Obstacle Avoidance for 2D Quadrotor with Hanging Load

Ayush Agrawal, Nathan Bucki, Prasanth Kotaru, David Meister, and Martin Xu

Abstract— **In this project, we use tools from optimal control to generate and stabilize feasible and safe trajectories for a two-dimensional model of a quadrotor with a cable suspended payload in cluttered environments. Due to the high degree of underactuation and nonlinear dynamics of the system, optimization-based techniques, for specifying control objectives and complex desired behaviors, are better suited over traditional control design methods for these systems. In particular, we develop our trajectory generation scheme using recent methods developed within the optimization-based collision avoidance framework, where obstacles are modeled as polyhedra and non-differentiable collision avoidance constraints are reformulated into smooth nonlinear constraints. We then use a controller in the Model Predictive Control (MPC) framework to track that trajectory and compare it with an infinite-time horizon LQR controller.**

I. INTRODUCTION

In recent years, small-scale unmanned aerial vehicles such as quadrotors have shown great potential in applications including load transportation in urban environment as well as remote areas. The problem of using a suspended payload is particularly interesting since this allows for more dynamic maneuvers as opposed to the problem where the load is rigidly attached to the quadrotor frame (the latter, in fact, leads to a higher moment of inertia leading to a more sluggish response). Adding a cable suspended payload, however, leads to additional control challenges due to (a) high degree of underactuation, (b) nonlinear dynamics of the system and (c) input and state constraints.

In this project, we approached the problem by first testing our trajectory planner and controllers on a 2D quadrotor without a hanging load. Our initial tests used only simple obstacles and defined the quadrotor as a point mass for collision detection purposes. Next, we considered the quadrotor as a full-dimensional object by defining a bounding box around the vehicle that is constrained to never intersect the obstacles. Finally, we added the hanging load with a similar bounding box and associated collision avoidance constraints.

Because the collision avoidance constraints are initially non-differentiable, we reformulate the constraints into smooth nonlinear constraints using strong duality as presented in [2]. This allows us to perform non-convex optimization using IPOPT in order to generate an optimal trajectory for the system.

After verifying our model, planner, and controllers, we tested our system in various scenarios as shown in Section V.

II. DYNAMICS DERIVATION

The dynamics for the quadrotor with a suspended payload (Figure 1) are defined based on the model presented in [1].

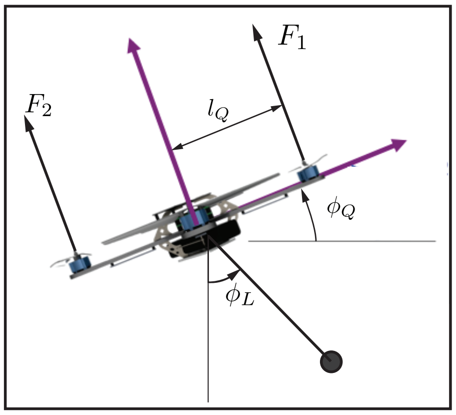

Fig. 1. 2D Model of Quadrotor with Load

TABLE I
VARIABLE DEFINITIONS FOR QUADROTOR WITH LOAD

Variable	Description
y_L, z_L	Load position
ϕ_L	Load angle w.r.t. z
l	Cable length
m_L	Load mass
$y_Q = y_L - l\sin\phi_L$ $z_Q = z_L + l\cos\phi_L$	Quadrotor position
ϕ_Q	Quadrotor angle w.r.t. y
m_Q	Quadrotor mass
J_Q	Quadrotor moment of inertia
l_Q	Quadrotor arm length
F	Total Thrust from propellers
M	Total Moment from propellers
F_1, F_2	Thrust from individual propellers

However, after deriving the equations of motion by using both the Newton and Lagrange methods, we discovered and corrected several errors in the equations of motion presented in [1]. Table I shows how the variables are defined, and the corrected equations of motion are given in (1).

$$\dot{\mathbf{X}} = \underbrace{\begin{bmatrix} \dot{y}_L \\ \dot{z}_L \\ \dot{\phi}_L \\ \dot{\phi}_Q \\ -\frac{m_Q}{m_Q+m_L} l\dot{\phi}_L^2 \sin\phi_L \\ \frac{m_Q}{m_Q+m_L} l\dot{\phi}_L^2 \cos\phi_L - g \\ 0 \\ 0 \end{bmatrix}}_{f_{vec}} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{-\sin\phi_L}{m_Q+m_L}\cos(\phi_Q-\phi_L) & 0 \\ \frac{\cos\phi_L}{m_Q+m_L}\cos(\phi_Q-\phi_L) & 0 \\ \frac{1}{m_Q l}\sin(\phi_Q-\phi_L) & 0 \\ 0 & \frac{1}{J_Q} \end{bmatrix}}_{g_{vec}} \begin{bmatrix} F \\ M \end{bmatrix}, \tag{1}$$

$$\begin{bmatrix} F \\ M \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -l_Q & l_Q \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} \tag{2}$$

$$\mathbf{X} = \begin{bmatrix} y_L & z_L & \phi_L & \phi_Q & \dot{y}_L & \dot{z}_L & \dot{\phi}_L & \dot{\phi}_Q \end{bmatrix} \tag{3}$$

Project Video can be found here: https://youtu.be/BfGQMpfwSIc

III. OPTIMAL TRAJECTORY GENERATION

We define the control problem in two parts: first we generate an optimal trajectory that includes obstacle avoidance constraints, and then we track the generated trajectory using different controllers. The following details our formulation of the optimal trajectory generation problem.

The obstacles are defined by (4), and the polyhedra representing the quadrotor are defined by (5), where $R(x_k)$ and $t(x_k)$ are the respective rotation matrices and translation vectors representing the rotation and translation of the polyhedra at time k. We use one bounding box to define the size of the quadrotor body and a second bounding box to define the size of the load.

$$\mathbb{O}^{(m)} = \{y \in \mathbb{R}^n \colon A^{(m)} y \leq b^{(m)}\}, \tag{4}$$

$$\mathbb{E}(x_k) = R(x_k)\mathbb{B} + t(x_k), \quad \mathbb{B} := \{y \colon Gy \leq g\}. \tag{5}$$

We reformulate the obstacle avoidance constraints using strong duality as shown in [2]. This gives the optimization problem in (6), where the variables are defined as in table II, $k = 0, \ldots, N$ and $m = 0, \ldots, M$.

TABLE II

VARIABLE DEFINITIONS FOR TRAJECORY GENERATION

Variable	Description
T_{opt}	Timestep length
x_k	State vector at time k
u_k	Control input vector at time k
$\lambda_k^{(m)}$, $\mu_k^{(m)}$	Dual variables for obstacle m at time k
q_1, q_2	Timestep cost tuning parameters
R	Control input cost tuning parameter
N	Total number of timesteps
M	Total number of obstacles
$f(x_k, u_k)$	System dynamics
$h(x_k, u_k)$	Input and state constraints
τ_F	Final time
x_F	Goal state

$$\begin{array}{ll}
\min\limits_{T_{opt}, \mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}} & q_1 T_{opt} + q_2 T_{opt}^2 + \sum\limits_{k=0}^{N} u_k^T R u_k \\
\text{s.t.} & x_0 = x(0), \; x_{N+1} = x_F, \tau_F = N T_{opt} \\
& x_{k+1} = f(x_k, u_k), \; h(x_k, u_k) \leq 0, \\
& -g^\top \mu_k^{(m)} + (A^{(m)} t(x_k) - b^{(m)})^\top \lambda_k^{(m)} > 0, \\
& G^\top \mu_k^{(m)} + R(x_k)^\top {A^{(m)}}^\top \lambda_k^{(m)} = 0, \\
& \|{A^{(m)}}^\top \lambda_k^{(m)}\|_* \leq 1, \; \lambda_k^{(m)} \leq 0, \; \mu_k^{(m)} \leq 0.
\end{array} \tag{6}$$

We chose our cost function such that the time it takes to reach the final state is minimized. Because a constraint on the final state of the system is included, we did not include any stage cost based on the state vector at each timestep. Thus, we seek to minimize some weighted sum of the time it takes to reach the goal position and the magnitude of the control inputs used at each timestep.

IV. TRACKING CONTROL

We tested the ability of both an MPC and infinite horizon discrete time LQR controller to track the trajectories generated by our optimization problem for several different scenarios.

A. DLQR Controller

The LQR controller was intended to serve as a baseline to which we could compare our MPC controller, and was defined as a full state feedback controller that minimizes the following cost function. Q_d and R_d are chosen to be diagonal matrices that result in reasonable tracking responses. $\hat{x}$ is the generated optimal trajectory to be tracked.

$$J = \frac{1}{2} \sum_{k=0}^{\infty} \left\{ e(k)^T Q_d e(k) + u^T(k) R_d u(k) \right\} \tag{7}$$

$$e(k) = x(k) - \hat{x}(k). \tag{8}$$

The system is linearized and discretized about the trajectory at each timestep, giving an A and B matrix with which we calculate the optimal feedback gain K. Finally, the optimal control input obtained from trajectory generation is added in as a feedforward term,

$$u(k) = -K e(k) + \hat{u}(k). \tag{9}$$

B. MPC tracking

Because the MPC controller is intended to run in real time on the system, we make some relaxations to the optimization problem given in (6). Namely, the MPC program does not include collision constraints and has a cost function based on the control effort and state deviation from the optimal trajectory. Equation (10) gives the optimization problem solved by the MPC tracking controller at time-step k, with horizon N, for tracking the trajectory $\hat{x}$.

$$\begin{array}{ll}
u_k^* = \arg\min\limits_{\mathbf{x}, \mathbf{u}} & \sum\limits_{i=0}^{N} (x_i - \hat{x}_{k+i})^T Q (x_i - \hat{x}_{k+i}) + (u_i^T - \hat{u}_{k+i}) R (u_i - \hat{u}_{k+i}) \\
\text{s.t.} & x_0 = \hat{x}(k), \\
& x_{i+1} = f(\hat{x}_i, \hat{u}_i) + A(x_i - \hat{x}_i) + B(u_i - \hat{u}_i), \\
& x_{iL} \leq x_i \leq x_{iU} \\
& u_{iL} \leq u_i \leq u_{iU}.
\end{array} \tag{10}$$

`MATLAB ode45` solver is used to simulate the forward dynamics. The dynamics $\dot{x} = f(x, u)$ are simulated between the time-span $[t_k, t_{k+1}]$ using the input u_k^* calculated in (10). Note that linearized dynamics (about the reference trajectory generated in section III) are used in the MPC control, whereas the complete non-linear dynamics are used in the simulations.

V. PERFORMANCE

We tested our trajectory generator and tracking controllers on three difference scenarios: flying through a window, swinging the mass into an inverted pendulum configuration, and flying between two offset triangles. Figure 2 shows the optimal trajectory generated by our trajectory generation program for the window obstacle scenario.

We tested our tracking controllers by changing either the control input limits or the dynamics of the system. Figure 3 shows the performance of the two tracking controllers when tighter control input constraints are imposed than during trajectory generation. Because the MPC controller is able to take the new control input constraints into account, it outperforms the LQR controller. Figure 4 shows how the control inputs saturate during the simulation for each tracking controller.

Figure 5 shows how the tracking controllers respond to a 10% increase in weight of the hanging load during simulation. Although both controllers perform roughly the same, the MPC controller has slightly less tracking error than the LQR controller.

Animations of the trajectory and plots for the other two test cases are shown in our presentation.

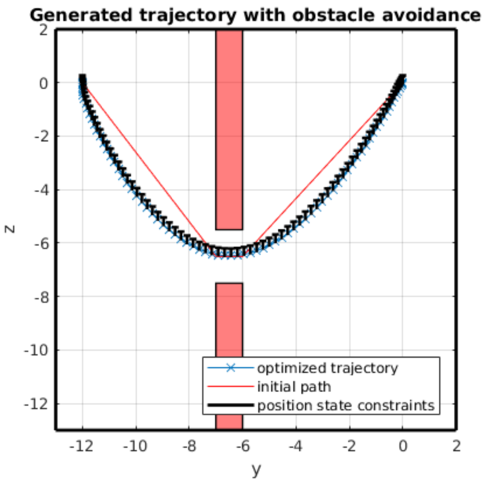

Fig. 2. Generated trajectory with window obstacle. The solver is given an initial trajectory that is kinematically but not dynamically feasible.

VI. CONCLUSIONS

Reformulating the obstacle constraints using strong duality allowed us to generate trajectories capable of avoiding obstacles while minimizing a weighted sum of control inputs and trajectory execution time. We used both an MPC and infinite horizon discrete time LQR controller to track the generated trajectory for three difference scenarios, which showed the superior tracking ability of of the MPC controller compared to the LQR controller in cases with tighter input constraints and model errors.

REFERENCES

[1] Koushil Sreenath, Nathan Michael, and Vijay Kumar. Trajectory generation and control of a quadrotor with a cable-suspended load – a differentially-flat hybrid system. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[2] Xiaojing Zhang, Alexander Liniger, and Francesco Borrelli. Optimization-based collision avoidance. *arXiv preprint arXiv:1711.03449*, 2017.

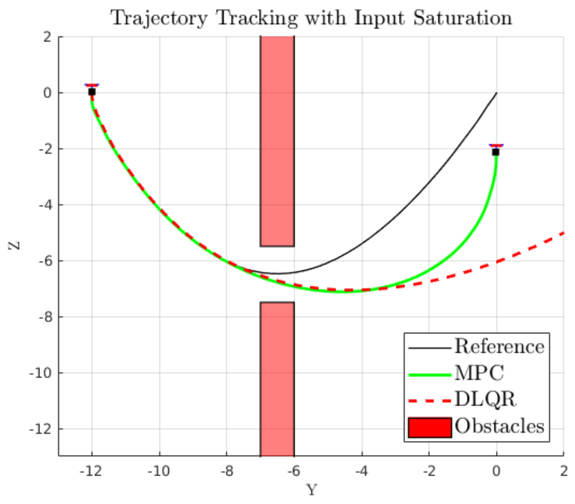

Fig. 3. Y-Z Trajectory of load when using either LQR or MPC tracking controller. Deviations from reference trajectory are due to tighter constraints on the control input of the tracking controllers.

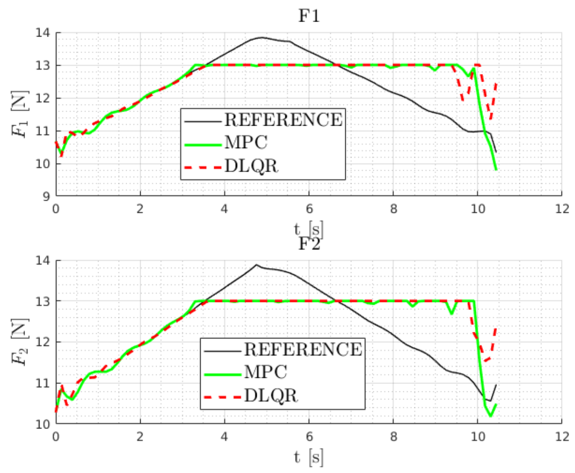

Fig. 4. Control inputs over time for both tracking controllers. The inputs saturate around t = 3, but the MPC controller performs actions to bring it closer to the desired trajectory than the LQR controller as shown by the small dips from the control input limit.

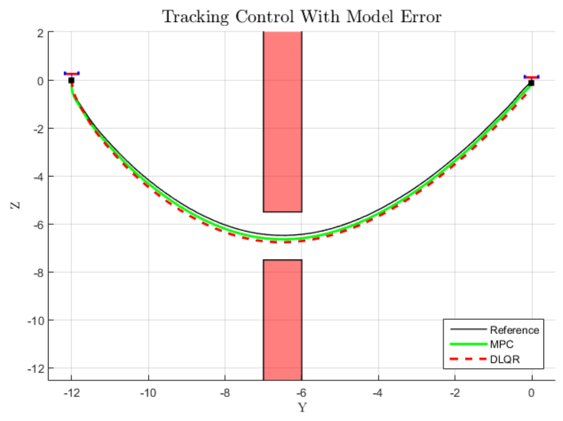

Fig. 5. Comparison of trajectory tracking error between LQR and MPC controllers with model uncertainty. During simulation the mass of the load was increased by 10%