Task Space Control of Quadcopter using Model Predictive Control for Obstacle-Aware Operations in Construction Environment

Antreas Kourris, Jonah Eggenkemper, Manas Reddy Ramidi, Mukil Saravanan

***Abstract*— This study presents a Model Predictive Control (MPC) framework for drone navigation in complex construction environments. By modeling the drone as a point mass in 3D space, the approach addresses challenges such as obstacle avoidance and constrained spaces. Simulations using the gym-pybullet-drones framework and ACADOS validate the robustness of the method in terms of waypoint tracking, obstacle avoidance, and computational efficiency in different scenarios. The model is also shown to be resilient to noise. The results are compared with state-of-the-art metrics and demonstrate the effectiveness of the MPC in ensuring safe and accurate navigation (with a mean waypoint tracking error of 0.27 m and mean velocity of 5 m/s). The potential for further optimization to handle dynamic obstacles and reduce computational overhead is also discussed.**

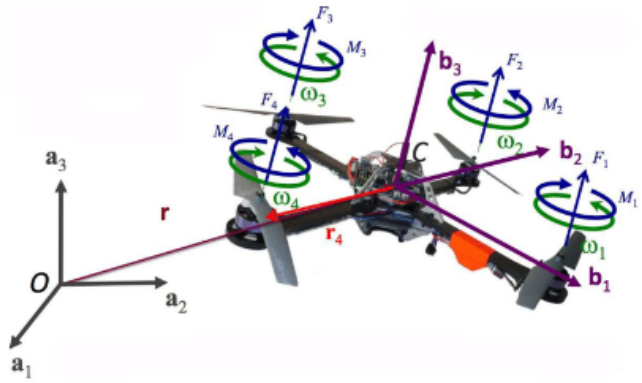

Fig. 1. Quadrotor model

I. INTRODUCTION

Drone technology has many applications in payload transport, aerial manipulation and agile flight. The functionality of drones can be enhanced with the right path planning approaches specific to the target applications. Their strengths over other mobile robotic systems lie in their ability to navigate the environment in three dimensions without the need for ramps or smooth and hard surfaces. For this project, the drone will be used to assist with tasks on an active construction site. The drone could be used in a variety of ways to assist in the construction process, particularly for high-rise buildings. Such tasks could include moving light tools and machinery from the ground to construction workers on each floor, or regularly inspecting the building to check and document the work being done by contractors. These applications were chosen because they provide different types of obstacles for the robot to navigate. for the robot to navigate through. We plan to do this using an MPC planner that is robust to different environments, and also hope to create a general purpose planning drone with payload carrying and an integrated manipulator.

We consider the quadrotor in 1 as a point mass, resulting in operations within the configuration space $\mathscr{C} = \mathbb{R}^3$. Model predictive control was chosen as the local planner because it can take into account multiple constraints. These constraints can be both physical, such as the maximum acceleration of the drone or the limits of the physical space in which it can operate, as well as safety constraints, such as obstacle avoidance. In addition, because MPC optimizes the trajectory at each step, it is to some extent robust to model mismatch. Finally, the cost function can be adapted to optimize for different values such as states, inputs or obstacle avoidance, as in the case of the potential field method.

The report is organized as follows. First, the problem to be solved is defined in Chapter II. Chapter III describes the implementation of motion planning with MPC. Finally, the results and comparisons obtained are discussed in Chapter IV. in Chapter V, including recommendations for future improvements for our implementations.

II. PROBLEM DEFINITION

Modern construction sites are inherently complex and dynamic, particularly when buildings are only partially completed. Structural elements such as exposed beams, scaffolding, and temporary supports create intricate spaces through which drones must navigate safely. Additionally, construction sites can be reconfigured daily, as new materials, machinery, or debris appear unexpectedly, introducing unforeseen obstacles. These factors pose substantial challenges for drones tasked with tasks like inspection, material delivery, or real-time progress monitoring. Consequently,there is a need for a robust, adaptable control strategy that can ensure safe and accurate flight in such challenging environments.

In such an environment, the drone must operate with acute awareness of its surroundings while maintaining a stable flight path. Collisions with structural components, equipment, or personnel are not only hazardous but can also disrupt critical construction processes. In addition, the complexity of partially constructed interiors, where lines of sight can be obstructed by walls or scaffolding, increases the risk of navigation errors.

This report presents a MPC-based approach to drone control tailored for construction sites, discussing the system architecture, dynamic modeling, and constraint handling.

In order to evaluate the robustness of the MPC three different scenarios were created. The first is an empty world,

the second has randomly placed spherical obstacles, and the third features a building with pillars. In all three scenarios, the drone must navigate to waypoints, which are either randomly generated or part of a predefined trajectory, while avoiding obstacles. Visualizations of these scenarios can be seen in figure 2

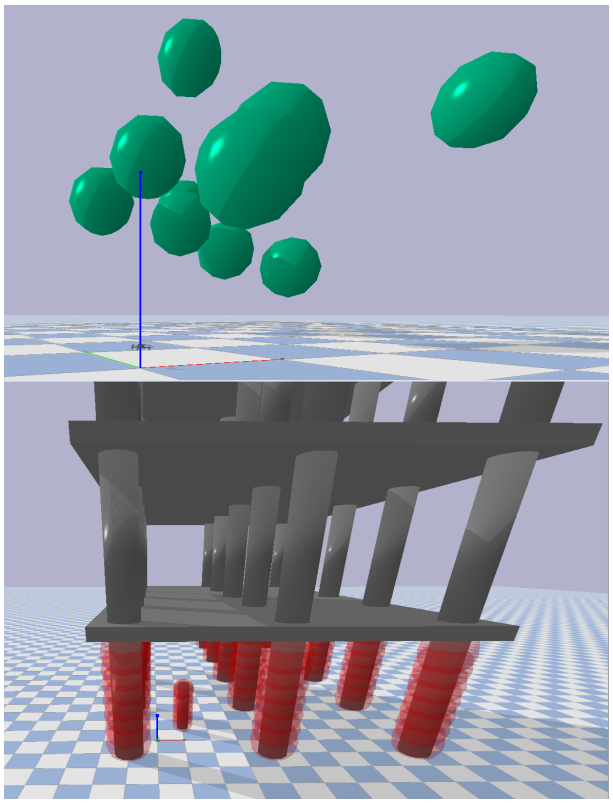

Fig. 2. Scenario with randomly generated spherical obstacles (top) and scenario with building (bottom)

The scenarios were created using the pybullet gym drones simulation environment [2].

III. MOTION PLANNING METHOD

To address the problems discussed, Model Predictive Control (MPC) is implemented, which provides an advanced control framework capable of handling state and input constraints, predicting future system states, and proactively adjusting inputs to optimize performance. Using an internal model of the drone's dynamics (including position, velocity), MPC anticipates how the drone will evolve over a short time horizon. It then calculates the optimal control actions needed to meet multiple objectives - maintaining stability, avoiding obstacles and staying within safety limits - while minimizing control effort and tracking error.

A. Point Mass Drone Model

The drone will be treated as a point mass in 3D space with linear dynamics. The work space of the drone is R^3 and the configuration space is R^3

The dynamic equations used are the following.

State Vector:

$$\vec{x} = \begin{bmatrix} x, y, z, v_x, v_y, v_z \end{bmatrix}^T$$

The state vector includes the drone position in x,y,z coordinates and the drone's velocities in those axes.

Input Vector:

$$\vec{u} = \begin{bmatrix} a_x, a_y, a_z \end{bmatrix}^T$$

The input vector is the acceleration in the x, y and z axis.

State and Input Matrices:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

From the expressions above, the system dynamics can be defined as:

$$\dot{\vec{x}} = A\vec{x} + B\vec{u}$$

where:

$$\begin{aligned} \dot{x} &= v_x \\ \dot{y} &= v_y \\ \dot{z} &= v_z \\ \dot{v_x} &= a_x \\ \dot{v_y} &= a_y \\ \dot{v_z} &= a_z - g \end{aligned}$$

B. Global Planner

As the focus of this work was the design and implementation of a model predictive controller, the global planner will be simple waypoints generated randomly or by fixed trajectories. This is motivated by the use case of construction. On a construction site, certain parts of the building need to be monitored during an inspection task, or the drone needs to constantly patrol a predefined route.

C. Local Planner

The MPC will discretize the dynamic model of the drone. At this stage, we formalize the cost function of a generic MPC with a finite time horizon N as follows

$$\min_{x,u} \frac{1}{2} \sum_{t=0}^{N} J_t(x_t, u_t, \text{ref})$$

such that states are constrained by the system dynamics and C-space of the robot while inputs are constrained by system limitations on the feasible velocities, accelerations, etc.,

$$x_{t+1} = f(x_t, u_t) \qquad \forall t \in [0, N) \tag{1a}$$

$$g(x_t, u_t) \leq d_t \qquad \forall t \in [0, N] \tag{1b}$$

$$x_0 = x_{init} \tag{1c}$$

The cost function used for this MPC uses takes into account the quadratic error between the current drone state and the target drone state, as well as a repulsive potential field force from the obstacle and an attractive one from the goal position. The constraints used are lower and upper bounds for the states along with the inputs. There are also spherical constraints for the obstacles. The optimization problem can be seen below.

$$\begin{aligned}\min_{\vec{x},\vec{u}} \sum_{t=0}^{N} \Big[(\vec{x}_t - \vec{x}_{\text{ref}})^T Q (\vec{x}_t - \vec{x}_{\text{ref}}) + \vec{u}_t^T R \vec{u}_t \Big] \\ + \vec{F}_{\text{atr}}^T W_{\text{atr}} \vec{F}_{\text{atr}} - \vec{F}_{\text{rep}}^T W_{\text{rep}} \vec{F}_{\text{rep}}\end{aligned}$$

$$\text{subject to:} \quad \begin{cases} \vec{x}_{t+1} = A\vec{x}_t + B\vec{u}_t & \text{(system dynamics)} \\ \vec{u}_{\min} \le \vec{u}_t \le \vec{u}_{\max} & \text{(input constraints)} \\ \vec{x}_{\min} \le \vec{x}_t \le \vec{x}_{\max} & \text{(state constraints)} \\ \|\vec{p}_{\text{obs}} - \vec{p}\|_2^2 \ge r^2 & \text{(obstacle avoidance)} \end{cases} \tag{2}$$

Where:

- $\vec{F_{att}} = \frac{1}{2} k_{att} (\vec{p} - \vec{p_{goal}})$
- $\begin{cases} \vec{F_{rep}} = \frac{1}{2} k_{rep} (\frac{1}{\rho(\vec{p})} - \frac{1}{\rho(\vec{p_{lim}})})^2 & \text{if} \rho(\vec{p}) \le \rho(\vec{p_{lim}}) \\ 0 & \text{if} \rho(\vec{p}) > \rho(\vec{p_{lim}}) \end{cases}$
- $\vec{p}$, $\vec{p_{\text{goal}}}$, $\vec{p_{\text{obs}}}$ are the positions of the drone, the target position and the target position respectively.
- $\rho(\vec{p})$ is the distance of point $\vec{p}$ from the obstacle position.
- ρ_{lim} is the threshold distance.
- Q, R, W_{rep}, W_{atr} are weight matrices.

IV. RESULTS

Experiments consist of a series of waypoint navigation in the presence of varying number of obstacles from zero (baseline) to 13 spherical obstacles in a volume of 5x5x5 m^3 workspace in PyBullet simulation environment [2]. Additionally, to measure the robustness of the system, the tasks are performed with the presence of additive Gaussian noise to the states. The implementation is done using the ROS2 framework in the python3 environment with ACADOS [1] as the MPC solver. The code-base is made available in the GitHub repository[1]. In each scenario, experiments are performed by running 10 trials. Moreover, tests were also performed in a simulated building construction environment.

A. Evaluation metrics

The performance of the system is evaluated based on the metrics as follows

1) Success rate
2) Waypoint tracking error
3) Control effort
4) Instantaneous velocity
5) Computation time
6) Distance to closest obstacle
7) Average Visibility

[1]https://github.com/AntreasK24/RO47005-ProjectGitHub repository

1) Success Rate: A successful navigation to a waypoint is defined when the quadrotor reaches the target way-point [3]

- When the quadrotor does not hit an obstacle
- Reached the target position within a timeout period from the initial position
- Reached the target position with a maximum error tolerance of 0.3 m

2) Waypoint tracking error: This metric evaluates the accuracy of reaching the target state. It is calculated by finding the mean of the L2 norm between the target waypoint and the current position over a settling period of 5 seconds.

3) Control effort: The control effort describes the amount of control input (accelerations) that has to be exerted in order to reach the target waypoint.

$$\text{control effort} = \int_0^T |u(t)| \, dt$$

where T is the total travel time

4) Computation time: It is defined as the time required for the MPC solver to solve the optimization at a discrete time step k.

5) Distance to closest obstacles: At each time step during navigation, it is defined as the distance from the current drone position to the nearest occupied space, i.e. the surface of the nearest sphere. This metric is averaged over all points on the path. The higher the value, the further away the drone is from obstacles, implying a better path for waypoint navigation.

6) Average visibility: Average visibility at a time step is defined as the average of the distances to an obstacle around the drone position [4]. This metric captures the closeness of the quadrotor to the obstacles. The smaller the value, the tighter the navigation.

Note that the latter two metrics are inspired and an extension of the state-of-the-art ground navigation metrics [4] in $\mathbb{R}^3$ configuration space for the quadrotor.

From Fig 4, it is observed that mean computation time increases over as the number of obstacles (constraints) are increased. It is also noted that noise to the state exacerbates the time by an average offset of 11 seconds.

When compared the control efforts with noise added to the velocity state of the quadrotor, it is noticed that the quadrotor exerts lesser acceleration in the form of inputs as depicted in Fig 5. Additionally, this behavior is blatant in Fig 7, despite the ensembles velocities across different scenarios remaining around a mean of 5 m/s and 7 m/s for without and with state noise respectively.

The quadrotor is able to precisely to reach the goal position with a mean waypoint tracking error of less than 0.1 m in a 3-obstacle environment. Notably, the waypoint tracking error has a very minimum standard deviation, implying the repeatability of the navigation system as depicted in Fig 6.

B. Target position

The first and simplest experiment conducted was to see if the robot could reach an arbitrary target position in an environment with obstacles. Using the random trajectory

generator, the drone is given random positions in a 3D world of spherical objects.

C. Computation time

To measure the computation time required to set up the solver. A random trajectory following task was performed in 3 different scenarios, first with 10 randomly placed spheres, then with a small building consisting of 50 spheres representing columns, and finally on a large building with 113 spheres. The results are shown in the figure 4.

D. Random Sphere Experiment

This experiment was designed to test the robustness of the obstacle avoidance of the MPC. Using the first scenario the drone was tasked to reach 10 random positions in environments consisting of 5, 10 and 15 spheres. This experiment was repeated 5 times with new randomly places spheres each time. The results of the experiment can be seen in figure 3.

E. Closest Obstacle Distance

This experiment aims to verify that the drone maintains a safe distance from nearby obstacles during flight. The number of random obstacles, both with and without state noise for each scenario. The data plot in figure 8 indicates a general decrease in the mean closest distance as the number of obstacles increases from around 1.5 m to 1.25 m. The presence of state noise introduces additional uncertainty, as demonstrated by the wide range of observed distances.

F. Average Visibility

This experiment confirms that the drone maintains a safe distance from obstacles, with average distances measured at each time step. The random obstacle were varied, both with and without state noise. From figure 9 as obstacle numbers increase, so does the mean distance, indicating effective avoidance due to increased repulsion from 2.2 to 2.5 m. State noise adds uncertainty, causing drone to fluctuate in proximity to obstacles.

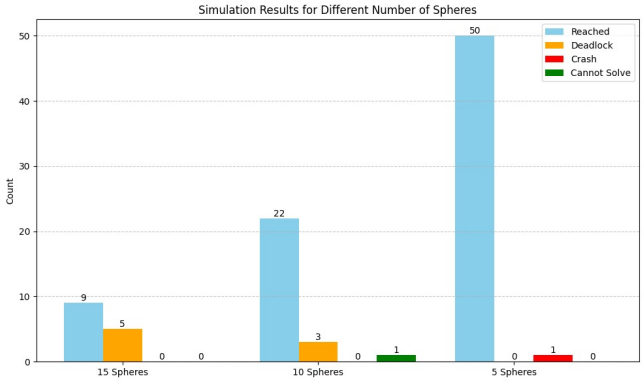

Fig. 3. Results from random sphere experiment

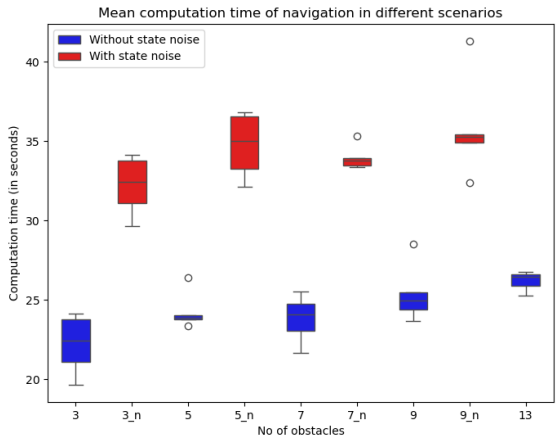

Fig. 4. Mean computation time of navigation in different scenarios

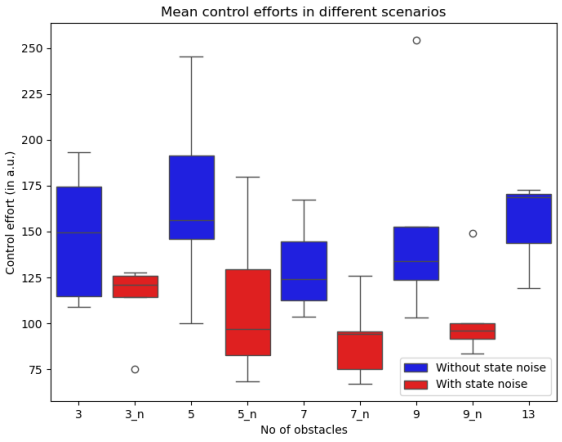

Fig. 5. Mean control efforts in different scenarios

V. DISCUSSION

It is clear from the computation time required to solve the problem for a single point on a medium sized building that this approach may be too slow for real life applications. By editing the solver to include the trajectory definitions, the computation time can be reduced as the solver does not need to be set up each time the drone reaches a target position. Another way to reduce computation time is to introduce new types of obstacle constraints other than spherical ones. For example, using a convex region in convex space will help to represent obstacles as cuboids instead of representing everything as spheres. Apart from editing the problem definition, if the environment is expected to be static, then the trajectories can be computed offline, so that no time is spent setting up the problem online.

It was noticed that when using the potential field method, the drone would sometimes get stuck in a local minimum if the obstacles were too close together. This was more prevalent in the first scenario due to the random placement of obstacles. This can be seen from the results of the random

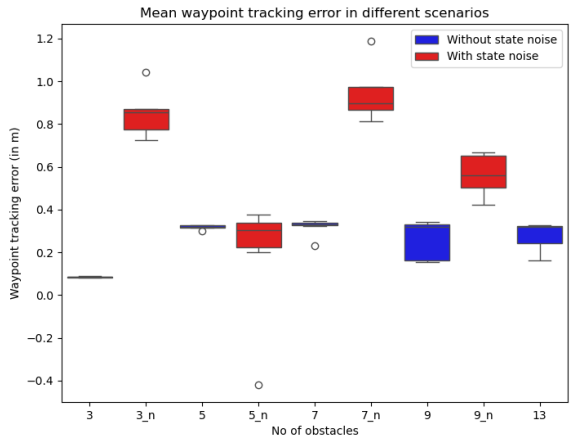

Fig. 6. Mean waypoint tracking error in different scenarios

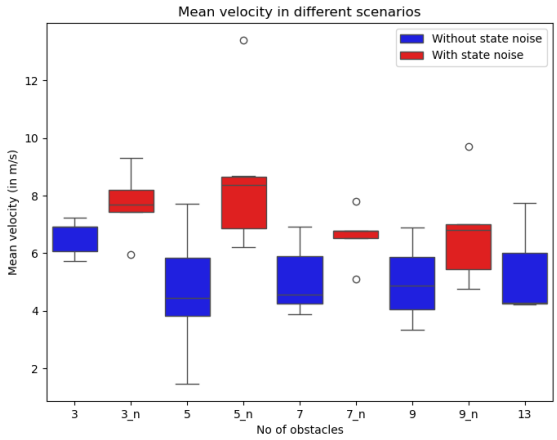

Fig. 7. Mean velocity in different scenarios

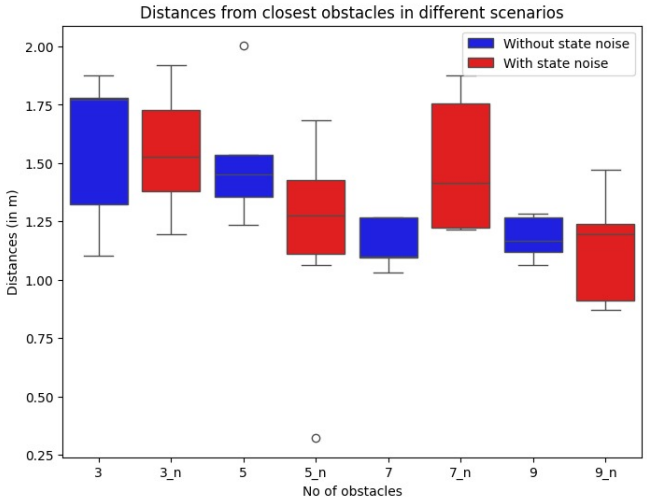

Fig. 8. Closest obstacle distance

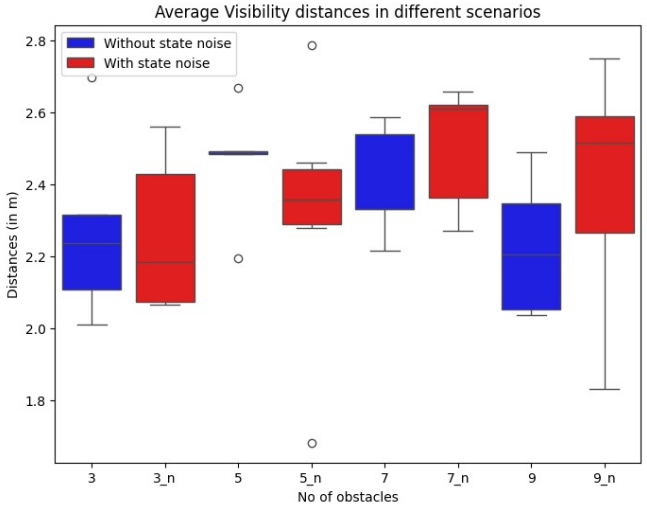

Fig. 9. Average visibility plot

sphere experiment. As the number of obstacles increased, so did the number of dead ends, resulting in a decrease in the number of points reached. It is also worth noting that increasing the number of obstacles does not seem to have an effect on the solver failing to solve the problem or the drone crashing into an object. Although a deadlock is not a desirable outcome, it is offered as an alternative to a crash because it does not cause any damage to the hardware.

During the second scenario local minima were more rare. As the second scenario is more representative of our use case the potential fields were kept.

Future Improvements

By using a more accurate model that includes the orientation of the drones, more accurate results and behavior can be achieved. Such a model has been derived following the work of [5], [6], where attempting to implement the model in the solver would give unrealistic solutions to the optimization problem.

While static obstacles may be sufficient for some tasks, dynamic obstacles are expected in an active construction site. Introducing constraints that can handle dynamic obstacles will improve the robustness of the solver.

REFERENCES

[1] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados – a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, vol. 13, no. 4, pp. 563–589, 2021, doi: 10.1007/s13162-021-00249-9.

[2] J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok, and A. P. Schoellig, "Learning to Fly—a Gym environment with PyBullet physics for reinforcement learning of multi-agent quadcopter control," *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 7512-7519, doi: 10.1109/IROS51168.2021.9635857.

[3] Brunke, Lukas, et al. "Safe learning in robotics: From learning-based control to safe reinforcement learning." Annual Review of Control, Robotics, and Autonomous Systems 5.1 (2022): 411-444.

[4] Perille, Daniel, et al. "Benchmarking metric ground navigation." 2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR). IEEE, 2020.

[5] Foehn, Philipp, et al. "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight." Science robotics 7.67 (2022): eabl6259.
[6] Foehn, Philipp, Angel Romero, and Davide Scaramuzza. "Time-optimal planning for quadrotor waypoint flight." Science robotics 6.56 (2021): eabh1221.