Obstacle driven RRT implementation for quad-rotor motion planning

Benjamin Bogenberger ([redacted]), Simon Gebraad ([redacted]),
Crina Mihalache ([redacted]), Andrei-Carlo Papuc ([redacted])

***Abstract*— This paper presents a new implementation of an RRT algorithm for a quad-rotor called biased RRT, which is compared numerically to existing sampling based algorithms. This global path planner is then combined with MPC to create a complete offline-online implementation and tested in multiple simulated environments. Results are promising but also provide various pathways for further research.**

I. INTRODUCTION

The article describes the implementation of a motion planning algorithm for a quad-rotor in an indoor environment. The use of quad-rotors in confined spaces, such as for quick delivery of small objects in hospitals, poses unique challenges due to the high density of obstacles present in indoor environments.

The current state of the art employs offline-online computation, classification through machine learning, sampling-based motion planning and trajectory smoothing, achieving real-time, kinodynamic high-speed planning for quad-rotors in dynamic environments [2] [13]. Sampling-based planning is used in many theoretical implementations [10] [12] and has even been shown to work experimentally [2].

However, in high-speed dynamic environments, computation time is often a constraint for motion planning algorithms. To address this issue, the article focuses on developing a variation for indoor environments of a 3-Dimensional Rapidly Exploring Random Tree (3D RRT), called 'biased RRT', that improves computation times by increasing the sampling rate near obstacles. This method built upon previous work done on RRT* [9] and informed RRT* [7], and code from [6] served as inspiration. Nonetheless, the algorithm was implemented from scratch.

The proposed algorithm simplifies non-kinodynamic planning, a state-of-the-art method, for two reasons: time constraints in developing a kinodynamic algorithm and the increased computational expense of kinodynamic RRT* as described by [10], compared to non-kinodynamic algorithms. However, this simplification means that the dynamics of the quad-rotor are not incorporated. Therefore, to address this limitation, this article also describes how the offline RRT algorithm can be combined with an online local motion planner: Model predictive control (MPC). This combination creates a full-stack motion planning algorithm with offline and online components that ensures the dynamics of quad-rotors can be incorporated. The resulting algorithm is similar to the one described in [11] with the addition that the proposed method in this article includes more variation in the shape and the placements of the obstacles. However, it has its limitations, as described in the final chapter.

The outline of this article is as follows: First, the dynamics of the robot model are introduced. Then, the motion planning implementation's specifics are discussed regarding RRT and MPC. Subsequently, the new RRT implementation is compared numerically to RRT and PRM. Additionally, the RRT-MPC combination is tested in simulation and compared to the original path generated by RRT. The final chapter presents the implications of the results, limitations and opportunities for further research.

II. ROBOT MODEL

The quadrotor model used in this work is a slight adaptation of the model presented by [5] and [8] – the gravitational force, as well as the rotor forces, are defined in reverse, as shown by Figure 1, to conform to the later used simulation environment. Furthermore, the drag forces used by [8] are not considered in this initial implementation. In addition, the following two reference frames are defined respectively, the world frame **O** and the body frame **C**. We select this model as it was shown to be suitable for linear model predictive control (MPC) [8]. The following model is linearized and time discretized in a second step.

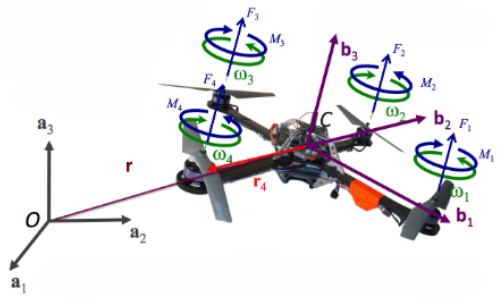

Fig. 1. Preliminary representation of quad-rotor dynamics

The reader is referred to [5] for more detailed derivations. The four control inputs are defined in matrix form u based on the rotation speeds of each rotor $\Omega_{1,2,3,4}$. They are responsible for the upwards thrust, roll, pitch and yaw rotation, respectively.

$$\underbrace{\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}}_{u} = \underbrace{\begin{bmatrix} K_f & K_f & K_f & K_f \\ 0 & -K_f & 0 & K_f \\ K_f & 0 & -K_f & 0 \\ K_m & -K_m & K_m & -K_m \end{bmatrix}}_{=\mathbf{K}} \underbrace{\begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix}}_{\Omega} \tag{1}$$

K_f is the aerodynamic force constant with force $F_i = K_f \Omega_i^2$ and K_m the moment constant with moment $M_i = K_m \Omega_i^2$. Furthermore, the state vector is selected as

$$\underline{x} = [\underbrace{x \;\; y \;\; z}_{\underline{r}} \;\; \underbrace{\dot{x} \;\; \dot{y} \;\; \dot{z}}_{\dot{\underline{r}}} \;\; \phi \;\; \theta \;\; \psi \;\; \underbrace{p \;\; q \;\; r}_{\underline{\omega}}]^T \tag{2}$$

where the first six elements are translational position $\underline{r}$ and velocity $\dot{\underline{r}}$ in the world frame $\mathbf{O}$. The latter six represent the orientation of the drone using Euler angles where ϕ θ ψ describe rotations in body frame $\mathbf{C}$ around axes x,y,z, respectively and their time corresponding rate of change using the angular body rates p, q, r. They relate to the Euler rates with the transformation

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{3}$$

The translational equations of motion given by $m\ddot{\underline{r}} = \begin{bmatrix} 0 & 0 & mg \end{bmatrix}^T + \mathbf{R}(\phi,\theta,\psi) \begin{bmatrix} 0 & 0 & -u_1 \end{bmatrix}^T$ are now

$$\begin{aligned} \ddot{x} &= \frac{u_1}{m}(\sin\phi\cos\psi + \cos\phi\cos\psi\sin\theta) \\ \ddot{y} &= \frac{u_1}{m}(\cos\phi\sin\psi\sin\theta - \sin\phi\cos\psi) \\ \ddot{z} &= -g + \frac{u_1}{m}(\cos\phi\cos\theta). \end{aligned} \tag{4}$$

Similarly, the rotational equations of motion given by $\mathbf{J}\dot{\underline{\omega}} + \omega \times \mathbf{J}\underline{\omega} + \underline{\omega} \times \begin{bmatrix} 0 & 0 & J_r\omega_r \end{bmatrix}^T = \begin{bmatrix} -lu_2 & -lu_3 & -u_4 \end{bmatrix}^T$ are

$$\begin{aligned} \dot{p} &= -\frac{l}{I_x}u_2 - \frac{J_r}{I_x}q\omega_r + \frac{I_y - I_z}{I_x}qr \\ \dot{q} &= -\frac{l}{I_y}u_3 - \frac{J_r}{I_y}p\omega_r + \frac{I_z - I_x}{I_y}pr \\ \dot{r} &= -\frac{l}{I_z}u_4 + \frac{I_x - I_y}{I_z}pq, \end{aligned} \tag{5}$$

where I_x, I_y, I_z are the area moments about the axes of the body frame C, l is the arm length of the quadrotor, and $\omega_r = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4$ is the relative rotor speed between clock- and counter-clockwise rotating rotors.

The non-linear state space representation $\dot{\underline{x}} = \underline{f}(\underline{x}, \underline{u})$ can now be constructed using equations (3), (4) and (5). Finally, this system is linearized and time discretized to be compatible with the later presented linear MPC controller. The operating point must satisfy the hover condition [8] $\ddot{z} = 0$, thus, the operating point is selected as $\begin{bmatrix} \underline{x}_{op}^T & \underline{u}_{op}^T \end{bmatrix}^T$ with $\underline{x}_{op} = \underline{0}$ and $\underline{u}_{op} = \begin{bmatrix} mg & 0 & 0 & 0 \end{bmatrix}^T$. The linearized time discretized system with time step t_s is now given by

$$\begin{aligned} \mathbf{A} &= \mathbf{I} + t_s \frac{\partial \underline{f}(\underline{x})}{\partial \underline{x}}|_{\underline{x}=\underline{x}_{op}, \underline{u}=\underline{u}_{op}} \\ \mathbf{B} &= t_s \frac{\partial \underline{f}(\underline{x})}{\partial \underline{u}}|_{\underline{x}=\underline{x}_{op}, \underline{u}=\underline{u}_{op}} \\ \underline{x}_{k+1} &= \mathbf{A}\underline{x}_k + \mathbf{B}\underline{u}_k \end{aligned} \tag{6}$$

In this model, the workspace is all points in $\mathbb{R}^3$. Configuration space is $\mathbb{R}^3 \times \mathbb{S}^3$, so $\mathbf{q} = [\underbrace{x \;\; y \;\; z}_{\underline{r}} \;\; \underbrace{\psi \;\; \theta \;\; \phi}_{\underline{\alpha}}]$ where $\underline{r}$ describes position and $\underline{\alpha}$ orientation. As quad-rotors are under actuated with respect to their configuration there are differential constraints on their movement, meaning they are non-holonomic systems.

III. MOTION PLANNING: ALGORITHM

This section proposes and evaluates a new variant of RRT*. This variation, called 'Biased RRT', makes use of the fact that indoor path planning often involves obstacles in the path between the start and goal, such as when having to go around a corner. Hence, it would seem beneficial to sample points close to obstacles if one wants to find the shortest path.

The biased RRT algorithm is an adaptation of RRT, so it is similar to the formulations in [9] and [7]. However, the random sampling of points is modified as described by the pseudocode in Algorithm 1.

Algorithm 1 Sampling with bias in the vicinity of obstacles

```
select = random()
if select < β then
    x_rand ← random point within radius R of obstacle
else
    x_rand ← random point in workspace
```

The algorithm modification is based on two parameters: bias β and radius R. The former describes which fraction of samples is biased, whereas the latter describes which radius around obstacles is sampled. A simple example in 2D is shown in Figure 2. Table I gives a basic overview of all variables involved in the proposed RRT algorithm.

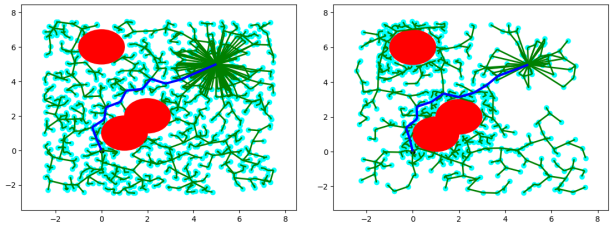

Fig. 2. Implementation in 2D of the obstacle-biased RRT* (right) versus normal RRT* (left)

To perform testing of the proposed algorithm, three base algorithms were developed from scratch: RRT, optimal RRT* as described by [9], and informed RRT* as described by [7]. Additional 'biased' variations of each algorithm were also created, resulting in six variations. Finally, an algorithm for Probabilistic Road Map (PRM) was also developed to compare the biased RRT implementation with a different sampling-based method.

To reduce computation times, motion planning implementations for drones sample in a lower configuration space $\mathbb{R}^3$ rather than $\mathbb{R}^3 \times \mathbb{S}^3$ and employ straight-line planning.

However, this neglects the drone's dynamics. To overcome this limitation, the following section describes how RRT algorithm is combined with Model Predictive Control (MPC) to adapt to the robot dynamics and account for the differential constraints on motion.

Algorithm parameters	
Variable	**Description**
Startposition	Starting point of graph search
Endposition	Goal position to reach
Threshold	Radius in which endposition is considered reached
Stepsize	Distance in which new nodes are generated
End condition	Condition to terminate the algorithm, can be path length or number of iterations
Additional parameters for biased algorithm	
β (bias)	Fraction of points sampled around obstacles
R	Radius around obstacles in which points are sampled

TABLE I. Overview of variables in the RRT algorithm.

IV. Motion Planning: Time Parametrization

Firstly, as the MPC controller needs a time-dependent reference trajectory, it is necessary to assign each point $\underline{p}_i$ $i = 1, ..., N$ of the path generated by RRT (or any other planning algorithm) to a point in time t_i as well as a velocity $\underline{v}_i$. This is done by connecting all $\underline{p}_i$ to $\underline{p}_{i+1}$ using bang-bang velocity profiles: Starting from $\underline{p}_i$, the drone accelerates with a_{max} until reaching v_{max} and continues until it has to break with $-a_{max}$ to reach v_{min} at $\underline{p}_{i+1}$. The orientation and angular speed of the reference are fixed upright and zero for all points, respectively. Finally, the resulting trajectory is sampled with a time equivalent to the discretization time step t_s of the linearization.

We found that bang-bang velocity profiles – compared to constant velocity profiles – ensure more minor path deviations at corners while still enabling fast traversing straight path sections.

V. Motion Planning: MPC Controller

To follow the resulting trajectory after time parametrization, a linear MPC controller for trajectory tracking is used with the linear model (6). The optimization nature of this approach shortens the final path further, incorporating the quad-rotor dynamics and offering the opportunity to incorporate local obstacle avoidance in future work. The MPC problem is given by:

$$\min \sum_{k=0}^{N} \left((\underline{x}_k - \underline{x}_{ref,k})^T \mathbf{Q} (\underline{x}_k - \underline{x}_{ref,k}) + \underline{u}_k^T \mathbf{R} \underline{u}_k \right) \quad (7a)$$

$$\text{s.t. } \underline{x}_{k+1} = \mathbf{A}\underline{x}_k + \mathbf{B}\underline{u}_k \quad k = 0, \dots, N-1 \quad (7b)$$

$$\underline{x}_0 = \underline{x}_{init} \quad (7c)$$

$$\mathbf{K}^{-1}\underline{u}_k \geq -\mathbf{K}^{-1}\underline{u}_{op} \quad k = 0, \dots, N \quad (7d)$$

$$\begin{bmatrix} -\pi \\ -0.5\pi \\ -\pi \end{bmatrix} \leq \begin{bmatrix} \phi_k \\ \theta_k \\ \psi_k \end{bmatrix} \leq \begin{bmatrix} \pi \\ 0.5\pi \\ \pi \end{bmatrix} \quad k = 0, \dots, N \quad (7e)$$

As the rotor speeds command the quadrotor, the small-signal input vector $\underline{u}$ must be converted to rotor speeds with (1) resulting in $\underline{\Omega} = \sqrt{\mathbf{K}^{-1}(\underline{u}_{op} + \underline{u})}$ (with element-wise square root), explaining the constraint (7d). The angular bounds of constraint (7e) are given by [8].

This MPC problem is solved with CVXPY [3], [1] using the ECOS solver [4]. Figure 3 shows position as well as translational velocity of an exemplary reference trajectory and the path taken by the MPC controlled drone, showing the smoothing of the path.

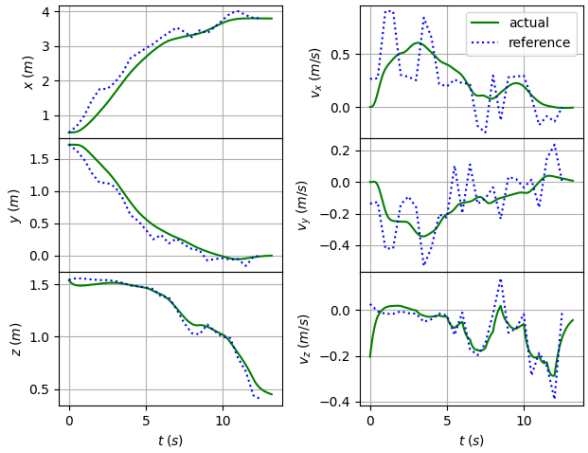

Fig. 3. Reference trajectory and actually taken trajectory using the MPC controller, corresponding to room 3 from Figure 9 (only translational values are shown).

VI. Results

To evaluate the various RRT algorithms, three simple obstacle environments were setup, as shown in Figure 4. Spherical obstacles were used for the numerical analysis to reduce computation times. Firstly, the biased algorithm was

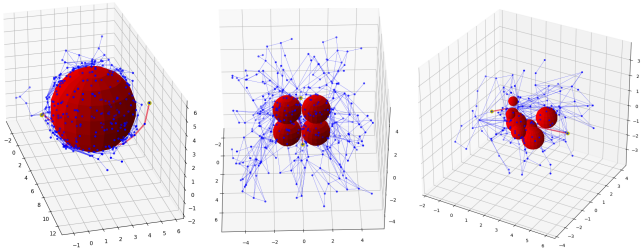

Fig. 4. Environment variations that were tested. Scenario a (left) consisted of a simple sphere placed in the direct path between start and end. Scenario b (middle) had a small hole in a wall of obstacles. Scenario c (right) consisted of a number of randomly generated obstacles.

evaluated in scenario a at several values for bias β, for a given path length. This is shown in Figure 5. A value of β of about 0.2 seems to be optimal.

Subsequently, the various algorithms and their biased variations were tested with regards to the computation time required for a certain path length. Results for scenario a are shown in Figure 6. Both biased versions outperform

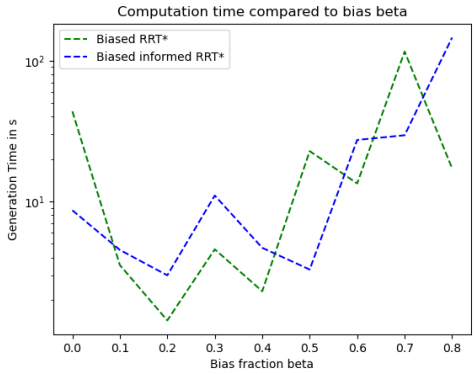

Fig. 5. The average computation time over 20 runs for various bias values β, for a path length to direct path length ratio of 1.36. RRT was not evaluated as it is not optimal and hence has very large computation times.

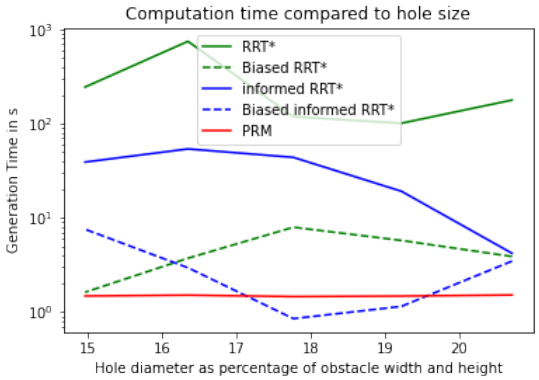

Fig. 7. The average computation time to find a path trough the hole over 5 samples for various RRT variations and PRM, compared to ratio of hole size to obstacle size in scenario b. RRT was not evaluated as it is not optimal and hence has very large computation times.

the unbiased ones, though the difference becomes less pronounced for less optimal path lengths. In general, Informed RRT* outperforms RRT*, but with the biased versions, this difference is less obvious, possibly because a path is generated so quickly that Informed RRT* does not get the chance to show its benefits. Finally, PRM is about as fast as the biased variant of RRT*.

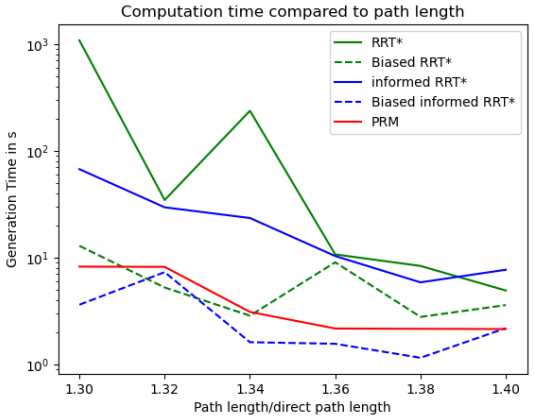

Fig. 6. The average computation time over 10 samples for various RRT variations and PRM compared to the ratio of path length to direct path length, in scenario a. RRT was not evaluated as it is not optimal and hence has very large computation times.

Additionally, the algorithms were tested in a scenario b were there is a hole in a 'wall' of obstacles. Results are shown in Figure 7. Computation times seemed to decrease exponentially with hole size for RRT* and informed RRT*. Informed RRT* was generally faster than RRT, which confirms the results as found by [7]. In contrast, the average time of the biased algorithms did not seem to have the same relationship to hole size, with the curves being more flat. Finally, PRM had a constant computation time, as it constructs a full road map. Subsequently, various numbers of obstacles were tested. Obstacles were generated randomly, and the time to find a path was recorded. To account for random variations in the generation and obtain reliable, statistically relevant results, 100 trials were conducted for each number of obstacles. The results can be seen in Figure 8. For RRT*, the time to generate a path increases with the number of obstacles, as would be expected. But again, for biased RRT*, this relationship does not hold.

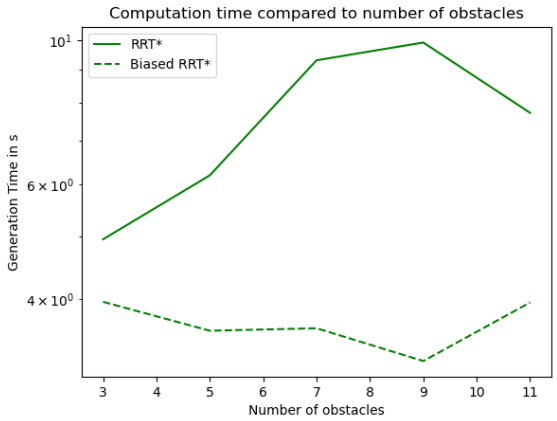

Fig. 8. The average computation time to find a path for various numbers of obstacles in scenario c. The obstacles were generated randomly between start and goal nodes, and the time until a path (irrespective of length) was found. Hence, only RRT* and its biased version were tested as informed RRT* is the same as RRT* until a path is found.

Hence, for scenarios where there are (many) obstacles between start and goal nodes, biased RRT* could provide real benefits, both in terms of time to generate shorter paths as well as time to generate a path, irrespective of length. Additionally, the results seem to confirm the analysis done by [7] regarding informed RRT*. In the 3D tests, biased RRT* was also generally faster than RRT*, especially in scenario b. This speed increase seemed to carry over to the biased variations as well, creating a combined algorithm that can very quickly find paths through gaps between obstacles. Especially for crowded indoor dynamic 3D environments, this could be beneficial to quickly recompute a path based on an updated obstacle map.

However, this version of RRT* only considers straight paths, and as such, does not take the dynamics of quad-rotors into account. Hence, as discussed in the motion planning section, after generating an offline path with 'biased informed RRT*', MPC is employed for online local motion planning.

So far, the analysed scenario has been kept simple so that the different algorithm options can be analysed fairly. Now that a selection has been made, the proposed method has been tested in more realistic scenarios by including rectangular-shaped obstacles. The three different rooms were defined with three increasing levels of difficulty regarding the number of total obstacles, their height and placement. The three different rooms have 7, 11 and 16 obstacles, including the top and bottom plates that mimic the floor and ceiling. The walls have been neglected for better visualisation. However, the graph search has been defined to account for the constraints of the walls of a room.

In room 1, the rectangular obstacles had the same height. In room 2, they were allowed to vary in height but had the same reference for the base, and lastly, in room 3, both the height and the base placement were varied, thus creating a wider range of possibilities of displacement for the drone. The results are presented in Figure 9. In the same graphs, the proposed path for navigating from start to goal determined from the graph search is presented. The graph search has a similar structure as was already presented in the simplified scenario with one obstacle in Figure 4. The number of sampled points, especially following the sampling algorithm with bias in the vicinity of obstacles, was not presented in this section to avoid cluttering the presented rooms.

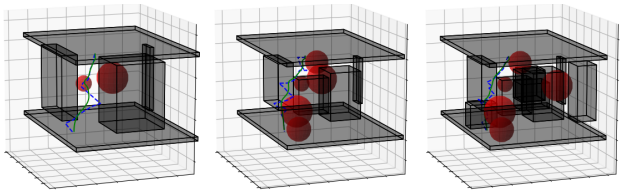

Fig. 9. Representation of the three different rooms with increasing levels of difficulty from left to right and the proposed navigation path resulting from the biased RRT* graph search shown in dashed blue. The followed path by the MPC controller is shown in green, being 83.4%, 72.5% and 78.4% shorter than the planned path from left to right, respectively.

Lastly, the reference trajectory created has been passed on further to the MPC and simulated in the pybullet simulator, with the results shown in Figure 9 as well. The path is followed closely, and the controller smoothes the sharper edges. Nonetheless, the maximum deviation from the reference trajectory is less than 10 [cm] and is considered an acceptable margin for this type of application. Additionally, the smoothing of the path optimizes its length to between 72.5% and 83.4% of the original trajectory. 2D plots of the followed path in simulation versus the reference trajectory for room 3 are an example in 3. Thus this method demonstrates the ability of the proposed obstacle-driven RRT algorithm to navigate through complex and realistic environments while taking into account the dynamics and constraints of the quad-rotor. These results support the potential of this algorithm for real-world applications, such as indoor navigation for drones and will be discussed in more detail in the following section.

VII. DISCUSSION AND FUTURE WORK

This paper presents the development of a version of RRT* optimized for indoor environments, called biased RRT*. The numerical analysis results showed promising benefits, especially for quick path generation in case of a large number of obstacles or small holes. In practice, this could be useful for crowded, dynamic indoor environments, allowing for quick re-computation of the global path in case of a change in environment, newly detected obstacles, or even continuous computation of the global path. Additionally, it is shown that RRT could be combined with a local path planner like MPC to create a full offline-online integrated motion planner for quad-rotors in 3D environments.

However, this approach has limitations that suggest potential directions for further research. Firstly, the numerical analysis was performed with a limited number of trials. Given the inherent randomness of RRT, a Monte-Carlo analysis can be performed to account for the random variations that occur during data generation and produce statistically relevant results.

Secondly, the developed biased RRT program does not consider robot dynamics. It relies on a simple straight-line metric which creates nodes based on minimum distance to the goal, not necessarily a time-optimal path. Combining the algorithm with a local planner such as MPC can address this issue, but it may still result in sub-optimal reference trajectories. Future research could focus on incorporating kinodynamic planning and working with more complex obstacle shapes, for example, by discretizing edges rather than checking collisions algebraically. Another possibility would be to evaluate how MPC or other local planners could be optimally combined with RRT to lift the constraints on straight paths and obstacles.

Thirdly, the code was created from scratch so it may be sub-optimal. The effort could be put into optimizing it because quick and efficient computation is essential for path generation in dynamic environments.

Furthermore, the simulation in this study only considered a scenario with a pre-generated global path combined with MPC due to time constraints. It would be valuable to investigate the real-time application of the biased (informed) RRT* algorithm by testing its ability to dynamically re-compute global paths and update them based on new sensor data in an online manner. This would provide insights into the feasibility and effectiveness of the algorithm in real-world planning for quad-rotors in complex dynamic environments.

Provided that the control method was not the main focus of this study, obstacle constraints were not included in the MPC formulation. Future work could incorporate local obstacle avoidance in the MPC to increase the reliability of the motion planning stack.

Finally, evaluating the proposed RRT-MPC combination in practice would also be a valuable addition as it would allow a practical validation of the experimental results.

REFERENCES

[1] Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[2] Ross E. Allen and Marco Pavone. A real-time framework for kinodynamic planning in dynamic environments with application to quadrotor obstacle avoidance. *Robotics and Autonomous Systems*, 115:174–193, 2019.

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

[4] Alexander Domahidi, Eric Chu, and Stephen Boyd. Ecos: An socp solver for embedded systems. In *2013 European Control Conference (ECC)*, pages 3071–3076, 2013.

[5] HTMN ElKholy. Dynamic modeling and control of a quadrotor using linear and nonlinear approaches. *American University in Cairo*, 2014.

[6] Fanjin Zeng. 2D implementation of RRT, version 1.0, from mit license copyright (c), 2019. https://gist.github.com/Fnjn/58e5eaa27a3dc004c3526ea82a92de80.

[7] Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. Informed RRT: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, sep 2014.

[8] M Islam, M Okasha, and MM Idres. Dynamics and control of quadcopter using linear model predictive control approach. In *IOP Conference Series: Materials Science and Engineering*, volume 270, page 012007. IOP Publishing, 2017.

[9] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning, 2011.

[10] Scott Drew Pendleton, Wei Liu, Hans Andersen, You Hong Eng, Emilio Frazzoli, Daniela Rus, and Marcelo H. Ang. Numerical approach to reachability-guided sampling-based motion planning under differential constraints. *IEEE Robotics and Automation Letters*, 2(3):1232–1239, 2017.

[11] Charles Richter, Adam Bry, and Nicholas Roy. *Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments*, pages 649–666. 04 2016.

[12] Dustin J. Webb and Jur van den Berg. Kinodynamic rrt*: Optimal motion planning for systems with linear differential constraints, 2012.

[13] Martin Zimmermann, Minh Nhat Vu, Florian Beck, Anh Nguyen, and Andreas Kugi. Two-step online trajectory planning of a quadcopter in indoor environments with obstacles, 2022.