $X_k^{(p)} = p_k p_k^\top$,

$$X_k^{(p)} := C_p X_k C_p^\top \in \mathbb{R}^{2\times 2}. \tag{14}$$

While we specialize to $p_k \in \mathbb{R}^2$ for the planar obstacle-avoidance setting studied in this paper, the same construction applies to any d-dimensional geometric subspace by taking $X_k^{(p)} := C_p X_k C_p^\top \in \mathbb{R}^{d\times d}$ and replacing disks with Euclidean balls defined over the corresponding geometric coordinates.

The nonconvex disk obstacle avoidance constraints (11) can also thus be expanded as,

$$\|p_k\|_2^2 - 2c_{k,j}^\top p_k + \|c_{k,j}\|_2^2 \ge r_{k,j}^2.$$

Replacing $\|p_k\|_2^2$ with $\mathrm{trace}(X_k^{(p)})$ yields the following lifted affine inequality constraint which is linear in the lifted variables and is exact when $X_k = x_k x_k^\top$,

$$\mathrm{trace}(X_k^{(p)}) - 2c_{k,j}^\top p_k + \|c_{k,j}\|_2^2 \ge r_{k,j}^2. \tag{15}$$

3) Per-Stage PSD Constraint: We impose a per-stage positive semidefinite constraint on the augmented state–input moment matrix:

$$M_k(\bar{x}_k, \bar{u}_k) := \begin{bmatrix} 1 & x_k^\top & u_k^\top \\ x_k & X_k & XU_k \\ u_k & UX_k & UU_k \end{bmatrix} \in \mathbb{S}^{1+n_x+n_u}. \tag{16}$$

In the exact (non-relaxed) case, M_k is rank-1. Relaxing to $M_k \succeq 0$ yields a convex relaxation that permits $X_k \neq x_k x_k^\top$ (and similarly for XU_k, UX_k, UU_k) when obstacle constraints are active. In our approach, the tightness of the relaxation is assessed a posteriori using the certificate in Section III-D. The physical variables x_k and u_k remain explicit decision variables in the MPC and are not reconstructed from the lifted moment.

4) Lifted MPC Problem: The resulting MPC problem is posed over lifted variables $\{\bar{x}_k\}_{k=0}^N$ and $\{\bar{u}_k\}_{k=0}^{N-1}$ with: (i) linear lifted dynamics (12), (ii) lifted costs (13), (iii) input bounds (10) on the physical inputs u_k, (iv) lifted geometric inequalities (15), and (v) per-stage PSD constraints (16). This problem is convex but includes per-stage semidefinite constraints, which are not directly supported by existing embedded solvers. We next show how to solve this problem efficiently using ADMM while preserving Riccati structure.

B. Riccati–ADMM Solver for PSD-Relaxed Lifted MPC

We solve the PSD-relaxed lifted MPC problem using a Riccati-based ADMM, extending the TinyMPC framework [38] to incorporate per-stage semidefinite constraints while preserving its computational structure.

We introduce a PSD slack variable $S_k \in \mathbb{S}_+^p$ and a scaled dual variable $H_k \in \mathbb{S}^p$ for the coupling constraint, which are both stored using the $\sqrt{2}$-scaled half-vectorization operator $\mathrm{svec}(\cdot)$, with inverse mapping $\mathrm{smat}(\cdot)$, to further reduce the memory requirements of our approach, while ensuring that Frobenius inner products are preserved under vectorization.

We then define $M_k(\bar{x}_k, \bar{u}_k)$ as the per-stage state–input moment matrix defined in (16) and $p := 1 + n_x + n_u$,

$$M_k(\bar{x}_k, \bar{u}_k) = S_k. \tag{17}$$

Using the scaled form of ADMM, each iteration alternates between a primal update, a PSD projection, and a dual update. To simplify notation, we define the augmented stage cost,

$$\underbrace{\ell_k(\bar{x}_k, u_k) + \frac{\rho_{\text{psd}}}{2}\big\|M_k(\bar{x}_k, \bar{u}_k) - S_k + H_k\big\|_F^2}_{\tilde{\ell}_k(\bar{x}_k, u_k; S_k, H_k)}. \tag{18}$$

The ADMM updates are thus given by,

$$(\bar{x}, u)^+ = \arg\min_{\bar{x}, u} \sum_{k=0}^{N-1} \tilde{\ell}_k(\bar{x}_k, u_k; S_k, H_k), \tag{19a}$$

$$S_k^+ = \Pi_{\mathbb{S}_+^p}\big(M_k(\bar{x}_k^+, \bar{u}_k^+) + H_k\big), \tag{19b}$$

$$H_k^+ = H_k + \gamma_{\text{psd}}\big(M_k(\bar{x}_k^+, \bar{u}_k^+) - S_k^+\big). \tag{19c}$$

where $\rho_{\text{psd}} > 0$ is the penalty parameter, and $\gamma_{\text{psd}} \in (0, 1]$ is an optional under-relaxation factor. Here $\Pi_{\mathbb{S}_+^p}(\cdot)$ denotes the Euclidean projection onto the positive semidefinite cone:

$$\Pi_{\mathbb{S}_+^p}(Y) := \arg\min_{X \succeq 0} \ \|X - Y\|_F^2.$$

Note that this projection has an analytic solution,

$$\Pi_{\mathbb{S}_+^p}(Y) = V \operatorname{diag}\big(\max(\lambda, 0)\big) V^\top, \tag{20}$$

where $V \operatorname{diag}(\lambda) V^\top$ is the eigen-decomposition of Y.

1) Primal Update (Riccati Step): With the slack and dual variables fixed, the primal subproblem (19a) is an equality-constrained quadratic program with linear dynamics and the lifted quadratic stage cost defined in (13). The PSD-augmented term in Eq. (18) can be expanded as follows, where $\|\cdot\|_F$ and $\langle \cdot, \cdot \rangle_F$ denote the Frobenius norm and Frobenius inner product, respectively, and we define $T_k := S_k - H_k$:

$$\frac{\rho_{\text{psd}}}{2}\|M_k - S_k + H_k\|_F^2 = \frac{\rho_{\text{psd}}}{2}\|M_k\|_F^2 - \rho_{\text{psd}}\langle T_k, M_k \rangle_F + \text{constant}.$$

Since the M_k depends affinely on the lifted variables, the inner-product term contributes additional linear terms in $\bar{x}_k$ and $\bar{u}_k$. Thus, for fixed (S_k, H_k), the primal stage cost retains the form $\bar{x}_k^\top \bar{Q}\, \bar{x}_k + \bar{u}_k^\top \bar{R}\, \bar{u}_k + q_k^\top \bar{x}_k + r_k^\top \bar{u}_k$, where the linear coefficients q_k and r_k incorporate the adjoint pullback of the PSD term. This corresponds to the "update linear costs via adjoint mapping" step in Algorithm 1.

Crucially, the system dynamics (12) remain unchanged, so the resulting primal problem is still a lifted LQR problem. As in TinyMPC [38], it can therefore be solved efficiently by a backward-forward Riccati sweep using the cached steady-state Riccati quantities computed offline (as described in Section II).

2) PSD Projection and Dual Update: At each ADMM iteration, the slack update (19b) is computed via (20), followed by the scaled dual update (19c).

Algorithm 1 TinySDP: PSD-Relaxed Lifted MPC

function TINYSDP_OFFLINE(Lifted Model & Costs)
  Construct lifted dynamics $(\bar{A}, \bar{B})$ via (12)
  Construct lifted quadratic costs $(\bar{Q}, \bar{R})$ via (13)
  for some ADMM penalty parameter ρ **do**
    Compute and cache Riccati quantities for the lifted LQR subproblem via (3)
  end for
  return Cached solver data $\mathcal{C}$
end function

function TINYSDP_ONLINE(x_k, Obstacles $\mathcal{O}_k$, Cache $\mathcal{C}$)
  Lift current state: $\bar{x}_0 \leftarrow [x_k^\top, \text{vec}(x_k x_k^\top)^\top]^\top$
  Initialize slack and dual variables
  while residual $> \varepsilon$ and iter $< K_{\max}$ **do**
    `// Primal Update (Riccati Step)`
    $q_k, r_k \leftarrow$ Update linear costs via adjoint mapping
    $d_k, p_k \leftarrow$ Backward Riccati recursion via (6)
    $\bar{x}, \bar{u} \leftarrow$ Forward rollout via (2)
    `// PSD Projection and Dual Update`
    for $k = 0$ to $N - 1$ **do**
      Form moment matrix $M_k(\bar{x}_k, \bar{u}_k)$ via (16)
      $S_k \leftarrow$ PSD Projection via (19b)
      $H_k \leftarrow$ Dual Update via (19c)
    end for
  end while
  return $(\bar{x}_{k:k+N}, \bar{u}_{k:k+N-1})$
end function

C. Overall TinySDP Algorithm

TinySDP operates in a standard receding-horizon MPC loop. Offline, the lifted dynamics and costs are constructed, and the steady-state Riccati quantities for the lifted primal subproblem (e.g., K_∞, P_∞, and associated cached matrices) are computed offline from the steady-state Riccati equation associated with (6) and stored in a cache:

$$\mathcal{C} := \{K_\infty, P_\infty, C_1, C_2, \ldots\},$$

where the ellipsis denotes the additional cached matrices required by the backward-forward primal sweep. Online, at each control timestep k, the current measured state x_k and obstacle set $\mathcal{O}_k$ are used to solve the finite-horizon PSD-relaxed lifted MPC problem. Within each ADMM iteration, the cached infinite-horizon Riccati quantities are used to carry out the backward-forward primal sweep efficiently. The solver returns a finite-horizon state/input sequence, of which only the first control input is applied; the problem is then re-solved at the next timestep using the updated state and obstacle information. The complete procedure is summarized in Algorithm 1.

D. A Posteriori Rank-1 Safety Certificate

a) Trace Gap and Lifted Margin: For clarity, we state the certificate for the planar case $p_k \in \mathbb{R}^2$, but the same argument extends directly to any d-dimensional geometric subspace by

Algorithm 2 TinySDP with Online Certificate

function TINYSDP_STEP($x_k, \mathcal{O}_k, \mathcal{C}, u_{\text{hover}}$)
  // Solve with TinySDP
  $\bar{x}_{k:k+N}, \bar{u}_{k:k+N-1} \leftarrow$ TINYSDP_ONLINE($x_k, \mathcal{O}_k, \mathcal{C}$)
  // Verify solution quality
  $\Delta_k \leftarrow$ Compute trace gap via (21)
  $\eta_k^{\min} \leftarrow \min_j \eta_{k,j}$ via (22)
  if $\eta_k^{\min} \geq 0$ **and** $|\Delta_k| \leq \eta_k^{\min}$ **then**
    $u_k \leftarrow \bar{u}_k$ ▷ First control in optimized sequence
  else
    $u_k \leftarrow u_{\text{hover}}$ ▷ Fallback Policy (brake and hover)
  end if
  return u_k
end function

replacing $X_k^{(p)} \in \mathbb{R}^{2\times 2}$ with $X_k^{(p)} \in \mathbb{R}^{d\times d}$. Let $p_k \in \mathbb{R}^2$ be the position at time k and $X_k^{(p)} \in \mathbb{R}^{2\times 2}$ be the position block of the lifted moment matrix. We define the *trace gap* as

$$\Delta_k := \operatorname{trace}(X_k^{(p)}) - \|p_k\|_2^2. \tag{21}$$

This scalar measures the mismatch between the relaxed lifted variable and the exact rank-1 moment induced by the physical state p_k. In particular, $\Delta_k = 0$ implies that the lifted position block is consistent with the exact physical second moment, while $\Delta_k > 0$ indicates that the relaxed lifted second moment occupies more volume than the physical state. Similarly, for obstacle j with center c_j and radius r_j, the *lifted margin* is

$$\eta_{k,j} := \operatorname{trace}(X_k^{(p)}) - 2c_j^\top p_k + \|c_j\|_2^2 - r_j^2, \tag{22}$$

which corresponds to the linear obstacle-avoidance constraint enforced by the solver.

b) Certification Logic: The true squared clearance to obstacle j is given by $\delta_{k,j} := \|p_k - c_j\|_2^2 - r_j^2$. Substituting the definitions above yields the identity: $\delta_{k,j} = \eta_{k,j} - \Delta_k$. Let $\eta_k^{\min} := \min_j \eta_{k,j}$. It follows that if,

$$\eta_k^{\min} \geq 0 \quad \text{and} \quad |\Delta_k| \leq \eta_k^{\min}, \tag{23}$$

then $\delta_{k,j} \geq 0$ for all j, certifying that the state at timestep k is collision-free.

The rank-1 condition is *sufficient*, but not necessary. A certification failure occurs when either the lifted margin is negative ($\eta_k^{\min} < 0$) or the relaxation gap exceeds the available lifted margin ($|\Delta_k| > \eta_k^{\min}$). However, we note that, even when the relaxation is not exactly rank-1, this simple trace-based certificate (23) suffices to certify geometric clearance from obstacles. When the certificate fails, the relaxation may be too loose, and while the physical state p_k might still be safe, we cannot provide a mathematical guarantee.

c) Online Safety Monitor: To enhance robustness in deployed settings, we integrate a runtime safety monitor into the control loop (Algorithm 2). If the solver produces a solution that fails the trace-based certificate (23), the system discards the planned update and executes a fallback stop-and-hover policy. This ensures that uncertified control inputs are never applied to the physical system. While this monitor provides a rigorous check on the deployed state, it serves as a reactive failsafe rather than a formal closed-loop guarantee under extreme conditions, such as high-velocity obstacles or significant modeling errors.

IV. EXPERIMENTS

We designed our experiments to evaluate both static and dynamic obstacle avoidance under known challenging geometries for baseline, state-of-the-art methods. In the static setting, we consider a concave U-shaped obstacle, a cul-de-sac, with the goal located beyond the obstacle, a canonical configuration known to induce deadlock and failure modes for local and reactive methods [15, 43]. In the dynamic setting, we construct a scenario consisting of a static obstacle directly in front of the agent together with two moving obstacles that periodically close a narrow passage, resulting in a three-obstacle interaction that requires continuous updates to the obstacle avoidance strategy, rather than one-shot reactive planning. We finally deploy our approach on a Crazyflie to evaluate our method's real-world feasibility.

We compare our TinySDP method against three baselines:

- TinyMPC-LIN, which enforces obstacle avoidance using linearized tangent half-space constraints [38],
- TinyMPC-HOCBF, which enforces relative-degree-2 control barrier function constraints within TinyMPC, inspired by [52, 53], and,
- RPCBF [23], a state-of-the-art sampling-based policy safety filter implemented outside TinyMPC, but evaluated on the same system dynamics.

TABLE I: Static U-shape results for four initial conditions.

Start	Method	Path Len	Goal Dist	Safe
Inside	TinySDP (ours)	**17.95**	**0.006**	✓
	RPCBF	26.03	0.091	✓
	TinyMPC-LIN (m=3.1m)	18.38	1.400	✓
	TinyMPC-LIN (m=0m)	–	–	✗
	TinyMPC-HOCBF (m=3m)	–	–	✗
	TinyMPC-HOCBF (m=0m)	–	–	✗
Outside Center	TinySDP (ours)	**10.15**	**0.021**	✓
	RPCBF	31.03	0.093	✓
	TinyMPC-LIN (m=3.1m)	24.58	1.400	✓
	TinyMPC-LIN (m=0m)	–	–	✗
	TinyMPC-HOCBF (m=3m)	–	–	✗
	TinyMPC-HOCBF (m=0m)	–	–	✗
Edge Up	TinySDP (ours)	**9.93**	**0.023**	✓
	RPCBF	36.81	0.132	✓
	TinyMPC-LIN (m=3.1m)	18.58	1.400	✓
	TinyMPC-LIN (m=0m)	–	–	✗
	TinyMPC-HOCBF (m=3m)	–	–	✗
	TinyMPC-HOCBF (m=0m)	–	–	✗
Edge Down	TinySDP (ours)	**9.93**	**0.023**	✓
	RPCBF	36.25	0.108	✓
	TinyMPC-LIN (m=3.1m)	23.64	1.400	✓
	TinyMPC-LIN (m=0m)	–	–	✗
	TinyMPC-HOCBF (m=3m)	–	–	✗
	TinyMPC-HOCBF (m=0m)	–	–	✗

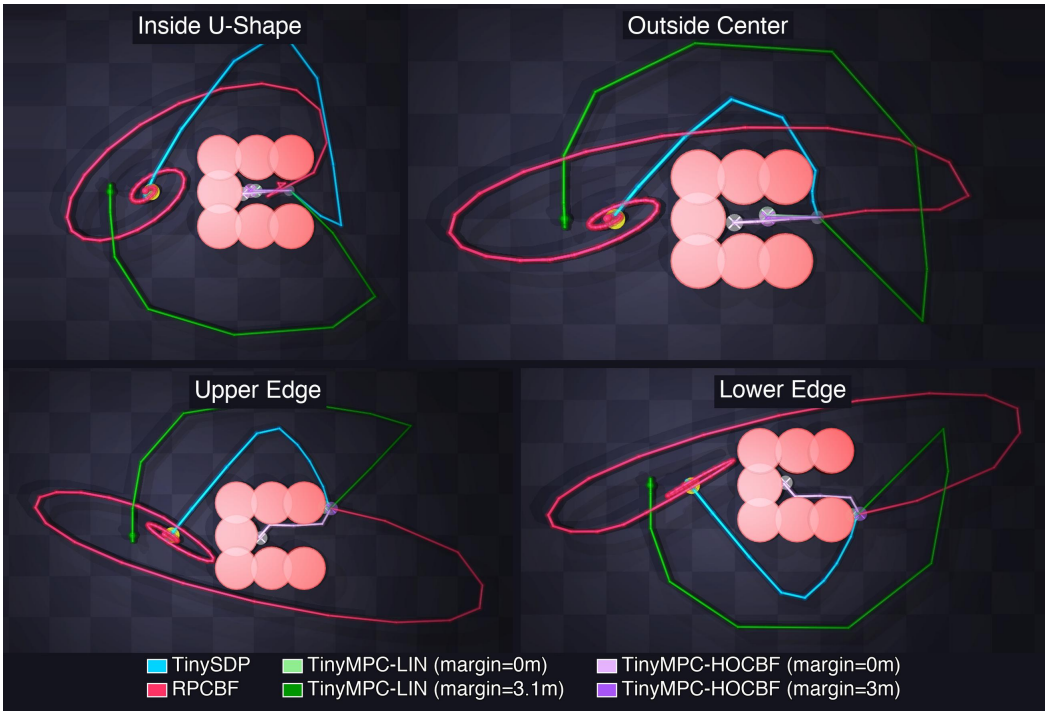

Fig. 2: **Static U-shape benchmark.** Trajectories for TinySDP (blue), RPCBF (red), TinyMPC-LIN (green), and TinyMPC-HOCBF (purple) across four initial conditions. **Key Takeaway:** TinySDP consistently navigates the cul-de-sac to reach the goal. TinyMPC-LIN (light green) and TinyMPC-HOCBF with any margin crash or get stuck. While the tuned TinyMPC-LIN (dark green) becomes safe with a $3.1\,\mathrm{m}$ margin, this inflated buffer forces the solver to terminate far from the actual goal. RPCBF is able to safely navigate the challenging scenario, but yields significantly longer, more conservative paths than TinySDP.

For all our experiments, all methods use identical discrete-time dynamics, input bounds, horizon length, and obstacle geometry, unless otherwise specified. Additionally, to ensure a fair comparison, we evaluate TinyMPC-LIN and TinyMPC-HOCBF with a zero safety margin, benchmarking against TinySDP which inherently requires no additional margin. For RPCBF, we similarly use zero safety margin but tune the underlying α parameters to achieve the best possible performance. Additionally, to provide a "best-case" analysis for the linearized baselines, we performed a grid search to determine the minimum safety margin required for TinyMPC-LIN and TinyMPC-HOCBF to successfully navigate the benchmarks. Our ablation studies reveal that these methods require a minimum safety margin of $1.5\,\mathrm{m}$ to avoid collision. We note that such margins are highly impractical for our target domain of embedded, small-scale robotic platforms, which have widths roughly equal to $100\,\mathrm{mm}$ [16]. On such platforms, a $1.5\,\mathrm{m}$ buffer artificially inflates the collision footprint by over $17\times$, effectively preventing navigation in the very environments these agile robots are designed to explore.

A. Simulation: Static U-Shape Cul-de-Sac

We first evaluate TinySDP against the baseline methods using a static U-shaped obstacle composed of overlapping disks. Four representative initial conditions are considered: starting inside the cul-de-sac, outside the center opening, near the upper edge, and near the lower edge. Figure 2 visualizes the resulting trajectories, and Table I summarizes performance across path length, distance to goal, and collision status.

TinyMPC-LIN and TinyMPC-HOCBF with a safety margin of 0 collide for every start scenario. Failures in TinyMPC-LIN arise from its reliance on nominal-dependent tangent half-space constraints, which do not impose a global geometric keep-out condition and may remain inactive until the system is already in an inevitable collision state. Similarly, TinyMPC-HOCBF enforces a continuous-time barrier condition evaluated only at discrete sampling instants; in the presence of bounded inputs and concave, trap-like geometries, this local reactive formulation fails to prevent collisions.

While TinyMPC-LIN can navigate the cul-de-sac with a safety margin of $3.1\,\mathrm{m}$, this buffer is approximately $30\times$ the Crazyflie diameter, rendering the approach impractical for real-world deployment in confined environments. Moreover, the inflated obstacle boundaries overlap with the goal region, forcing the solver to terminate approximately $1.4\,\mathrm{m}$ away from the target. TinyMPC-HOCBF fails to avoid collisions *regardless of the safety margin*, highlighting the limitations of local reactive methods in concave environments where constraint conflicts arise under finite control authority.

RPCBF [23] is the strongest baseline and also remains collision-free in all cases with a 0m margin. However, it follows a conservative strategy, producing significantly longer trajectories than TinySDP.

Overall, TinySDP achieves 31–73% shorter paths than

TABLE II: Dynamic obstacle scenario results.

Method	Path Len	Goal Dist	Safe
TinySDP (ours)	19.11	**0.018**	✓
RPCBF	27.33	0.069	✓
TinyMPC-LIN (m=1.5m)	**13.61**	0.023	✓
TinyMPC-LIN (m=0m)	–	–	✗
TinyMPC-HOCBF (m=3m)	33.80	0.077	✓
TinyMPC-HOCBF (m=0m)	–	–	✗

RPCBF and 2–59% shorter paths than the tuned TinyMPC-LIN. TinySDP also consistently reaches within $0.02\,\mathrm{m}$ of the goal across all scenarios, making it 4–$15\times$ more precise than RPCBF and $70\times$ more precise than TinyMPC-LIN.

B. Simulation: Dynamic Obstacles

We next evaluate all methods in a dynamic obstacle scenario consisting of a static disk at the start, followed by moving disks that periodically open and close a narrow passage. This task requires continuous updates to the obstacle avoidance strategy, rather than one-shot reactive planning.

Quantitative results are summarized in Table II, with representative trajectories shown in Figure 3. As illustrated, TinySDP successfully reaches the goal while maintaining positive clearance and remaining rank-1 at every timestep. In contrast, TinyMPC-LIN and TinyMPC-HOCBF with no safety margin both collide early, as local linearizations and discrete-time barrier evaluations are insufficient for fast-moving geometries. While these methods can be made safe using margins of $1.5\,\mathrm{m}$ and $3\,\mathrm{m}$ respectively, such values are 17–$30\times$ the size of the drone and are therefore impractical for real dynamic environments with narrow corridors. RPCBF [23] again remains collision-free, but due to its conservative strategy, yields significantly longer paths.

Quantitatively, TinySDP achieves 30% shorter paths than RPCBF and 43% shorter paths than the tuned TinyMPC-HOCBF ($3\,\mathrm{m}$ margin). While the tuned TinyMPC-LIN ($1.5\,\mathrm{m}$ margin) produces the shortest path at $13.61\,\mathrm{m}$, TinySDP converges $1.3\times$ closer to the goal ($0.018\,\mathrm{m}$ vs $0.023\,\mathrm{m}$) and is 3.8–$4.3\times$ more precise than RPCBF and TinyMPC-HOCBF, respectively. The shorter path length of the tuned TinyMPC-LIN could be attributed to its specific trajectory, which curves from below and arrives at the intersection only after the moving obstacle has passed. This timing allows the solver to effectively, and luckily, bypass the active avoidance maneuver. In contrast, TinySDP actively deviates to negotiate the obstacle while it is still blocking the path, ensuring certified safety at the cost of a slightly longer trajectory, and also more generality for diverse real-world scenarios.

C. Simulation: Extension to 3D Obstacle Avoidance Demos

As noted earlier, the proposed lifting is not inherently limited to planar obstacle avoidance. In a 3D setting, lifting the position state $[1, x, y, z]^\top$ yields a 4×4 moment matrix, so the same lifted semidefinite obstacle-avoidance formulation and applied-state certificate structure used in the planar case extend naturally to spherical obstacles. To illustrate this extension, we consider two dynamic 3D scenarios: *Sweeping Barrier* and *Vertical Gate*. The same construction naturally extends to other quadratic sets such as ellipsoids and cylinders as well. Of course, these richer geometric formulations require larger PSD cones and therefore more expensive projections, inducing tighter embedded compute and memory tradeoffs.

In *Sweeping Barrier*, two moving spheres sweep laterally across the route while a third upstream sphere blocks the direct start-to-goal path, requiring an early nonplanar deviation rather than a purely reactive planar sidestep. In *Vertical Gate*, two static side spheres together with a centrally located sphere oscillating in height create a time-varying 3D passage, forcing a timed ascent-descent maneuver rather than a planar bypass. Representative trajectories are shown in Figure 4. In both scenarios, TinySDP reaches the goal without collisions.

D. Real World Deployment: Dynamic Obstacles

To validate real-world feasibility on resource-constrained platforms, we deploy TinySDP on a Crazyflie 2.1 Brushless quadrotor, powered by an STM32F405 microcontroller (168 MHz with only 192 KB of SRAM). We apply the semidefinite relaxation strictly to the planar position of the quadrotor's 12-dimensional state, (p_x, p_y), yielding a 3×3 moment matrix,

$$M_k = \begin{bmatrix} 1 & p_x & p_y \\ p_x & p_x^2 & p_x p_y \\ p_y & p_x p_y & p_y^2 \end{bmatrix},$$

where the PSD constraint $M_k \succeq 0$ is enforced via projection within the ADMM solver. Since the nonconvex obstacle constraints depend only on planar position, this corresponds to the minimal task-induced lift sufficient to certify geometric clearance. The remaining state variables (altitude, velocity, attitude) are handled via box constraints as in prior work [38].

The solver runs onboard at 25 Hz with a 20-step horizon. In the deployed implementation, ADMM is capped at 5 iterations per control update to guarantee timing. In the latest logged hardware run, the solver executed all 5 iterations and reported a total MPC solve time of $14.842\,\mathrm{ms}$, leaving approximately $25.2\,\mathrm{ms}$ of margin within the $40\,\mathrm{ms}$ control period. The compiled firmware footprint is $272.4\,\mathrm{KB}$ of flash and $139.0\,\mathrm{KB}$ of on-chip RAM ($83.5\,\mathrm{KB}$ main SRAM and $55.6\,\mathrm{KB}$ CCM). The PSD-specific overhead is modest, adding approximately $1.47\,\mathrm{KB}$ of fixed static memory, about $720\,\mathrm{B}$ of peak additional stack usage along the PSD projection path, and an estimated 2.5–$3.4\,\mathrm{KB}$ of flash. Since the lifted variable is only a symmetric 3×3 matrix over $[1, p_x, p_y]$, the PSD projection is implemented using a specialized small-matrix eigendecomposition with eigenvalue clamping and is not a practical bottleneck in this planar setting.

A low-level PID controller tracks the planned trajectory at 500 Hz. We validate the approach in a dynamic obstacle avoidance scenario where the Crazyflie flies 1 m forward while a robotic arm sweeps horizontally across its path. As shown in Figure 1, the drone successfully anticipates the moving obstacle, autonomously deviating to maintain safety before converging to the goal. Notably, as shown in Figure 5, the online safety certificate (Section III-D) remained valid[1]

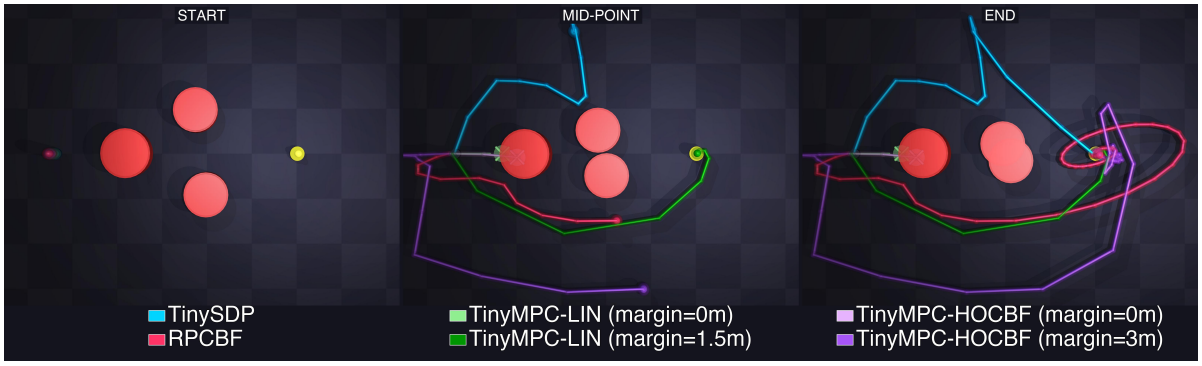

Fig. 3: **Dynamic moving-gap benchmark.** Time-lapse comparison of TinySDP (blue) against RPCBF (red) and the linearized baselines (green, purple). **Key Takeaway:** TinySDP anticipates the moving obstacle, deviating early to maintain clearance. The zero-margin baselines (light green, light purple) fail to anticipate the motion and collide. The tuned baselines (dark green, dark purple) achieve safety only by using impractically large margins ($1.5\,\mathrm{m}$ and $3\,\mathrm{m}$ which are 17–$30\times$ the size of the drone).

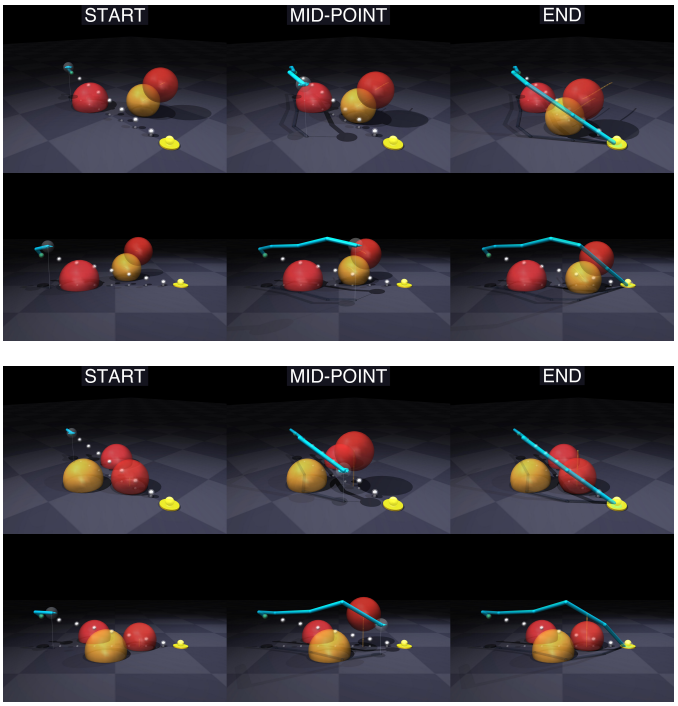

Fig. 4: **3D dynamic obstacle avoidance extension.** TinySDP navigating two dynamic 3D sphere scenarios: *Sweeping Barrier* (top) and *Vertical Gate* (bottom). The two rows provide complementary views of the same 3D motion, while the columns show the approach, midpoint, and end of the maneuver. The white dotted path indicates the direct start-to-goal line, which would intersect the obstacle field. **Key Takeaway:** TinySDP extends naturally to 3D sphere avoidance, executing nonplanar maneuvers with positive clearance in both scenarios.

($\Delta_k \approx 0$) at every control step throughout the flight for five independent trials, confirming that the rank-1 geometry was preserved even under real-world disturbances.

[1]The certificate is evaluated with a small numerical tolerance ($\varepsilon = 10^{-2}$) to account for finite-precision arithmetic on the embedded hardware platform.

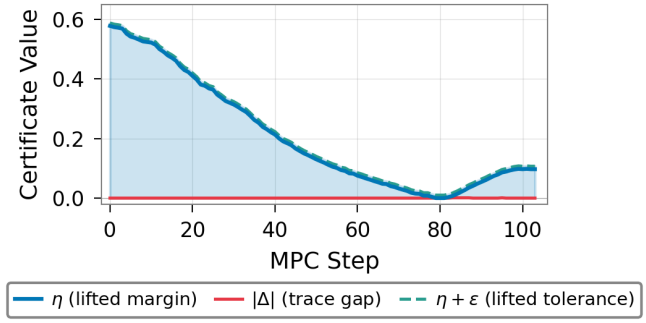

Fig. 5: Evolution of the Rank-1 certificate as the Crazyflie avoided the moving arm, demonstrating certifiable safety (shaded region) during real-world operation.

V. Limitations

While TinySDP achieved zero failures across all hardware trials, several limitations warrant discussion. Most fundamentally, our trace-gap certificate provides a rigorous per-timestep geometric check on the applied state, but does not constitute a formal closed-loop guarantee. In particular, we do not prove recursive feasibility over the full predicted horizon, although offline analysis confirms horizon-wide certificate satisfaction on our nominal trajectories, this remains an empirical observation rather than a property of the closed-loop system itself. While simulations show our lifting framework extends naturally to 3D spherical obstacles, onboard deployment reveals a computational tradeoff. The 3x3 moment matrix used in our planar hardware demonstration keeps PSD projections tractable; however, higher dimensions or complex geometries expand the lifted block, driving up per-iteration eigendecomposition costs. Transitioning this full 3D capability to microcontrollers remains a key focus for future work, necessitating careful co-design of the SDP solver against strict embedded memory and compute constraints. Finally, as established in our discussion of the online safety monitor III-D, the fallback policy acts solely as a reactive failsafe. Extreme conditions such as sufficiently fast-moving obstacles,

large modeling errors, or severe external disturbances could in principle outpace this fallback. Extending the framework with forward reachability analysis to provide proactive rather than purely reactive recovery guarantees is a natural next step.

VI. Conclusions and Future Work

We introduced TinySDP, a PSD-relaxed lifted MPC framework that enables certifiable obstacle avoidance on embedded platforms while preserving cached-Riccati-based computational structure. By integrating per-stage semidefinite lifting with an ADMM-based solver, we establish a methodological pathway for incorporating semidefinite relaxations into embedded MPC. Furthermore, by exploiting per-stage structure and separating optimization from certification, we show that SDP-based safety constraints are not inherently incompatible with real-time receding-horizon control. This perspective bridges the gap between offline semidefinite planning and practical embedded deployment, and suggests that richer convex safety models can be integrated into high-rate controllers when solver design respects MPC structure. Future work includes extending the framework to more challenging hardware demonstrations and richer obstacle representations to identify where our relaxations are loose. More broadly, we hope this work motivates the development of more low-compute and low-memory footprint solvers for real-time control.

Acknowledgments

We thank Roy Xing, Gabriel Bravo, Moises Mata, and Charles Chen for help with hardware experiments. This work was supported by the National Science Foundation (under Awards 2144634, 2231350, and 2411369) and by the Center of AI Technology (CAIT) in collaboration with Amazon. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the funding organizations.

References

[1] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2016.

[2] Manel Ammour, Rodolfo Orjuela, and Michel Basset. A mpc combined decision making and trajectory planning for autonomous vehicle collision avoidance. *IEEE Transactions on Intelligent Transportation Systems*, 23 (12):24805–24817, 2022.

[3] Bassam Bamieh. Linear-quadratic problems in systems and controls via covariance representations and linear-conic duality: Finite-horizon case. *arXiv preprint arXiv:2401.01422*, 2024.

[4] Nicolas Boumal, Vladislav Voroninski, and Afonso S Bandeira. Deterministic guarantees for burer-monteiro factorizations of smooth semidefinite programs. *Communications on Pure and Applied Mathematics*, 73(3): 581–608, 2020.

[5] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[6] Jia-Ruei Chiu, Jean-Pierre Sleiman, Mayank Mittal, Farbod Farshidian, and Marco Hutter. A collision-free mpc for whole-body dynamic locomotion and manipulation. In *2022 international conference on robotics and automation (ICRA)*, pages 4686–4693. IEEE, 2022.

[7] Jason J Choi, Donggun Lee, Koushil Sreenath, Claire J Tomlin, and Sylvia L Herbert. Robust control barrier–value functions for safety-critical control. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 6814–6821. IEEE, 2021.

[8] Mark L Darby and Michael Nikolaou. Mpc: Current practice and challenges. *Control Engineering Practice*, 20(4):328–342, 2012.

[9] Charles Dawson, Zengyi Qin, Sicun Gao, and Chuchu Fan. Safe nonlinear control using robust neural lyapunov-barrier functions. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1724–1735. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/dawson22a.html.

[10] A. Domahidi, E. Chu, and S. Boyd. ECOS: An SOCP solver for embedded systems. In *IEEE European Control Conference (ECC)*, 2013.

[11] Shiying Dong, Zhipeng Shen, Rudolf Reiter, Hailong Huang, Bingzhao Gao, Hong Chen, and Wen-Hua Chen. A fast semidefinite convex relaxation for optimal control problems with spatio-temporal constraints. *arXiv preprint arXiv:2601.03055*, 2026.

[12] Frederike Dümbgen, Connor Holmes, Ben Agro, and Timothy Barfoot. Toward globally optimal state estimation using automatically tightened semidefinite relaxations. *IEEE Transactions on Robotics*, 40:4338–4358, 2024.

[13] Davide Falanga, Philipp Foehn, Peng Lu, and Davide Scaramuzza. Pampc: Perception-aware model predictive control for quadrotors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

[14] Hand Joachim Ferreau, Stefan Almér, Robin Verschueren, Moritz Diehl, Damian Frick, Alexander Domahidi, JL Jerez, Giorgos Stathopoulos, and Colin Jones. Embedded optimization methods for industrial automatic control. *IFAC-PapersOnLine*, 50(1):13194–13209, 2017.

[15] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE robotics & automation magazine*, 4(1):23–33, 2002.

[16] Wojciech Giernacki, Mateusz Skwierczyński, Wojciech Witwicki, Paweł Wroński, and Piotr Kozierski. Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering. In *2017 22nd interna-*

tional conference on methods and models in automation and robotics (MMAR), pages 37–42. IEEE, 2017.

[17] Antoine Groudiev, Fabian Schramm, Éloïse Berthier, Justin Carpentier, and Frederike Dümbgen. Sampling-based global optimal control and estimation via semidefinite programming. *arXiv preprint arXiv:2507.17572*, 2025.

[18] Didier Henrion and Andrea Garulli. *Positive polynomials in control*, volume 312. Springer Science & Business Media, 2005.

[19] Brian E Jackson, Tarun Punnoose, Daniel Neamati, Kevin Tracy, Rianna Jitosho, and Zachary Manchester. Altro-c: A fast solver for conic model-predictive control. In *IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China, May 2021.

[20] Juan L Jerez, Paul J Goulart, Stefan Richter, George A Constantinides, Eric C Kerrigan, and Manfred Morari. Embedded online optimization for model predictive control at megahertz rates. *IEEE Transactions on Automatic Control*, 59(12):3238–3251, 2014.

[21] Rudolf E. Kálmán. Contributions to the theory of optimal control. 1960. URL https://api.semanticscholar.org/CorpusID:845554.

[22] Shucheng Kang, Guorui Liu, and Heng Yang. Global contact-rich planning with sparsity-rich semidefinite relaxations. In *Robotics: Science and Systems (RSS)*, 2025.

[23] Luzia Knoedler, Oswin So, Ji Yin, Mitchell Black, Zachary Serlin, Panagiotis Tsiotras, Javier Alonso-Mora, and Chuchu Fan. Safety on the fly: Constructing robust safety filters via policy control barrier functions at runtime. *IEEE Robotics and Automation Letters*, 2025.

[24] Dimitris Kouzoupis, Hans Joachim Ferreau, Helfried Peyrl, and Moritz Diehl. First-order methods in embedded nonlinear model predictive control. In *2015 European Control Conference (ECC)*, pages 2617–2622. IEEE, 2015.

[25] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous robots*, 40:429–455, 2016.

[26] Frank L. Lewis, Draguna Vrabie, and V.L. Syrmos. Optimal Control, 1 2012.

[27] Simin Liu, Changliu Liu, and John Dolan. Safe control under input limits with neural control barrier functions. In *Conference on Robot Learning*, pages 1970–1980. PMLR, 2023.

[28] Xinfu Liu, Zuojun Shen, and Ping Lu. Entry trajectory optimization by second-order cone programming. *Journal of Guidance, Control, and Dynamics*, 39(2):227–241, 2016.

[29] Tomas Lozano-Perez. Spatial planning: A configuration space approach. *IEEE transactions on computers*, 32(02): 108–120, 1983.

[30] Cong Ma, Kaizheng Wang, Yuejie Chi, and Yuxin Chen. Implicit regularization in nonconvex statistical estimation: Gradient descent converges linearly for phase retrieval and matrix completion. In *International Conference on Machine Learning*, pages 3345–3354. PMLR, 2018.

[31] Ishaan Mahajan and Brian Plancher. Robust and efficient embedded convex optimization through first-order adaptive caching. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025.

[32] Ishaan Mahajan, Khai Nguyen, Sam Schoedel, Elakhya Nedumaran, Moises Mata, Brian Plancher, and Zachary Manchester. Code generation and conic constraints for model-predictive control on microcontrollers with conic-tinympc, 2025.

[33] Anirudha Majumdar and Russ Tedrake. Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36(8):947–982, 2017.

[34] Tobia Marcucci, Mark Petersen, David von Wrangel, and Russ Tedrake. Motion planning around obstacles with convex optimization. *Science robotics*, 8(84):eadf7843, 2023.

[35] Jacob Mattingley and Stephen Boyd. Cvxgen: A code generator for embedded convex optimization. *Optimization and Engineering*, 13:1–27, 2012.

[36] KN McGuire, Christophe De Wagter, Karl Tuyls, HJ Kappen, and Guido CHE de Croon. Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. *Science Robotics*, 4 (35):eaaw9710, 2019.

[37] Kunal S Narkhede, Abhijeet M Kulkarni, Dhruv A Thanki, and Ioannis Poulakakis. A sequential mpc approach to reactive planning for bipedal robots using safe corridors in highly cluttered environments. *IEEE Robotics and Automation Letters*, 7(4):11831–11838, 2022.

[38] Khai Nguyen, Sam Schoedel, Anoushka Alavilli, Brian Plancher, and Zachary Manchester. Tinympc: Model-predictive control on resource-constrained microcontrollers. In *IEEE International Conference on Robotics and Automation (ICRA)*, Yokohama, Japan, May. 2024.

[39] Brendan O'Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, June 2016.

[40] Antonis Papachristodoulou, James Anderson, Giorgio Valmorbida, Stephen Prajna, Pete Seiler, Pablo Parrilo, Matthew M Peet, and Declan Jagt. Sostools version 4.00 sum of squares optimization toolbox for matlab. *arXiv preprint arXiv:1310.4716*, 2013.

[41] Alan Papalia. Introduction to certifiable robotics, 2025. URL https://github.com/alanpapalia/Intro-Certifiable-Robotics.

[42] Pablo A Parrilo. Exploiting algebraic structure in sum of squares programs. In *Positive polynomials in control*,

pages 181–194. Springer, 2005.

[43] Vignesh Rajagopal, Kasun Weerakoon Kulathun Mudiyanselage, Gershom Devake Seneviratne, Pon Aswin Sankaralingam, Mohamed Elnoor, Jing Liang, Rohan Chandra, and Dinesh Manocha. Dr. nav: Semantic-geometric representations for proactive dead-end recovery and navigation. *arXiv preprint arXiv:2511.12778*, 2025.

[44] Arthur Richards and Jonathan P How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proceedings of the 2002 American control conference (IEEE Cat. No. CH37301)*, volume 3, pages 1936–1941. IEEE, 2002.

[45] Tom Schouwenaars, Bart De Moor, Eric Feron, and Jonathan How. Mixed integer programming for multi-vehicle path planning. In *2001 European control conference (ECC)*, pages 2603–2608. IEEE, 2001.

[46] Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. Osqp: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020. ISSN 1867-2957. doi: 10.1007/s12532-020-00179-2.

[47] Akshay Thirugnanam, Jun Zeng, and Koushil Sreenath. Safety-critical control and planning for obstacle avoidance between polytopes with control barrier functions. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 286–292, 2022. doi: 10.1109/ICRA46639.2022.9812334.

[48] Sander Tonkens and Sylvia Herbert. Refining control barrier functions through hamilton-jacobi reachability. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13355–13362. IEEE, 2022.

[49] Guillem Torrente, Elia Kaufmann, Philipp Föhn, and Davide Scaramuzza. Data-driven mpc for quadrotors. *IEEE Robotics and Automation Letters*, 2021.

[50] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996.

[51] Peter Wieland and Frank Allgöwer. Constructive safety using control barrier functions. *IFAC Proceedings Volumes*, 40(12):462–467, 2007.

[52] Wei Xiao and Calin Belta. High-order control barrier functions. *IEEE Transactions on Automatic Control*, 67 (7):3655–3662, 2021.

[53] Jun Zeng, Bike Zhang, and Koushil Sreenath. Safety-critical model predictive control with discrete-time control barrier function. In *2021 American Control Conference (ACC)*, pages 3882–3889, 2021. doi: 10.23919/ACC50511.2021.9483029.