Underactuated Trajectory Tracking with Control Lyapunov Functions

Delia Stephens

***Abstract*—Combining the concepts of control-Lyapunov Functions (CLFs) and trajectory optimization, we demonstrate the usefulness of modified CLFs to track the long-term behavior of a highly constrained system. The Robust mCLF algorithm improves the trajectory tracking performance of an optimal trajectory in the presence of disturbances. Relaxing the constraints on the system by improving control authority increases the performance of the Robust mCLF; removing them entirely creates a trajectory tracking system which converges asymptotically to the optimal trajectory. The viability of the Robust mCLF algorithm is demonstrated on a special instantiation of a Dubins car with no velocity or yaw rate authority restricted by a minimum turn radius, which mimics the performance of an underwater glider or a high-altitude, long endurance aircraft.**

I. Introduction

TRAJECTORY optimization allows us to find optimal control inputs to guide a system from an initial state to a final state, subject to its dynamics constraints. However, computing this optimal trajectory is computationally costly, requiring mathematical optimization techniques that take time, and the local stability of this optimal trajectory is unverified. For this reason, it becomes valuable to develop an approach for trajectory stabilization, or trajectory tracking, for nonlinear systems under uncertainty. Typical approaches include the Linear Quadratic Regulator (LQR), which linearizes along the trajectory in the error coordinates, and Model-Predictive Control (MPC), which measures the current state, generates an optimal trajectory, executes the first action, and lets the dynamics evolve. Where LQR is particularly useful for certifying that trajectories that begin close to the planned trajectory will stay near the planned trajectory, they rely on linearization, which can yield sub-ideal results when trajectories diverge early. MPC is computationally expensive; however, receding-horizon, Linear MPC is considered the de-facto generalization of LQR for controlling a *linear* system subject to linear constraints.

Here, we'll explore the usefulness of control-Lyapunov functions (CLFs) as methods of tracking trajectories for a simple nonlinear system—the Dubins car as outlined by [1]. Dubins cars are useful for simulating systems like high-altitude, long-endurance aircraft and underwater gliders, which are often constrained in their control inputs and highly dependent on efficient energy expenditure to perform their ultra-long-duration flights.

In particular, we examine a highly-constrained Dubins car moving forward with a constant velocity. The vehicle's single control input, yaw rate, is limited by the minimum turn radius. Our goal is to control the vehicle to a manifold, $\mathcal{C}$, which is a circle of minimum turn radius, r, around a desired final position. While it is impossible to stabilize this vehicle to stop on a desired final state—the vehicle will always continue past its destination—employing control-Lyapunov functions to guide the vehicle along an optimal trajectory was highly effective for tracking the long-term behavior of the system to $\mathcal{C}$. Relaxing the control constraints (e.g., decreasing the minimum turn radius or permitting velocity control) increases the system stability, and, in turn, increases the performance of a modified CLF tracking a reference optimal trajectory.

II. The Dubins Car

We consider a specific instantiation of the Dubins Car given by:

$$f(\mathbf{x}, u) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ u_1 \end{bmatrix} \tag{1}$$

Here, the vehicle's position in the Cartesian coordinate system is given by (x, y), and θ is the vehicle's heading. The yaw rate, u_1, is the only control for the vehicle. Note that the vehicle's velocity is not controllable, making this a specific class of Dubins vehicle that cannot stop; instead, our desired end state will be an orbit around some final position. We furthermore constrain the yaw rate, u_1 by a fixed minimum turn radius, r, such that:

$$-\frac{1}{r} = -u_{\max} \leq u \leq u_{\max} = \frac{1}{r}.$$

Analysis of this system was first (and more extensively) performed in [1]; here, I've chosn to analyze the portions relevant to the trajectory optimization comparison.

A. UAV-Based Coordinates

Our desired end state is to control the aircraft to a manifold, $\mathcal{C}$, where $\mathcal{C}$ is given by:

$$\mathcal{C} = \{(x, y, \theta) | x = r\sin\theta, y = -r\cos\theta\}$$

It becomes advantageous to define UAV-based coordinates to further simplify the analysis in later sections. These coordinates, and the new dynamics may be written as:

$$\tilde{\mathbf{x}} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2}$$

$$\dot{\tilde{\mathbf{x}}} = \begin{bmatrix} u\tilde{y} + 1 \\ -u\tilde{x} \end{bmatrix} \tag{3}$$

Similarly, the new manifold $\tilde{\mathcal{C}}$ is given by

$$\tilde{\mathcal{C}} = \{(\tilde{x}, \tilde{y}, \theta) | \tilde{x} = 0, \tilde{y} = -r\}$$

We perform one final coordinate transform to stabilize to a submanifold given by:

$$\bar{\mathcal{C}} = \{(\bar{x}, \bar{y}, \theta) | x = 0, y = 0\}$$

Our final dynamics equations, which we will use for the Lyapunov analysis, are as follows:

$$\bar{\mathbf{x}} = \begin{bmatrix} \tilde{x} \\ \tilde{y} + r \end{bmatrix} \tag{4}$$

$$\dot{\bar{\mathbf{x}}} = \begin{bmatrix} u\bar{y} + 1 - ru \\ -u\bar{x} \end{bmatrix} \tag{5}$$

III. Control Lyapunov Approach

A control-Lyapunov function V is a positive-definite function that decreases the V for all $\mathbf{x}$. Simply put, the control-Lyapunov function allows us to pick a $\mathbf{u}(\mathbf{x})$ that guides our system to the desired end state. A mathematical formulation of the control-Lyapunov function (CLF) is as follows:

$$\forall \mathbf{x} \neq 0, \exists \mathbf{u} \quad \dot{V}(\mathbf{x}, \mathbf{u}) = \frac{\partial V}{\partial \mathbf{x}} f(\mathbf{x}, \mathbf{u}) < 0$$
$$\text{and} \quad \exists \mathbf{u} \quad \dot{V}(0, \mathbf{u}) = 0.$$

A. Lyapunov Analysis

We choose to perform our Lyapunov Analysis in the UAV-centric, shifted coordinates, $(\bar{x}, \bar{y})$. Our candidate Lyapunov function is as follows:

$$V(\bar{x}, \bar{y}) = \bar{x}^2 + \bar{y}^2$$

In order for our candidate Lyapunov function to be a valid Lyapunov function, we need

$$\dot{V}(\bar{x}, \bar{y}) = \frac{\partial V}{\partial \bar{\mathbf{x}}} f(\bar{x}) = 2\bar{x}(1 - ru) \leq 0 \tag{6}$$

B. Control Lyapunov Function

We now choose feedback control, $k \in \mathbb{R}^2 \to [-u_{\max}, u_{\max}]$ and $u(t) = \bar{k}(\bar{x}(t), \bar{y}(t))$, which makes Condition 6 true, stabilizing the system to the desired manifold. Remembering that $u_{\max} = \frac{1}{r}$, Condition 6 yields $u = u_{\max}$ when $\bar{x} \geq 0$. Additionally, our feedback, $\bar{k}$ is a smooth feedback control which satisfies the following function:

$$\bar{k}(\bar{x}, \bar{y}) = \begin{cases} u_{\max} & \bar{x} \geq 0 \\ -u_{\max} \leq \bar{k}(\bar{x}, \bar{y}) < -u_{\max} & \bar{x} < 0 \end{cases}$$

Here, we will define a few parameters as outlined in [1]. Let ϵ be a positive real number, and a be a number such that $-u_{\max} \leq a < -u_{\max}$. An example of one such function which satisfies the "smooth" requirement is visible in Figure 1.

$$\bar{k}(\bar{x}, \bar{y}) = \begin{cases} a & \bar{x} \leq -\epsilon \\ \frac{u_{\max} - a}{1 + e^{(1/(\bar{x}+\epsilon)) + 1/\bar{x}}} + a & \bar{x} \in (\epsilon, 0) \\ u_{\max} & \bar{x} \geq 0 \end{cases}$$

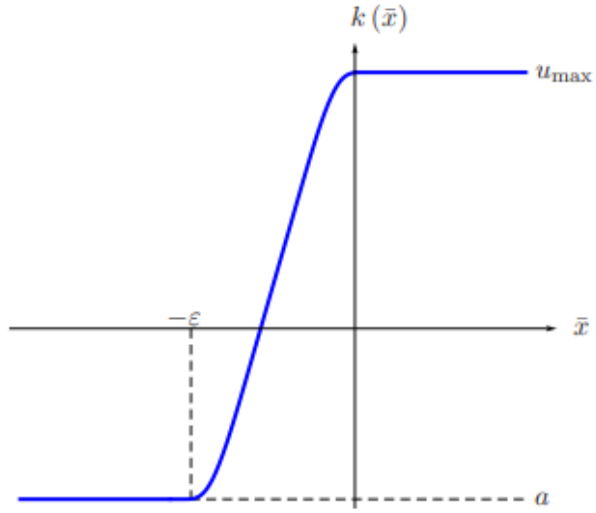

Fig. 1. Control input scheme for CLF as outlined in [1].

C. Stability Analysis

LaSalle's invariance principle states that for a scalar function $V(x)$ with continuous derivatives, $V(\mathbf{x}) \succ 0, \quad \dot{V}(\mathbf{x}) \preceq 0$, and $V(\mathbf{x}) \to \infty$ as $||\mathbf{x}|| \to \infty$, then $\mathbf{x}$ will converge to the largest invariant set where $\dot{V}(\mathbf{x}) = 0$. The following analysis closely mirrors the work from [1] and [2]. Because of the input limitations imposed on our control, it is evident that the solution to 6 starting at some initial position $\bar{x}_0, \bar{y}_0$ with $\bar{x}_0 > 0$ escapes from the invariant set $E = \{(\bar{x}, \bar{y}) | \dot{V}(\bar{x}, \bar{y}) = 0\} = \{(\bar{x}, \bar{y}) | \bar{x} \geq 0\}$ in finite time.

When $x_0 > 0$, the solution to the closed-loop system satisfied on E:

$$\begin{cases} \dot{\bar{x}} = u_{\max}\bar{y} \\ \dot{\bar{y}} = -u_{\max}\bar{x} \end{cases}$$

This solution escapes from E in time

$$t = \begin{cases} \frac{-1}{u_{\max}} \arctan\left(\frac{\bar{x}_0}{\bar{y}_0}\right) & \text{if } \bar{y}_0 < 0 \\ \frac{\pi}{2u_{\max}} & \text{if } \bar{y}_0 = 0 \\ \frac{1}{u_{\max}}(\pi - \arctan\left(\frac{\bar{x}_0}{\bar{y}_0}\right)) & \text{if } \bar{y}_0 > 0 \end{cases}$$

The largest invariant set contained in E is the equilibrium point $(0, 0)$. In the error coordinates as outlined in Section III-D, the equlibrium point is (x_1, x_2). Thus, our control input steers our system asymptotically to $\bar{\mathcal{C}}$.

D. Error Coordinates

To navigate our Dubins vehicle to an arbitrary manifold centered around the point (x_1, x_2), it becomes helpful to define UAV-centered error coordinates as given by:

$$\tilde{\mathbf{x}} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - x_1 \\ y - x_2 \\ \theta - \theta_{\text{ref}} \end{bmatrix} \tag{7}$$

$$\dot{\tilde{\mathbf{x}}} = \begin{bmatrix} u\tilde{y} + 1 \\ -u\tilde{x} \\ \dot{\theta}_e \end{bmatrix} \tag{8}$$

where $x_{1,e} = x - x_1$, $x_{2,e} = y - x_2$, and $\theta_e = \theta - \theta_{\text{ref}}$. The dynamics equations do not change because $\tilde{y}$ considers the error. In this way, our tracking problem becomes a stabilization problem to the origin, with our final manifold $\bar{\mathcal{C}}$ remaining unchanged. Our original Lyapunov analysis is valid. This was observed experimentally; it was possible to command the UAV to arbitrary (x_1, x_2) positions on the map.

E. Experimental Results

The performance of our CLF to a final manifold of radius 2 is verified below.

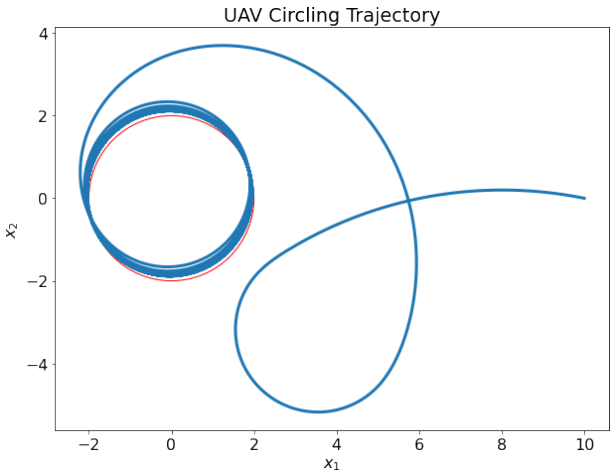

Fig. 2. Lyapunov controller with $\mathbf{x}_0 = \begin{bmatrix}10 & 1 & \pi - 0.2\end{bmatrix}^T$.

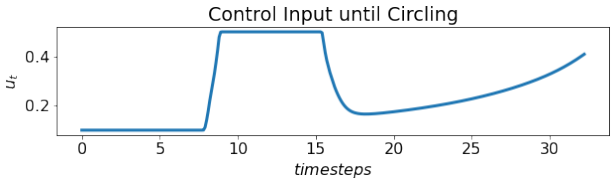

Fig. 3. Control input of CLF with $\mathbf{x}_0 = \begin{bmatrix}10 & 1 & \pi - 0.2\end{bmatrix}^T$.

IV. Trajectory Optimization

The CLF approach is capable of stabilizing the Dubins vehicle to the final manifold, but it is far from optimal; in cases where the systems are highly sensitive to cost, developing an optimal path is essential. We use a direct transcription approach to perform our nonlinear trajectory optimization. The trajectory optimization can be formulated as follows:

$$\begin{aligned} \min_{\mathbf{x}[\cdot],\mathbf{u}[\cdot]} \quad & \ell_f(\mathbf{x}[N]) + \sum_{n_0}^{N-1} \ell(\mathbf{x}[n], \mathbf{u}[n]) \\ \text{subject to} \quad & \mathbf{x}[n+1] = \mathbf{f}(\mathbf{x}[n], \mathbf{u}[n]), \quad \forall n \in [0, N-1] \\ & \mathbf{x}[0] = \mathbf{x}_0 \\ & + \text{additional constraints.} \end{aligned}$$

We discretize our continuous dynamics using an implicit Euler formulation. Our "initial guess" is a straight line from the starting position, x_0, to the *bottom* point on the circle of radius r centered around $x_f = (x_1, x_2)$, which is the center of the manifold. Rather than control to a final state, we formulate the trajectory optimization with a cost that increases with the distance from the final manifold, as follows:

$$\ell(\mathbf{x}[n], \mathbf{u}[n]) = (x[n] - x_1)^2 + (y[n] - x_1)^2 - r^2$$

We also add control and time costs in order to get the solution that minimizes the time and control. Additional constraints, in the form of the control constraints outlined in Section II, are enforced at each time step, t.

Figure 4 shows the optimal trajectory from an initial position of $\mathbf{x}_0 = \begin{bmatrix}10 & 1 & \pi - 0.2\end{bmatrix}^T$. Such a position is a worst-case scenario for a trajectory optimizer—it is difficult for optimizers to figure out whether to go "above" or "below" the desired trajectory. To assist the optimizer in determining the trajectory, we added a small $\Delta\theta$ of 0.2 and set our reference trajectory to finish at the top of the circle. Along the way, we added relatively large disturbances. These changes drastically improved the performance of the trajectory optimization for a variety of starting positions.

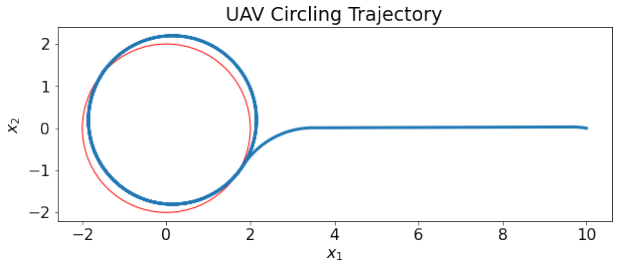

Fig. 4. Optimal trajectory for $\mathbf{x}_0 = \begin{bmatrix}10 & 1 & \pi - 0.2\end{bmatrix}^T$.

As expected, our optimal controller is the "bang-off-bang" controller shown in Figure 5, in which the input u_1 is either at $\pm u_{\max}$ or 0.

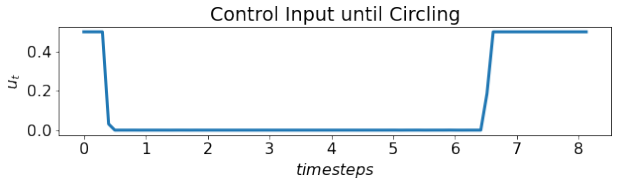

Fig. 5. Optimal control input for $\mathbf{x}_0 = \begin{bmatrix}10 & 1 & \pi - 0.2\end{bmatrix}^T$

.

V. Trajectory Tracking with Control Lyapunov Functions

Small disturbances of the initial state, trajectory, or control input are present, the optimal controller fails to reach the final desired state, as in Figure 8. Therefore, it is desirable to perform some method of trajectory "tracking", which uses online control functions to track some reference optimal trajectory.

Here, we combined a modified implementation of our Control Lyapunov Function (mCLF) that is capable of stabilizing

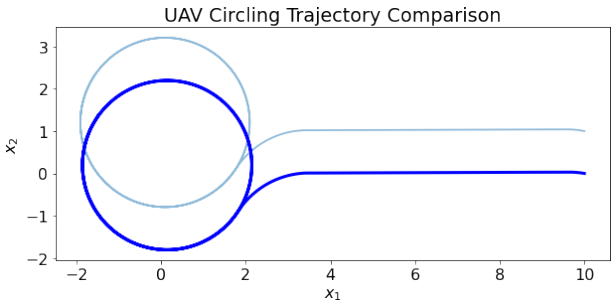

Fig. 6. Disturbed suboptimal trajectory for $\mathbf{x}_0 = \begin{bmatrix}10 & 1 & \pi - 0.2\end{bmatrix}^T$.

our system to an arbitrary state. An optimal trajectory and control inputs are generated offline. At every timestep, the current state of the system is compared to the optimal state, and a control input is generated based on the error between the optimal and current states. The activities of the algorithm depend on whether or not the actual trajectory, $\mathbf{x}_{\text{act}}$, has reached an arbitrary threshold of "closeness" to the optimal trajectory. If this has not yet occurred, then the mCLF will "look ahead" to indices further in the future in order to meet the trajectory tangentially, rather than head-on. Once the closeness threshold is met, the trajectory tracking mCLF will no longer "look ahead."

This small change arrives from the realization that meeting the closeness threshold often means that the current trajectory is similar to the optimal trajectory, and the dynamics and control constraints of the system make it desirable to look "into the future" so as to avoid overshoot. This small change drastically improved the performance of the trajectory tracking.

The algorithm is summarized in Algorithm 1.

A. Modified CLF

The current CLF is capable of tracking our system to a manifold *around* an arbitrary position, but it is not capable of navigating our vehicle to a specific state. Such a CLF is necessary to track a trajectory to an arbitrary point. Here, we use the CLF proposed in [3] which verifies the Lyapunov function

$$V = \frac{1}{2}(\tilde{x}^2 + \tilde{y}^2) + \frac{1 - \cos(\theta_e)}{k_2}$$
$$\dot{V} = -k_1\tilde{x}^2 - \frac{k_3 \sin^2(\theta_e)}{k_2}$$

where $\tilde{x}, \tilde{y}$, and θ_e are the error coordinates outlined in Section III-D. Stability for this Lyapunov function was verified by the following control assignment for $k_{1,2,3} > 0$ and $\ddot{\theta}_e = 0$, providing local exponential stability for a vehicle with velocity and unlimited yaw rate. Note that this setup is slightly different than the one proposed in Section II. Without the relaxation of this constraint, it is impossible to find a trajectory tracking Dubins vehicle that controls to a desired final state. Just as an aircraft cannot hover at one position, our Dubins car will reach the final state and continue past it. The viability of the mCLF for a less-constrained Dubins car is confirmed in [3]; here, we extend the concept to our more constrained formulation.

Algorithm 1: CLF Trajectory Tracking

```
// Optimal Trajectory
x_opt[·] ← {x_0, x_i...x_N}
u_opt[·] ← {u_0, u_i...u_{N-1}}
x_0 = x_opt[0]
// Lookahead Constant
k = 5
// Actual Trajectory
x_act[·] ← {0, 0...0}
u_act[·] ← {0, 0...0}
// Closeness Variable
close = False
for i = 0; i < N; i = i + 1 do
    // Set desired destination for mCLF
    if trajectory not yet "close" to nominal then
        if trajectory is now "close" then
            close = True
            // Aim to current closest
               position
            i_close ← index of current closest position
            x_f,des[i_close] = x_opt[i_close + 1]
            u_f,des[i_close] = u_opt[i_close]
        else
            // Look ahead
            x_f,des[i] = x_opt[i + k]
            u_f,des[i] = u_opt[i + k − 1]
        end
    else
        // Trajectory has approached
           nominal
        // Do not look ahead
        x_f,des[i] = x_opt[i + 1]
        u_f,des[i] = u_opt[i]
    end
    // Calculate trajectory
    x_f,act[i] ← mCLF (x_0, x_f,des[i], u_opt[i])
    // Set new initial position
    x_0 = x_f,act[i]
end
```

For simplicity, we chose $k_{1,2,3} = 0$. It is possible that performance could be increased by modifying these constants. Our proposed controller is:

$$u = u_{\text{ref}} - 2\tilde{y} + \sin(\theta_e)$$

When $|u| > u_{\max}$, it is clipped to $\pm u_{\max}$.

VI. Results

For our analysis, we chose to use our worst-case scenario starting point of $\mathbf{x}_0 = \begin{bmatrix}10 & 1 & \pi - 0.2\end{bmatrix}^T$. We generated 10 alternate trajectories from different starting points; our ultimate goal was to circle around the position $\mathbf{x}_f = (x_f, y_f)$ as described in the original problem statement.

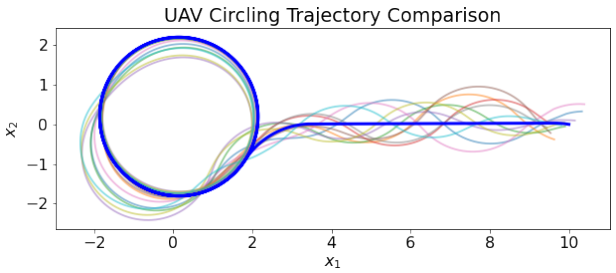

Fig. 7. Robust mCLF applied to disturbed system. Notice that the system experiences sizeable oscillations near the beginning of the trajectory, but the end result roughly approximates the desired manifold.

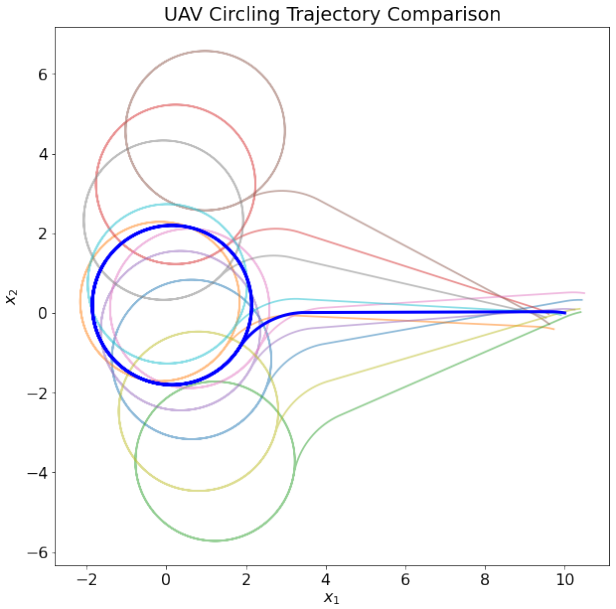

Fig. 8. Optimal controller with a disturbed $\mathbf{x}_0$. The optimal controller in a disturbed state was the worst-performing system.

A. Qualitative Analysis

While the Lyapunov controller does not generate perfect results, it closely mirrored the optimal trajectory. The "overshoot" visible is a direct result of the dynamics of our Dubins car—the constant forward velocity makes it impossible for the car to "stop" on a certain point, and the yaw rate limitations restrict the turn radius.

The more relaxed the control constraints, the better the system performance to the optimal trajectory. This suggests that removing the restrictions on our Dubins vehicle (constant forward speed, highly restricted control inputs) would cause the system to track a trajectory in a non-oscillatory way. Indeed, we would expect an asymptotically or exponentially stable Lyapunov function capable of tracking a trajectory to a point, to track the reference trajectory quite well. This hypothesis is confirmed in [3], which demonstrates trajectory tracking via Lyapunov functions for a less-constrained instantiation of the Dubins vehicle.

B. Costs

Costs were calculated until the vehicle approached some δ outside the desired final manifold; in this case, our δ was 0.5. This cost function does not capture all of the intricacies of the

We tracked control cost and time cost of all methods of control. The optimal trajectory typically had the smallest costs, but was most sensitive to disturbances. The CLF implementation had high cost, but was tolerant of disturbances. The robust trajectory tracker with the modified CLF occupied the space between the optimal trajectory and the CLF implementations. A more detailed breakdown of the cost is in Appendix A.

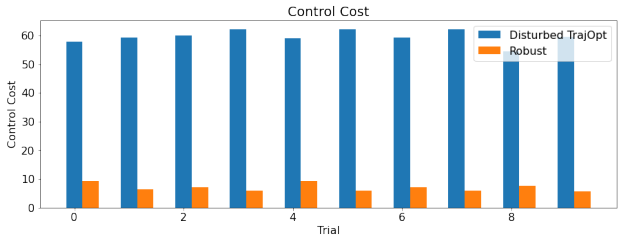

Fig. 9. Control cost comparison for disturbed trajectories.

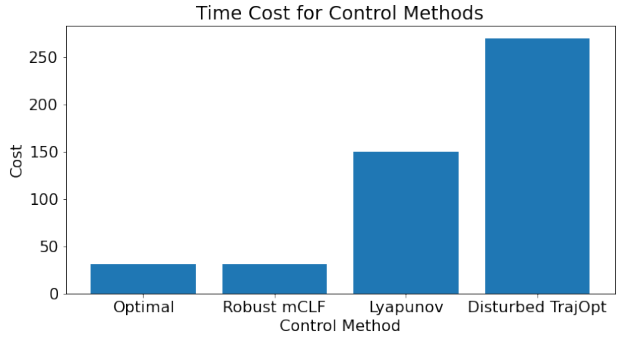

Fig. 10. Time cost comparison across control methods.

VII. Conclusion

The mCLF proposed her is a promising method for controlling the long-term behavior of a highly constrained Dubins car to a final manifold, $\mathcal{C}$. While the short-term behavior is concerning, it is possible that, the method proposed could become viable in environments where some wrong-way behavior is not highly undesirable.

By blending the robustness of the CLF with the optimality of trajectory optimization, the mCLF is a compelling alternative to other trajectory tracking methods. In particular, its capability of dealing with nonlinear, constrained systems makes it viable over a technique like Linear MPC. When the control and dynamics constraints are relaxed (e.g., decreasing the minimum radius and adding in velocity control), the mCLF as propsed in [3] is capable of tracking optimal trajectories with ease. The CLF proposed in Section II can circle a slower Dubins airplane operating with more restrictive limitations (larger maximum turn radius, slower velocity).

APPENDIX A
COST ANALYSIS

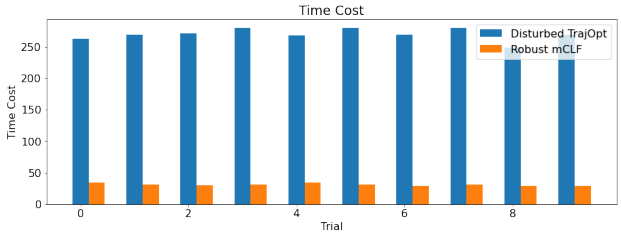

Fig. 11. Time cost comparison for disturbed trajectories.

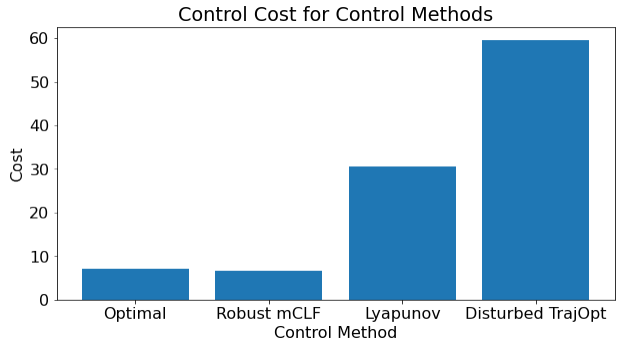

Fig. 12. Control cost comparison across control methods.

REFERENCES

[1] T. Maillot, U. Boscain, J.-P. Gauthier, and U. Serres, "Lyapunov and minimum-time path planning for drones," Jul. 2013. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00847812

[2] R. Tedrake, "Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation (Course Notes for MIT 6.832)," May 2021. [Online]. Available: http://underactuated.mit.edu/

[3] B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli, "A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles," *arXiv:1604.07446 [cs]*, Apr. 2016, arXiv: 1604.07446. [Online]. Available: http://arxiv.org/abs/1604.07446