ESAQ Planner
Environmental and System Aware Quadrotor Motion Planning

Aniket Gujarathi[1]

***Abstract*— Autonomous navigation of quadrotors in tight and constrained spaces is a well-studied problem. However, risk-aware navigation at high speeds still remains an open challenge. Most of the existing literature rely on robust control approaches to solve the problem of trajectory planning for quadrotors, which leads to a conservative solution. Furthermore, the dynamics models used in these formulations do not include the stochasticity present in the system due to noise, aerodynamics, and actuator dynamics. Such a formulation fails to capture the actual dynamics of the system while planning. In this project, we propose an environmental and system aware motion planning framework. We propose an environmental aware hierarchical MPC formulation that assures smooth and collision free trajectory planning in constrained environments. To incorporate system awareness in the formulation, we use a multi-layered neural network to estimate the system dynamics of quadrotors.**

Fig. 1: Composite image of a fast flight experiment in cluttered environment [1]

I. INTRODUCTION

Quadrotors are widely used in problems such as mine exploration, search and rescue, drone delivery due to their distinctive agility and flexibility. These problems require quadrotors to move safely in constrained spaces while aiming for an optimal policy to reach the goal. Many motion planning methods for quadrotor navigation undertake a conservative approach to guarantee safety throughout the navigation. Although safety is an important criteria in robotics, it is not always optimal to act conservatively, especially in time critical scenarios such as search and rescue operations.

Let us take an example of a search and rescue operation performed by a quadrotor in a forest environment. When the quadrotor is moving closer to the trees or objects, it must move less aggressively in order to assure safety. However, while moving in an empty field in the forest or outside the dense tree zones, the quadrotor can fly at higher speeds and aggressiveness to save time and perform the task in a time efficient manner. Such a risk-aware adaptive planning strategy can help in generating efficient trajectories and velocity profiles for the quadrotor.

Some works like [1], [2], [3] mention the importance of such an adaptive safety-aware framework for motion planning of quadrotors. However, these approaches assume the quadrotor to be a particle due to the property of differential flatness of quadrotors [4]. Due to this assumption, the planners assume perfect tracking of the states through a d^{th} order integrator model. Such an assumption may not hold true while performing aggressive quadrotor movements where there can be a discrepancy between the lower controller and the planner outputs. Assuming perfect tracking, the planner may output ideal future states that may not be feasible for the lower controller, or the lower controller may be delayed in reaching the desired states given by the planner. In simpler words, the planner is generally not aware of the system dynamics and relies on the lower controller to handle the stochasticity.

In this project, we propose an environmental and system aware motion planning framework for quadrotor navigation. We use [2] as our baseline approach to formulate an environmental aware motion planning framework. EVA-planner takes a modular approach to generate smooth, collision free, and kinodynamically feasible trajectory. To incorporate system awareness in the framework, we propose using a multi-layer neural networks as a dynamics model which we use in the MPC formulation.

The rest of the report is organized as follows: Section II explains briefly the literature needed to understand the problem. Section III explains the methodology of our problem formulation. Section IV goes through the experiments done and the frameworks used for the experiments. Section V explains the prospective future work that can be done in this project.

II. PROBLEM STATEMENT

The goal of this project is to develop a planning architecture for quadrotor navigation that can adaptively replan trajectories quickly using an environmental-aware and system-aware hierarchical MPC formulation. To understand the problem, we need to understand some basics of motion

*This work is a part of the course project for ECE1643

[1]Aniket Gujarathi is with the University of Toronto Institute of Aerospace Studies, Toronto, Ontario. aniket.gujarathi@mail.utoronto

planning algorithms for quadrotors, model-predictive control, and the basics of a Euclidean Signed Distance Field (ESDF).

A. Quadrotor Motion Planning

Autonomous navigation of quadrotors generally includes use of sensors and algorithms that help the robot to perceive its environment and make informed decisions in order to reach a fixed goal. Motion planning is generally divided in two parts: direct and hierarchical methods. Direct methods of motion planning use graph-based algorithms to find a path from the starting point to goal position. These direct methods include algorithms such as the A* [5] algorithm, which is a type of path-finding algorithm that takes into account the distance between the quadrotor's starting point and goal, as well as the cost of moving through the environment. Other direct methods for motion planning include sampling-based algorithms, such as the rapidly-exploring random tree (RRT) [6].

In addition to direct methods, there are also hierarchical methods for motion planning for quadrotors. These hierarchical methods break down the problem in several manageable subproblems and solve them separately. This can be useful for dealing with complex environments that may be difficult to navigate using a direct motion planning algorithm. Our baseline [2] employs a hierarchichal planning strategy wherein a direct method like A-star is used to provide a global path. Once we get the global guiding path, a low-level MPC is used to smoothen the guiding path. After getting the smooth trajectory, a high-level MPCC (Model Predictive Contouring Control) is used to generate a local trajectory by optimizing the safety constraints, tracking error, aggressiveness.

B. Model Predictive Control (MPC)

Model Predictive Control (MPC) is a type of control algorithm that involves optimizing a control law over a finite time horizon by solving an optimization problem at each time step. The optimization problem is typically formulated as a constrained optimization problem, where the objective is to minimize a given cost function and the constraints are the system's dynamics and any additional constraints on the control inputs. We will explain our MPC formulation in detail in Section III, however, the basic MPC formulation can be explained mathematically as follows:

$$
\begin{aligned}
&minimize\ J = x'Qx + u'Ru \\
&subject\ to:\ x[k+1] = f(x[k], u[k]),\ k = 1, 2, ..., N-1 \\
&and\ x[0] = 0, \\
&u[min] <= u[k] <= u[max], for\ k = 0, ..., N-1
\end{aligned}
$$

where x is the system state, u is the control input, Q and R are the weighting matrices, f is the system's dynamics, and N is the length of the time horizon. The optimization problem is solved at each time step, and the resulting control law is applied to the system. This process is then repeated at the next time step, using the updated system state as the initial condition.

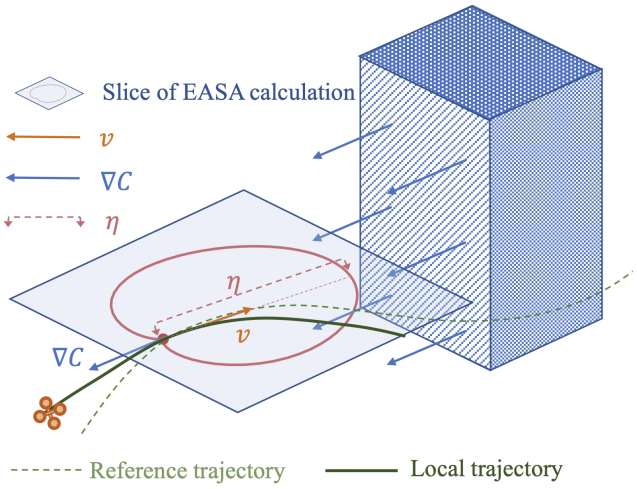

Fig. 2: Environmental Adaptive Safety Aware calculation during trajectory optimization [2]

C. Euclidean Signed Distance Field (ESDF)

An ESDF (Euclidean Signed Distance Field) is a data structure that is used to represent the distance from a point in space to the nearest obstacle or other relevant feature in the environment. To create an ESDF, the space is first discretized into grids, and each cell contains information about the signed distance to the nearest obstacle. In simpler terms, for a point in space, the ESDF constructs a real-valued scalar field that represents the distance to the point in the space. Mathematically, the ESDF can be represented as a function $f(x, y, z)$ that maps a point in space (x, y, z) to the signed distance to the nearest obstacle at that point. If the point (x, y, z) is outside of an obstacle, the distance is positive and is equal to the shortest distance from the point to the boundary of the obstacle. If the point is inside an obstacle, the distance is negative and is equal to the negative of the shortest distance from the point to the boundary of the obstacle. If the point is on the boundary of the obstacle, the distance is zero. For our use case, we will use such a representation to calculate the gradient of the ESDF at a particular point wrt the obstacles and formulate an environmental aware mechanism to attribute risk.

III. Methodology

The main objective of this work is to develop a hierarchical MPC formulation that can be used for fast and aggressive trajectory replanning by quadrotors moving at high speeds through constrained environments.

A. Environmental Adaptive Safety Aware (EASA)

First, let us discuss about the Environmental Adaptive formulation used in [2] that we are going to follow in this project.

EASA regulates the flight behaviour of the quadrotor based on the angle between the velocity and gradient of the Euclidean signed distance field (ESDF). EASA corresponds

each pair of $\{v,\ \nabla c\}$ to a risk weight η which is then used to control the quadrotor's movements based on the risk.

$$\eta(\beta) = \frac{2}{1 + e^{\alpha\beta}}$$
$$\beta = \frac{< v, \nabla c >}{||v||||c||}$$

where α is the rate change coefficient. As explained in [2], the situation where $\beta \in (0, 1]$ is considered as safe, and $\beta \in [-1, 0]$ is considered as dangerous. η is large when a quadrotor flies closer to an obstacle, thus informing the optimizer about the risk and that it needs to fly cautiously. Whereas, when the quadrotor seems to leave an obstacle, the value of η decreases informing the controller that it is out of a danger area and can increase the speed. Fig. 2 shows the discussed EASA formulation when the quadrotor approaches an object.

B. Low-level MPC

Before getting into the low-level MPC, the first step in the planner is to generate a global guiding path. [2] uses an A-star based planner to find the global path, however, we can use any of the standard path-finding algorithms such as RRT, RRT-star, etc. Once we get the guiding path, the job of the low-level MPC is to optimize the geometry of the reference trajectory with reduced computational resources. The objective function is defined as:

$$min\ \kappa_1 J_s + \kappa_2 J_c + \kappa_3 J_u \tag{1}$$

, where J_s is the similarity penalty of the distance between the trajectory and the guiding path, J_c is the collision cost, and J_u is the smoothness term. The low-level MPC is explained in detail in the baseline paper [2]. For this project, we do not make any changes to the EASA and the Low-level MPC formulation, as these formulations are sufficient to handle our use-case and do not need any additional changes to incorporate the system-aware module. Our main contribution lies in the neural-network based dynamic model of the system and its inclusion in the High-Level MPCC to propagate the future states appropriately.

C. Neural-Network Dynamics

Although neural-network based approaches are seen to be dominating the research in robotics nowadays, traditional methods work just as robustly and reliably without loss of generality in the domain of motion planning and controls. A hybrid approach to combine learning-based approaches with traditional methods seem to provide better results in these domains. Instead of generating an end-to-end learning based architecture to solve a complex controls problem, we can focus on solving specific sub-problems using the data-driven approaches to try to enhance the performance of well established traditional algorithms. In this project, we propose to use data-driven approach to develop a neural-network based dynamic model of the system.

Inspired from [7], we will try to incorporate a purely learned system model in our environmental aware MPC formulation. By using a learning based system identification, we can accurately and efficiently model a quadrotor's behaviour, taking into account stochastic factors such as aerodynamics, actuator dynamics, and sensor noise. [7] only learns the acceleration of the system. Given the following state transition:

$$x_{k+1} = F(x_k, u_k) = \begin{bmatrix} p_k + \dot{p}_k \Delta k \\ \dot{p}_k + f(x_k, u_k)\Delta k \end{bmatrix} \tag{2}$$

, where $x = (p, \dot{p})$ with p being the configuration of the system, and k is the timestep. We train f with a fully-connected multi-layer neural network based on a dataset of state-action-acceleration triplets.

To generate the data, we need to perform experiments using real hardware and train the neural network using the collected data. [7] uses a bootstrapping approach to collect dataset similar to a DAgger [8]. At each iteration, the imitative controller is used to generate control actions for a given input, and the expert is queried to provide the correct action for that input. This expert-provided action is added to the demonstration data, and the controller is retrained using this augmented data. This process is repeated until the controller's performance reaches a satisfactory level.

1) High-Level MPCC: Once we have our model of the system, the next step would be to include the model in our MPCC formulation. This final layer is responsible of generating collision free local trajectory while optimizing the safety constraints, tracking error, flight aggressiveness, and dynamic feasibility. The jerk is taken as input and the state consists of $p_\theta, v_\theta, a_\theta$ reference points that follow the reference trajectory. The objective function is defined as:

$$min\ \lambda_1 f_s + \lambda_2 f_p + \lambda_3 f_e + \lambda_4 f_c + \lambda_5 f_d + \lambda_6 f_m \tag{3}$$

, where f_s is the tracking error, f_p indicates progress of the reference points, f_e represents the EASA error, f_c is the collision cost, f_d is the penalty for violating kinodynamic feasibility, and f_m is the propagation error using the model learned from the neural network. λ_i are the weights of these terms. f_s and f_p are the factors used to make the trajectory fast while ensuring low tracking error. f_e term is used to control the speed of the robot depending on the risk factor from the EASA calculations. In the absence of risk, the terms f_s and f_p are the dominant factors to determine the trajectories, whereas in the presence of risk, f_e is used to vary the speed of the quadrotor to handle the risk assigned to the state. With the addition of f_m, the optimizer is now aware of the system dynamics and thus propagates appropriate states in the future timesteps that can be handled by the lower controller. Without the f_m component, the system acts as an ideal integrator model for future propagation of states that does not account for the stochastic changes in the system. However, with the addition of the term, the MPCC is now more aware of the system's capabilities and the generated trajectories. This approach helps in bridging the gap between the planner outputs and the lower controller's ability to follow them accurately.

$$f_s = \sum_{i=1}^{N} ||p_i - p_{\theta,i}||^2 \tag{4}$$

$$f_p = -\delta t \sum_{i=1}^{N} v_{\theta,i} \tag{5}$$

$$f_e = \sum_{i=1}^{N} \eta(\beta) F_c(c(p_i))(||v_i|| - v_{thr})^2 \tag{6}$$

$$f_c = \sum_{i=1}^{N} F_c(c(p_i)) \tag{7}$$

$$f_d = \sum_{i=1}^{N} (F_d(v_i) + F_d(a_i) + F_d(j_i) + F_d(v_{\theta,i})) \tag{8}$$

$$f_m = \sum_{i=1}^{N} ||v_i - v_{i-1} - f(x_i, u_i)\Delta t||^2 + ||p_i - p_{i-1} - v_i \Delta t||^2 \tag{9}$$

Eq. 4-8 can be referenced in [2] for more details. In the f_m term, we query the acceleration obtained by the neural network to propagate the next velocities and consequently the position. Given that we learn the acceleration of the system using the neural network, we need an initial guess using the default integrator system. From the subsequent timesteps, we can query the model to generate trajectories using an appropriate learned model. Hence, the trajectory at the first timestep will not be representative of the actual system, but will default to the learned model from subsequent timesteps.

IV. EXPERIMENTAL SETUP

Although we weren't able to test the proposed system due to time and resource constraints, we carried out a few experiments using our baseline approaches. For the experiments we use ROS - Robot Operating System and visualize in RVIZ. In this section, we will give a few details about the framework we tested on and some qualitative results.

A. ROS

ROS is a free and open-sourced framework for robotics. It contains a set of software-libraries and tools for robotics applications. Different modules in a robotics application are controlled by nodes in ROS. Each node is responsible for a single module in the application. Communication between various nodes is enabled through topics, services, actions and parameters. Communication in ROS takes place through a publisher and subscriber strategy. For example, suppose two nodes need to communicate between each other through some ros topics, wherein the output of node 1 is used by node 2 for another task. In such a case, first the nodes are launched. Once node 1 completes it's task and node 1 is ready to send the message through a ros topic to node 2, it publishes the topic at a described frequency. This published topic is then subscribed by node 2, which uses the information to complete its task. This is a simple example of communication using ros-topics, but there are other ways to enable communication between ros nodes too. Fig. 3 shows

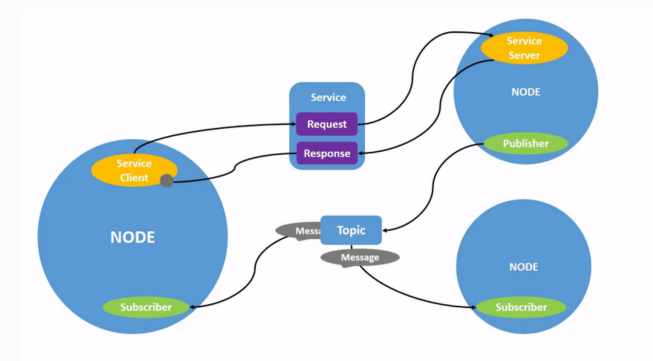

Fig. 3: ROS node graph

the different ways ros nodes can communicate. ROS provides extensive documentation and tutorials if one needs to go deeper in this domain[1].

B. Baseline Experiments

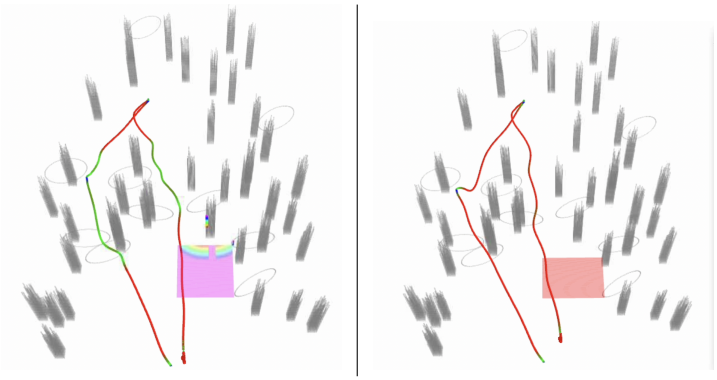

TABLE I: Qualitative experiment with and without EASA visualized in RVIZ

For this project, we used the ROS Noetic version on a UBUNTU 20.04 machine. We used and modified the code provided by [2][2]. For the environment, we generate a $40m \times 5m$ random forest map consisting of poles and hoops as obstacles. The quadrotors are equipped with depth cameras for perception and are used to generate the ESDF map of the environment. The velocity limit of the quadrotor is set to $3m/s$.

Fig. I, shows the environment and the path taken during two experiments. The left image, shows the trajectory taken by the quadrotor with EASA activated. The red path shows that the quadrotor is travelling at fast speeds, whereas the green trajectory indicates slower speeds. As one can see in the figure, the quadrotor seems to slow down near any obstacles and increases the speed once it leaves the obstacle region. This behavior is in line with the methodology discussed in Section III.

On the other hand, the right image in Fig. I shows the experiment without EASA activated. We can clearly see

[1]https://docs.ros.org/en/foxy/Tutorials.html
[2]https://github.com/ZJU-FAST-Lab/EVA-planner

that the quadrotor does not change its speed throughout the experiment. Such a behavior is not safe or optimal with quadorotors moving at high speeds. These experiments show us how being environmentally aware can help in appropriate navigation of quadrotors in cluttered environments.

As seen in the figure, the quadrotor is assumed to be a point-particle and the trajectory is replanned using a d^{th} order integrator model as the system model. Although, this approach works well in the simulations and real-world experiments carried out by [2], we can see during these experiments that the controller takes time to shift to a new plan when the goal position is changed and the trajectory has to be re-planned. Integrating our learned model in the planning architecture can help in reducing the delay caused due to the lower controller catching up to the generated trajectories by the high level controllers.

V. Conclusion and Future Work

In this work we highlight the importance of having an environmental and system aware motion planner for quadrotors. We proposed a hierarchical planner with two layers - a low-level MPC and a high-level MPCC. The low-level MPC is used to generate smooth trajectories, whereas the high-level MPCC is responsible for adhering to the safety and kinodynamic constraints. Further, we proposed the addition of a purely learned dynamics model of the system in our MPC formulation to make the planner aware of its system capabilities. A multi-layered neural-network was used to provide a model of our system based on the real-world data provided. This learning-based model not only generates a data-driven model, but also captures the stochasticity in the system such as the actuator dynamics, sensor noise, etc. We saw how our hierarchical planning architecture uses information about the environment to make informed decisions around obstacles. This planner establishes a trade-off between a safe solution when in the vicinity of obstacles and acts more freely in open-spaces. Overall, it is clear that the development of effective safety-aware motion planning algorithms is critical for the successful deployment of quadrotors in a wide range of applications. Further research is needed to develop algorithms that can handle more complex environments and provide guarantees of safety and performance.

Future work could include performing intensive experiments based on the proposed formulation. Further, this work does not look into handling dynamic obstacles or multi-agent systems. An interesting extension to the project would be to incorporate a multi-agent distributed MPC performing risk-aware navigation in cluttered environments while handling dynamic obstacles.

References

[1] B. Zhou, J. Pan, F. Gao, and S. Shen, "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight," 07 2020.

[2] L. Quan, Z. Zhang, X. Zhong, C. Xu, and F. Gao, "Eva-planner: Environmental adaptive quadrotor planning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 398–404.

[3] Z. Li, Arslan, and N. Atanasov, "Fast and safe path-following control using a state-dependent directional metric," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6176–6182.

[4] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525.

[5] Peter Hart and Nils Nilsson and Bertram Raphael, "A formal basis for the heuristic determination of minimum cost paths."

[6] S. M. LaValle, "Rapidly-exploring random trees : a new tool for path planning," *The annual research report*, 1998.

[7] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic mpc for model-based reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1714–1721.

[8] S. Ross, G. Gordon, and J. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," *Journal of Machine Learning Research - Proceedings Track*, vol. 15, 11 2010.