MPC Control of Multiple Quadcopters Cooperatively Lifting an Object

Raghav Anand, Jason Anderson, Rachel Lim, Rohan Sinha

***Abstract*— In this project, we use Model Predictive Control ("MPC") methods to direct and control a team of three Unmanned Aerial Vehicles (each, a "drone") in Robot Operating System ("ROS") simulation to lift a box using tensile ropes, modeled as a spring-damper connection, extended from each drone center of mass. The drones follow planned trajectories and use on-board state estimation to reject the disturbance force from the box. Drone delivery systems, typically for the last-mile or short warehouse trips, are becoming an increasingly popular research topic in advanced control. They can provide superior capability over traditional transportation methods such as trucks and reduce human operator requirements, among other transportation costs associated with ground-based travel. Performing the complex maneuvering in the transportation method requires solving computationally-heavy problems in perception, planning and control, only recently achievable on small embedded systems in real-time. In this paper, we provide a simulation-based modelling approach to the problem. A video of our project can be found here: https://tinyurl.com/ybkfdj76. All code is publicly available at [1].**

I. Introduction

Due to quadcopters' agile and mechanically simple nature, they are often the subject of recent research regarding cooperative tasks. In this paper, we discuss using quadcopters to cooperatively achieve a lifting task one quadcopter could not complete. We present a cooperative Model Predictive Control ("MPC") method to control multiple drones to move an object suspended between them with spring-damper ropes. The lifted object state and the presence of other drones is not known to each drone MPC controller. The affect of the lifted object is accounted in the MPC through an MPC disturbance rejection augmentation and state estimation. Therefore, the drone is not able to make predictions of how other drone's will affect the hanging mass. The drones, the lifted object, and the controllers were implemented and tested using a non-linear multi-body simulator built with Robotic Operating System ("ROS"). Applications of our method include aerial last-mile delivery systems of massive objects and craning or lifting heavy objects in airspace-only accessible places.

II. Model

For notation, see the Appendix.

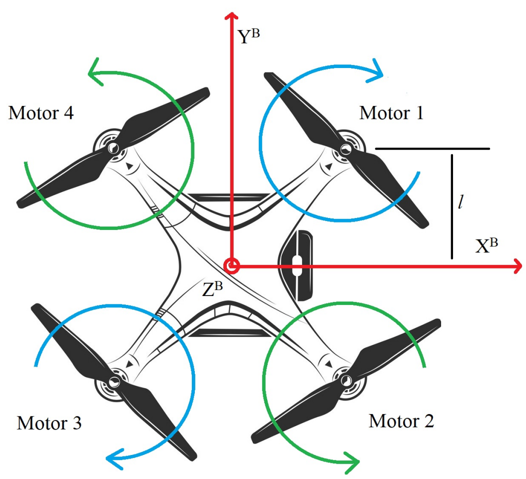

Fig. 1. Drone Coordinate Frame

A. Non-linear Drone Model

The control system is tested using a non-linear dynamics simulator, where orientations are transformed from Euler angles to quaternions to avoid gimbal locking. Forces and moments from the rotors are transformed from the body coordinate frame to the global one, after which the dynamics are advanced using a forward Euler discretization using a fixed timestep. The nonlinear dynamics equations in the global coordinate frame are given by:

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}, \quad \frac{d}{dt}\begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \end{bmatrix} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \omega$$

$$\frac{d}{dt}\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \mathbf{F_{ext}} + \mathbf{R}(\theta_x, \theta_y, \theta_z)\begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^{4} F_i \end{bmatrix} \tag{1}$$

$$\frac{d}{dt}\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \mathbf{R}(\theta_x, \theta_y, \theta_z)(\mathbf{J}^{-1}(\mathbf{M} - \omega \times (J\omega))) \tag{2}$$

Here $\mathbf{R}(\theta_x, \theta_y, \theta_z)$ is the 3D rotation matrix parametrized by the orientation of the drone. $\mathbf{F_{ext}}$ denotes a general external force, in this case as a result of tension forces acting on the drone from the box. $\mathbf{M}$ is the resulting moment from the rotor forces and reaction torques, given by:

$$\mathbf{M} = \begin{bmatrix} 0 \\ 0 \\ k(\sum_{i=1}^{4}(-1)^{i-1}F_i) \end{bmatrix} + \left(\sum_{i=1}^{4} \mathbf{S_i} \times \begin{bmatrix} 0 \\ 0 \\ F_i \end{bmatrix}\right) \tag{3}$$

$$\mathbf{S_1} = \begin{bmatrix} l \\ l \\ 0 \end{bmatrix}, \mathbf{S_2} = \begin{bmatrix} l \\ -l \\ 0 \end{bmatrix}, \mathbf{S_3} = \begin{bmatrix} -l \\ -l \\ 0 \end{bmatrix}, \mathbf{S_4} = \begin{bmatrix} -l \\ l \\ 0 \end{bmatrix} \tag{4}$$

The simulation assumes a ground plane exists at z = 0, and as such the drone dynamics are floored whenever the drone is stationary on the ground.

B. Linearized Discrete Drone Model

We derive a linear model suitable for MPC control according to the method provided by Mueller [5], which linearizes the system assuming small θ_x and θ_y. Equation (5) provides the continuous time form of the dynamics.

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \quad \frac{d}{dt}\begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \end{bmatrix} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{5}$$

$$\frac{d}{dt}\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} \theta_y g \\ -\theta_x g \\ -g \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{m}\sum F_i \end{bmatrix}$$

$$\frac{d}{dt}\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = J^{-1}\begin{bmatrix} l & -l & -l & l \\ -l & -l & l & l \\ \kappa & -\kappa & \kappa & -\kappa \end{bmatrix}\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix}$$

This linear model matches the non-linear formulations best when the drone is in a steady-state level hover. To create the full continuous dynamic system, we create the appropriate block matrices A_c and B_c that correspond to the standard control law with the state vectors of Equation (6). The subscript c denotes the continuous form.

$$\frac{d\mathbf{x}}{dt} = A_c\mathbf{x} + B_c\mathbf{u} + b_{g-c}$$

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \\ \theta_x \\ \theta_y \\ \theta_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \quad b_{g-c} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -g \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{6}$$

The discrete forms of these equations are calculated using a forward Euler difference equation, given below.

$$A = I + A_c\Delta t \tag{7}$$

$$B = B_c\Delta t \tag{8}$$

$$b_g = b_{g-c}\Delta t$$

Equation (9) provides the final linearized difference dynamics for the drone, where t denotes the time index.

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B_t\mathbf{u}_t + b_g \tag{9}$$

C. Linearized Model Mismatch

To benchmark the accuracy of the linearized model, a test control sequence (obtained by solving a CFTOC problem) is simulated using both the linearized an non-linear model from the same initial conditions with a timestep of .1 seconds over a 3 second period. Figure 2 shows that as long as the yaw and pitch of the drone are constrained to be small, the deviation between the linear and non-linear model is small, so the approximations are valid.

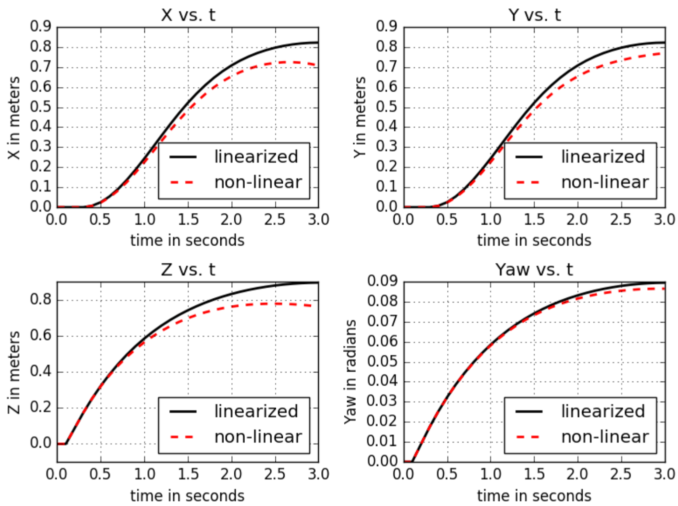

Fig. 2. Trajectory Comparisons of the nonlinear and linearized models.

D. Box Model

The box model treats the ropes as springs which exert an external force on the drones as they move away from the box. For each drone, the spring force acting on the box is found by Equation (10), where (x_B, y_B, z_B) is the box position, l_0 the unstretched spring length, k is the spring constant, and (x,y,z) is the position of the connected drone center of mass. The external forces with respect to the drone are then given by Equation (11), and the total forces on the box can be found by Equation (12) where $F_g = -mg$ is the force of gravity and c is the damping constant. k and c are chosen such that the ropes are critically damped, mimicking the elasticity of a real rope.

$$l_i = \begin{bmatrix} x_B \\ y_B \\ x_B \end{bmatrix} - \begin{bmatrix} x^i_{drone} \\ y^i_{drone} \\ z^i_{drone} \end{bmatrix}$$

$$F^i_s = -k * (||l_i|| - l_0) * \frac{l_i}{||l_i||} - c(v_{box} - v^i_{drone}) \tag{10}$$

$$F^i_{ext} = -F^i_s \tag{11}$$

$$F = F_g + \sum_{i=1}^{4} F^i_s \tag{12}$$

F^i_s is set to zero whenever $l_i < l_0$ as a rope can only exert forces when in tension.

III. MPC Control

We incorporate the linearized discrete dynamic model of the drone of Section II-B into the standard MPC formulation provided in Equation (13) [2].

$$\begin{aligned} \min_{\forall \mathbf{x}_t\ \forall \mathbf{u}_t} \quad & \mathbf{x}_n P \mathbf{x}_n^\top + \sum_{t=0}^{n-1} \mathbf{x}_t Q \mathbf{x}_t^\top + \mathbf{u}_t R \mathbf{u}_t^\top \\ s.t.\ & \mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t + b_g \ \forall t \in [0 \dots n-1] \\ & \mathbf{x}_0 = \mathbf{x}_0 \\ & \mathbf{x}_t \in X \ \forall t \in [1 \dots n] \\ & \mathbf{u}_t \in U \ \forall t \in [0 \dots n-1] \end{aligned} \tag{13}$$

A. State and Input Constraints

From Equation (13), X is set to restrict each of the lateral angular positions to within 15° of zero and each of the lateral angular velocities to within 10° per second of zero. This ensured that the drone never ventured too far away from the small-angle assumptions of the linearized model of Section II-B, and that the drone wouldn't spin out of control. No other constraints were made to ensure efficient MPC solving time.

B. Disturbance Rejection

To account for disturbances from the drone environment, we assume that all disturbances are described completely by a 3-state disturbance force vector. This vector is appended to Equation (9) into Equation (14) and implemented into the MPC formulation of Equation (13).

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t + b_g + B_d F_d \tag{14}$$

The matrix B_d maps F_d to the appropriate velocity states in $\mathbf{x}$ while incorporating the affect of m and Δt.

C. Disturbance State Observation

We incorporate a disturbance force observer into the MPC code. This observer works according to Equation (15), derived from Borrelli[3].

$$\begin{bmatrix} \hat{\mathbf{x}}_{t+1} \\ \hat{\mathbf{d}}_{t+1} \end{bmatrix} = \begin{bmatrix} A & B_d \\ 0 & I_3 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}_t \\ \hat{\mathbf{d}}_t \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} \mathbf{u} + \begin{bmatrix} b_g \\ 0 \end{bmatrix} - L(\hat{\mathbf{x}}_t - \mathbf{x}_t) \tag{15}$$

From Equation (15), $\hat{\mathbf{d}}$ is the 3-vector concatenation of the estimated disturbance forces in the x, y, and z directions, and $\hat{\mathbf{x}}$ is an estimated state. We calculate $\hat{\mathbf{x}}$ separately from the simulated state so that the state estimator is ready for hardware implementation. The matrix L provides $\begin{bmatrix} A & B_d \\ 0 & I_3 \end{bmatrix} - LC$ eigenvalues with satisfactory disturbance state convergence, where $C = [I_{12}\ 0]$. Through a small amount of trial and error, we found these eigenvalues to be $\frac{[93,94,\dots,107]}{130}$.

D. Tracking with Disturbance

To achieve disturbance rejection, we apply the augmentation procedure provided by Borrelli[3] on the dynamics to solve the appropriate set of equilibrium states that track a reference with the disturbance forces. The tracking position coordinates are denoted x_∞, y_∞, z_∞, and the corresponding equilibrium states are denoted $\mathbf{x}_\infty$ and $\mathbf{u}_\infty$. Here, the tracking is restricted to position only, as the drone may need to change its orientation to compensate for disturbance. We solve for the appropriate equilibrium states by the optimization problem of Equation (16).

$$\min_{\mathbf{x}_\infty \ \mathbf{u}_\infty} \ \mathbf{u}_\infty R \mathbf{u}_\infty^\top$$
$$\begin{bmatrix} A-I & B \\ I_3 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_\infty \\ \mathbf{u}_\infty \end{bmatrix} = \begin{bmatrix} -B_d F_d - b_g \\ x_\infty \\ y_\infty \\ z_\infty \end{bmatrix} \tag{16}$$

Because the constraints of Equation (16) form an under determined linear system, it must be formulated as an optimization problem with the equations as constraints. Here, the cost is a function of the forces, in order to minimize the drone power required. From the calculated $\mathbf{x}_\infty$ and $\mathbf{u}_\infty$, we modify the MPC formulation to attribute costs to deviations from $\mathbf{x}_\infty$ and $\mathbf{u}_\infty$.

E. Final MPC

In our simulation, the state of the drone is perfectly known at each iteration time step and the simulator, controller, and disturbance estimator run at the same frequency. At each simulation iteration, we solve the final MPC problem of Equation (17) using the previous iteration's state and estimated disturbance, $\mathbf{x}_{-1}$ and $\hat{\mathbf{d}}_{-1}$. In the MPC problem, the disturbance $\hat{\mathbf{d}}_{-1}$ is assumed constant, but it is still updated with each MPC iteration. After calculating the final MPC problem solution, $\mathbf{u}_0$ is actuated by the drone through the simulator.

$$\begin{aligned} \min_{\forall \mathbf{x}_t \ \forall \mathbf{u}_t} \ & (\mathbf{x}_n - \mathbf{x}_\infty) P (\mathbf{x}_n - \mathbf{x}_\infty)_n^\top \\ & + \sum_{t=1}^{n-1} (\mathbf{x}_t - \mathbf{x}_\infty) Q (\mathbf{x}_t - \mathbf{x}_\infty)_t^\top \\ & + \sum_{t=0}^{n-1} (\mathbf{u}_t - \mathbf{u}_\infty) R (\mathbf{u} - \mathbf{u}_\infty)_t^\top \\ s.t. \ & \mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t + b_g + B_d \hat{\mathbf{d}}_{-1} \ \forall t \in [0 \dots n-1] \\ & \mathbf{x}_0 = \mathbf{x}_{-1} \\ & \mathbf{x}_t \in X \ \forall t \in [1 \dots n] \\ & \mathbf{u}_t \in U \ \forall t \in [0 \dots n-1] \end{aligned} \tag{17}$$

IV. Motion Planning

The MPC strategy from the previous section can accurately track a reference position subject to external forces from the box. However, the algorithm assumes no knowledge of the other drones, first, because the drones might not be able to communicate with each other in an application, and second, so that new drones can attach themselves during the control task without changing the problem formulation. Instead, a separate motion planner takes in high level destination commands from a user and generates reference targets for each drone in the configuration ensuring the drones stay at a fixed distance from each other. The motion planner dynamically regulates the reference state for each drone based on its current position, providing the next waypoint when each drone has reached its reference. The simpler the trajectories, the sparser the references can be, simulating a cooperative decentralized strategy for the drones.

V. Simulator

A. ROS and Python

We implemented our formulations using ROS, a middleware system widely used to facilitate modular simulation and development of complex robotics projects. Our simulator is able to simulate any number of drones in closed loop, with or without a box and can be used to develop other control architectures in the future.

Simulations of reference tracking, disturbance rejection and box were implemented and tested in the ROS framework. All of our code is written in Python. We used CVXPY [4] for all the required optimization.

We used ROS's RViz, a built-in ROS package to create a vizualizer for the simulations. The visualization script first finds all the drones and the box in the simulation. It then plots their current position, future waypoints and previous trajectories in a 3D space.

VI. Results

Figure 3 shows the visualization of 3 drones the box to different waypoints. In simulation, the drones and the box started of on the ground, lifted up, and then followed a path set out by the motion planner. As the figure shows, the drones are able to effectively adapt to the disturbance force from the box and maintain their reference trajectory. In the simulation, the initial disturbance estimate was set to zero. As is visible from the figure, the drones are pulled towards each other when the ropes to the boxes reach tension, but the estimator measures the disturbance well enough for the drones to compensate well before a collision occurs. With or without the box, the drones are able to achieve accurate reference tracking. With the box, it is notable that the pulling of one drone on the box does not seem to be amplified by the control actions from any of the other drones. Since the lifting configurations are highly symmetrical, the drones achieve a stable equilibrium coupled to each other, each carrying a third of the load. Benchmarking the lifting of the box to the trajectory tracking without the box does not show significant impact on performance. We recorded videos of the simulations with and without the box, and they are available at link in the abstract.

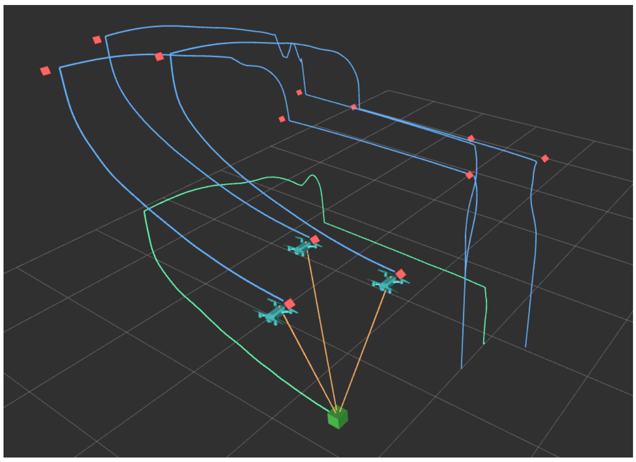

Fig. 3. Trajectories of 3 drones carrying a box. Notice that the drones are pulled into each other when the connections reach full extension.

VII. Conclusion and Future Work

We have presented a control framework for the cooperative lifting of objects with multiple quadcopters. The most significant result of this work is the accurate real-time estimate of the disturbance force induced by the hanging object. A disturbance rejecting MPC design will be able to lift an object in the presence of multiple other agents in a stable configuration.

The entire control problem does not need to be solved in a centralized fashion, optimizing over the control inputs of all drones simultaneously, nor will a decentralized strategy need to predict future actions of the other agents. The motion planner does not need to distinguish between a task where the drones are lifting the box, or performing simple point to point navigation. Future work could include allowing the drones to know about the agents and dynamically plan their paths to optimize the trajectory of the lifted object. All authors contributed equally to this work.

REFERENCES

[1] https://www.github.com/GoldeneyeRohan/Multi_Drone_Control.

[2] Francesco Borrelli. Model predictive control algorithm, feasibility and stability. *University of California, Berkeley, Department of Mechanical Engineering, ME 231A: Experimental Control Design I.*

[3] Francesco Borrelli. Mpc: Tracking, soft constraints, move-blocking. *University of California, Berkeley, Department of Mechanical Engineering, ME 231A: Experimental Control Design I.*

[4] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[5] Mark Mueller. Quadcopter dynamics. *University of California, Berkeley, Department of Mechanical Engineering, ME 136: Introduction to Control of Unmanned Aerial Vehicles*.

APPENDIX

In this paper, x, y, z, correspond to the Cartesian position state where x and y are the lateral coordinates with x generally meaning forward and z meaning upward, and x, y, and z forming a right-hand coordinate system. The velocities in the corresponding coordinate axes are denoted v_x, v_y, v_z; the Euler angle positions, θ_x, θ_y, θ_z ; and the Euler angle velocities, ω_x, ω_y, ω_z. For any states with respect to the body frame of the drone (centered at the drone center of mass with the aforementioned directional definitions) have the super script B, whereas, with no superscript, the states are global. A concatenates state vector is in bold, for example, $\mathbf{x}$ and $\mathbf{u}$, which are defined later in Equation (6). The diagonal matrix, J, contains three elements that are the three moments of inertia of the x^B, y^B, and z^B axes of the drone. The drone mass is m; κ proportionally relates the force of a drone propeller to the moment induced on the drone. If the drone were a square aligned with the x^B and y^B axes with the four propeller centers at the square corners, the side length of the square is $2l$. The length of time in between iterations is Δt, either in the simulator or in the controller. The iteration number of a specific state is denoted with a subscript, with x_0 being the x position at present.