Engineering Science and Technology, an International Journal 48 (2023) 101560

Contents lists available at ScienceDirect

Engineering Science and Technology, an International Journal

journal homepage: www.elsevier.com/locate/jestch

Full length article

A 3D navigation algorithm switching between waypoint and Bezier curves based local plans for micro air vehicles

Furkan Cakmak *, Sirma Yavuz

Yildiz Technical University, Department of Computer Engineering, Esenler, Istanbul 34220, Turkey

ARTICLE INFO

Keywords:
MAV navigation
Navigation with Bezier curves
Local planner switching

ABSTRACT

Micro Air Vehicles (MAVs) are playing an increasingly prominent role in our lives. Numerous studies have been conducted on autonomous mobility. In this study, an architecture has been developed for 3D mapping and 3D navigation for MAVs. The flight time of a MAV is considerably shorter than ground vehicles. Therefore, they are expected to perform tasks more efficiently and with higher performance. MAVs can plan a route to a given target and follow this path using the developed 3D navigation algorithm. Tracking paths produced by classical waypoint-based planners is slow. For this reason, a local path planner based on Bezier curves has been developed for MAVs, with path planning represented as a curve on the low-level control card. Since it is not always possible to reach the given target with a single curve, the multi-hop Bezier curves approach has been proposed. Particularly in environments with low complexity, this approach yields very effective results. In cases where the Bezier path cannot be generated in high-complexity environments, a metric has been developed to measure the complexity, allowing for a switch between waypoint and multi-hop Bezier curves based local plans on the environment complexity metric. Thanks to these developed methods, the aim is to extend the flight time of MAVs by optimizing their battery usage. Within the scope of this study, all experiments were conducted in the ROS-GAZEBO simulation environment. Firstly, waypoint-based local planning methods such as Dijkstra and A*, which will function within the navigation algorithm, were implemented to operate in 3D, and performance tests were conducted in the designed simulation environments. Subsequently, a multi-hop Bezier curves-based local planner was developed, and performance comparisons were made with waypoint-based methods using the same simulation environments, taking into account environmental complexity parameters.

1. Introduction

Within the scope of this study, various techniques were developed and performance analyses were carried out to shorten the time to reach the target and extend the battery life of the micro air vehicle, which navigates autonomously to a given target. In recent times, the utilization of Micro Air Vehicles (MAVs) has significantly increased. Autonomy software developed for MAVs has gained substantial recognition in the literature. A closer examination of the research in this field reveals that the primary challenge lies in the battery life of MAVs, which serves both flight and processing functions. This issue has raised several critical questions for researchers: Can more efficient flight planning be achieved? How can flight duration be reduced? Is it possible to optimize computing power usage? In the context of this study, diverse techniques were developed, and performance analyses were conducted to enhance the MAV's ability to autonomously navigate to a given target, reducing time to reach the target and extending battery life.

The objective of this project is to develop 3D path planning, navigation, localization, and mapping algorithms for MAVs to enable autonomous navigation. Local planners based on waypoint and Bezier curves have been created to facilitate the successful navigation of MAVs within various environments. In this proposed method, we introduce a navigation approach that can seamlessly switch between local planners by assessing the complexity of the environment.

1.1. Literature review

The most significant challenge faced by MAVs is the limited computing and storage capacity. Hence, it is imperative that the methods designed for MAVs have low processing complexity. In the literature, various methods are employed for path planning operations in MAVs. Effective waypoint selection for each path node is a prominent area of research in path planning [1]. Another method involves assessing the effectiveness of the planned path using a Bayesian approach [2].

* Corresponding author.
E-mail addresses: fcakmak@yildiz.edu.tr (F. Cakmak), smyavuz@yildiz.edu.tr (S. Yavuz).

https://doi.org/10.1016/j.jestch.2023.101560
Received 3 July 2023; Received in revised form 13 October 2023; Accepted 22 October 2023
Available online 6 November 2023

Additionally, many path planning algorithms are based on the A* algorithm [3]. The primary objective of all these methods is to plan a path to a destination while avoiding obstacles. It is worth noting that these research endeavors are undertaken assuming knowledge of the environment map, whether it is feature-based, grid-based, or graph-based.

Autonomous navigation presents another challenge for MAVs. The navigation process necessitates acquiring information about the environment, which includes addressing the following issues: defining the exact goal for the MAV, generating the desired path, and reaching the goal by implementing various speed commands along the created path. This process requires essential information, such as the environmental map, the pose of the MAV, and the target's position on the map, commonly referred to as robot localization.

In the literature, several studies on this subject have shown that navigation is often addressed in conjunction with localization and mapping. This is because information about the environment, the MAV, and goal positions is essential for navigation. The concurrent process of mapping and navigation is commonly referred to as active SLAM (Simultaneous Localization and Mapping). In this article [4], we have compiled various active SLAM studies that explore different active SLAM types used on various robot platforms. Upon evaluating the active SLAM techniques discussed in this survey, several techniques that have been considered state-of-the-art for some time appear worthy of examination.

In a study conducted [5], performing autonomous robot navigation in a multi-floor environment was carried out on 2.5D map. The writer describes 2.5D map as follows. 2-dimensional map is created on a space that the MAV is located. If the height, which the robot is located, reaches a certain threshold in the same space, a layer is further added to the 2D map. Thus, the multi-floor environment mapping was carried out in 2.5D space. In another study [6] carried out for robot navigation speed has come to the fore. The realization of the fast navigation was solved by making plan in a very short time. Writers are mentioned that there are basically three important features which separates this algorithm from the other methods. These can be listed as follows: immediate implementation of the seen obstacle's representation on the map by using raw sensor data, automatically determining the threshold to the destination according to the vehicle's dynamics and structure of the target environment, carried out the test in real environments. In another method [7], navigation can be performed after calculating the location estimation from extracted visual odometry. The sensitivity of the information odometry is important for navigation control mechanisms.

In a study conducted [5], autonomous robot navigation in a multi-floor environment was performed using a 2.5D map. The author defines a 2.5D map as follows: a 2-dimensional map is created in the space where the MAV is located. If the height at which the robot is situated exceeds a certain threshold in the same space, an additional layer is added to the 2D map. This approach allowed for mapping of multi-floor environments in a 2.5D space. In another study [6] focusing on robot navigation speed, achieving fast navigation became a primary concern. The authors mentioned that their algorithm stands out due to three key features: immediate incorporation of observed obstacles into the map using raw sensor data, automatic determination of the threshold to the destination based on the vehicle's dynamics and the target environment's structure, and conducting tests in real-world environments. Additionally, in another method [7], navigation is performed after estimating the location from extracted visual odometry data. The accuracy of odometry information is crucial for navigation control mechanisms

3D mapping is essential for robots in need of navigation. Therefore, in this study, mapping is planned to be executed as described in [8]. The implementation of the SLAM algorithm using a single camera and an IMU sensor is presented in [7]. In these studies, feature extraction is performed from RGB images, and matches between features in successive frames are calculated. In another study, the mapping process is conducted using stereo cameras [9]. The MAV is equipped with a laser range measurement sensor that scans a 270-degree angle up to 30 m. High-frequency 2D odometry information is derived from laser data, and it is combined with IMU data to obtain a quaternion pose of the MAV. This process also involves mapping features extracted by the stereo camera, resulting in the creation of a 3D map. In a comparative study [10], evaluating the performance of the techniques presented in the mentioned articles, it is observed that point cloud-based mapping techniques perform better for robot navigation.

Conventional voxel-based occupancy grid frameworks often suffer from reduced planning efficiency when dealing with large numbers of grid cells [11]. Additionally, numerous methods have been developed for obstacle avoidance within navigation algorithms [12–15]. In contrast to ground vehicles (AGVs), MAVs feature a higher-dimensional design space, rendering the path planning for MAVs a challenging task. Nevertheless, some researchers have also conducted investigations related to position estimation for testing navigation systems in indoor environments [16–18]

OctoMap is a 3D mapping algorithm widely utilized in various fields, particularly in robotics research [19–21]. This method processes data from depth sensors and stores it in an octree (octal tree) data structure [21].

The flight dynamics of MAVs should be taken into consideration when conducting path planning, as they are capable of reaching high speeds, unlike ground robots. Traditional 3D pathfinding methods primarily aim to find the shortest path in the least amount of time [22]. Some of these methods employ heuristic approaches to find the shortest path, while others prioritize faster execution [23]. In certain studies [24], experiments have been conducted to develop safe and smooth trajectory planning for navigation. Enhancing navigation methods for local planners is a technique frequently employed by researchers. In another study [25], local planners were designed for social robots that need to navigate in environments with frequent human presence.

In addition to these methods, various approaches have been explored to understand the flight dynamics of MAVs [26]. However, most of these approaches predominantly focus on speed-oriented maneuvers of MAVs in highly structured competition environments, without addressing the need for 3D mapping and indoor navigation. In the model presented in this study for indoor environments, the emphasis is on calculating path planning based on the driving dynamics of MAVs. To achieve this, Bezier Curve [27,28] is employed, as its curves closely resemble the driving dynamics of MAVs. Bezier curves are a parametric curve format frequently used, particularly in computer graphics and related fields. Recently, researchers have also conducted studies on path planning approaches with Bezier curves to address gaps in robot navigation [29]. Another study focuses on the development of local planners using Bezier curves for robot platforms [30]. Furthermore, B-spline smoothing techniques based on Bezier curves are employed to refine the created navigation path [31].

Upon examining all of these studies, it becomes evident that numerous investigations have explored robot navigation using Bezier curves. Nonetheless, a notable gap in the literature exists regarding the application of multi-hop Bezier curves and the seamless transition between local planners based on environmental complexity. In this study, new methods have been developed to address and bridge this gap in the existing literature.

In this study, curves are generated based on the selected control points. Typically, while the first and last points lie on the curve, the other selected points are not necessarily located directly on the curve. Leveraging these properties of the Bezier curve, path planning was implemented for the path tracking module integrated into the control systems of MAVs. This method, functioning as a global planner within the navigation algorithm, effectively avoids enlarging the search space. Although there may be some marginal performance trade-offs in terms of path calculation time, significant time savings in MAV path execution have been achieved.

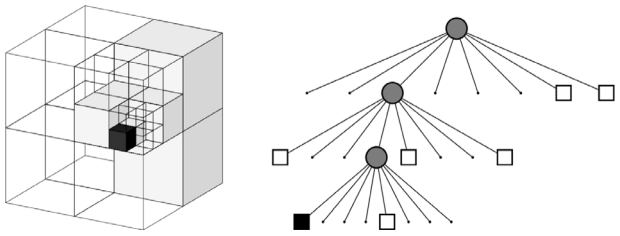

Fig. 1. It is the way in which full (black) and empty (white) cells are retained in leaves in Octree. The volumetric model is on the left, the tree model is on the right [21].

2. Methods

In this study, OctoMap was employed as a 3D mapping technique for MAVs. Additionally, the study involved the development of costmap extraction and global and local route planning methods integrated into the navigation algorithm. All experiments and research were conducted within the ROS/Gazebo (The Robot Operating System) framework.

2.1. OctoMap - 3D mapping algorithm

In the OctoMap method, a three-dimensional map is stored in an octree structure, with a maximum of eight leaves per node. In 3D maps, each cubic volume, referred to as a voxel, serves as a node within the octree. This volume is recursively subdivided into 8 sub-cubes until it reaches the minimum voxel size. The OctoMap tree structure, as depicted in [21], is illustrated in Fig. 1.

In mapping methods, at the most basic level, cells on the map are designated as either occupied or empty by assessing their likelihood of being full. The occupied or empty state of each map node is updated using newly acquired sensor data. In occupancy grid mapping methods, this updating process is typically governed by Eq. (1).

$$L(n \mid z_{1:t}) = L(n \mid z_{1:t-1}) + L(n \mid z_t) \tag{1}$$

In the OctoMap method, which is rooted in this approach, the occupancy of the point cloud coordinates derived from the depth sensor is determined using the probability formula as provided in Eq. (2).

$$L(n \mid z_{1:t}) = max(min(L(n \mid z_{1:t-1}) + L(n \mid z_t), l_{max}), l_{min}) \tag{2}$$

A is the probability that an n leaf node in the Octree is full, based on $z_{1:t}$ sensor measurements from the first moment so far. The most recent sensor measurement $P\left(n|z_{1:t}\right)$ is expressed as a priori probability $P(n)$, the probability of occupancy of the n node in the previous time $P(n|z_{1:t-1})$, the occupancy probability of the n node based on the last sensor measurement taken by $P(n|z_t)$. The logarithm of the odds ratio, which is the ratio of a probability to its complement, is called the log-odds ratio or logit, and is given by Eq. (3).

$$L(n) = log\left[\frac{P(n)}{1 - P(n)}\right] \tag{3}$$

Eq. (4) is obtained when the log-odds ratio of Eq. (3) is calculated.

$$P\left(n \mid z_{1:t}\right) = \left[1 + \frac{1-P(n \mid z_t)}{P(n \mid z_t)} \frac{1-P(n \mid z_{1:t-1})}{P(n \mid z_{1:t-1})} \frac{P(n)}{1-P(n)}\right]^{-1} \tag{4}$$

Eq. (1) necessitates k measurements for a given n cell to transition from being marked as full to being marked as empty after k measurements. While this approach is suitable for static environments, it may prove inadequate for dynamic applications. Therefore, in the OctoMap method, Eq. (2) is employed to expedite the reflection of environmental changes on the map.

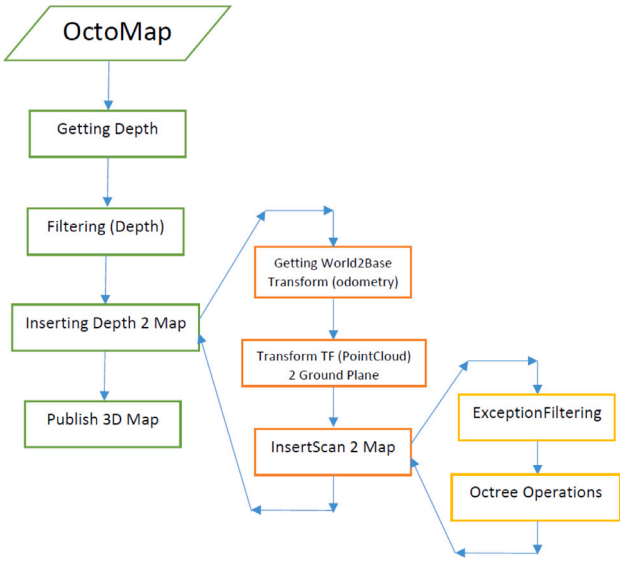

Fig. 2. The flow chart of OctoMap algorithm.

The values l_{max} and l_{min} represent the minimum and maximum ranges for the log-odds ratio. This range constraint helps ensure that the log-odds ratio of a given n node does not increase or decrease excessively due to sensor measurements, thereby preventing the need for an excessive number of sensor measurements to update it.

When a cell is identified as full, the cells located between the sensor and these coordinates are designated as empty using raycasting. Furthermore, if all the leaves within a recursively divided node share the same state (whether they are all occupied or all empty), the leaves are pruned [21].

Thanks to this tree structure and enhancements in its implementation, OctoMap enables significant compression of 3D maps. The study related to this technology expresses the memory occupied by maps obtained at various resolutions and sizes.

In applications like real-time 3D navigation, where the rapid creation of maps is essential, the speed and processing power consumed by the employed methods play a crucial role. OctoMap stands out for its efficient memory usage, even in significantly large areas, making it a preferred choice for real-time mapping applications.

In OctoMap, distinguishing between free and unmapped areas on a map is of great significance, especially when the study's focus is on navigation and exploration. Only a few 3D mapping algorithms account for this characteristic. However, it is important to note that OctoMap is primarily a mapping algorithm and not a SLAM algorithm. Therefore, an external localization algorithm is required to provide odometry information to the OctoMap algorithm. The basic flowchart of OctoMap is outlined in Fig. 2.

2.2. Navigation algorithm

Another crucial aspect of this study involves the development of a navigation module for MAVs. Navigation involves guiding the robot towards a specified goal using motion tools like motors, servos, and more, making it a vital module for autonomous vehicles. To operate effectively, the navigation module requires input data, including the environment map, sensor data, and trajectory information for each robot. A visual representation of the designed navigation module can be found in Fig. 3. The core components of the navigation algorithm are depicted in this graphic.

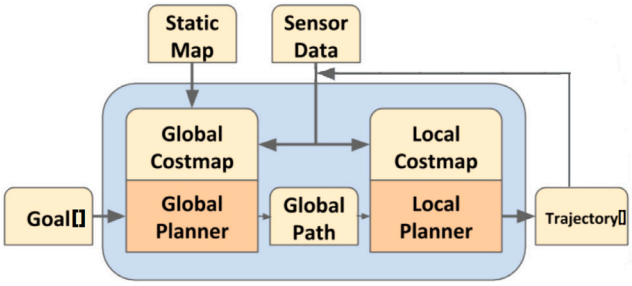

Fig. 3. The architecture of navigation module.

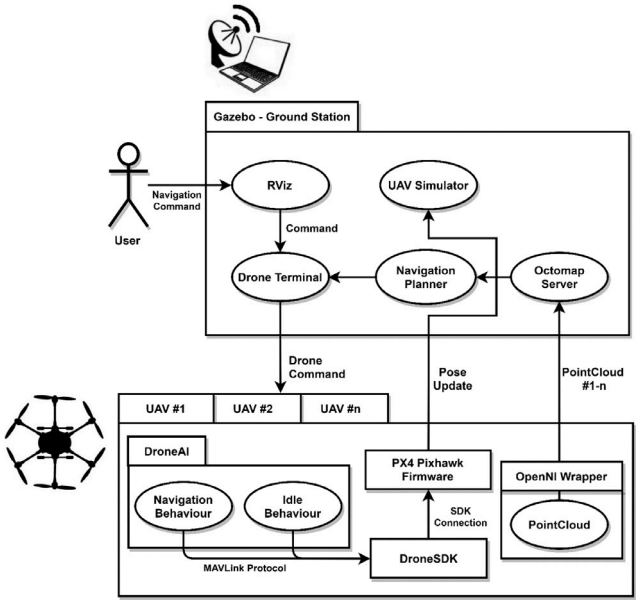

Fig. 4. MAV navigation system design.

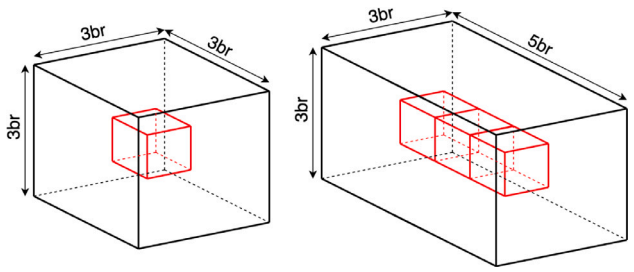

Fig. 5. Inflating a cell with a side of 1 unit on the left by 1 unit in three dimensions. Inflating 3 adjacent cells with a side of 1 unit on the right by 1 unit in three dimensions. Cells are visualized in red, inflated in black.

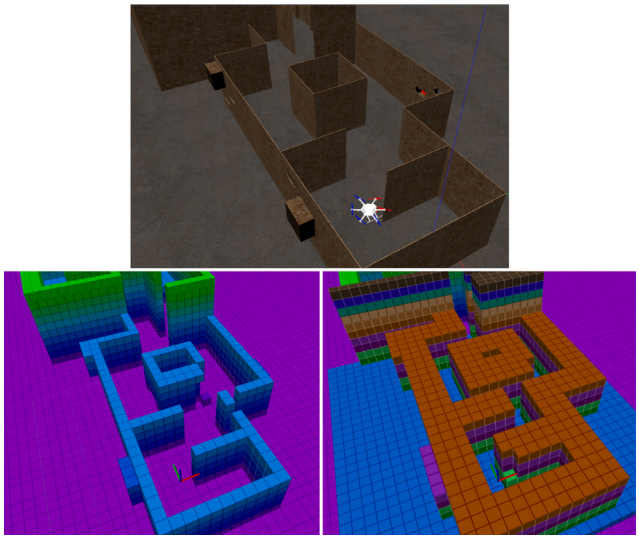

Fig. 6. The experimental environment used in the study above, the OctoMap map of the environment on the left, and the cost map, which is the 1 cube inflated version of the map, are given on the right. One side of the cube cells in the map and cost map is 25 cm in size. Cells of the same height are visualized with the same color.

The designed navigation algorithm extract a global costmap using OctoMap data. Only one instance of global map is calculated because there is only one map on the entire system. Local costmaps are calculated using MAV depth sensor information. MAV has its own local costmap information to be able to move dynamic environments.

Using local and global costmaps, the system produced a global path to given goal using A* algorithm. Each 3D matrix cell has 27 different directions and the point of A* route array is consist of these directions. In a nutshell, navigation module calculate two different costmap: Local costmap using sensor data, Global costmap using OctoMap data.

Using two different costmaps for route calculation is not efficient. Therefore, the navigation algorithm is designed to merge all costmap information and publish a unified costmap. This approach reduces the access time to each 3D cell. With this developed method, systems have been established to facilitate the movement of multiple MAVs within the environment. The operational diagram of the developed system can be seen in Fig. 4.

2.2.1. Navigation path calculation and path tracking

One of the most basic tasks in mobile robot studies is to calculate the shortest path from a starting position to a target position through unobstructed areas [32]. A 3D version of the A* algorithm was used to plan the route from the MAVs to the target points on the obtained map. The cost from the starting point to the n node is expressed in $g(n)$, the estimated cost to the destination point in $h(n)$, and the total cost in $f(n)$. Situations that require changing the direction between two adjacent cells are added to the $g(n)$ cost as an extra cost. In this way, it is aimed to calculate the paths that MAVs can follow by changing their direction less. $h(n)$ is calculated based on the Euclidean distance to the target. The basic A* equation is expressed by Eq. (5).

$$f(n) = g(n) + h(n) \tag{5}$$

Costmaps were utilized for route planning on the generated 3D map. These maps enable the robot to be represented as a point on the map, facilitating path calculations from the robot's current location to the target point.

The costmap of the cells is obtained by parametrically inflating the map cells marked as full. Fig. 5 visualizes how the map cells are inflated.

The cost of these inflated cells, which should not enter the point representing the robot, is determined as a higher value than the cost of empty cells. Thus, in this study, the cost function of the A* algorithm is; It is used to calculate a path that the robot can follow without hitting obstacles from empty cells whose cost is considered to be zero. The image of the costmap created for the simulation area used in the study is given in Fig. 6.

2.3. Local plan extraction based on Bezier curves

Methods such as Dijkstra and A*, which are frequently used to produce global plans, can bring high costs in terms of calculation [33]. These methods were used as a global planner in this study to compare proposed method.

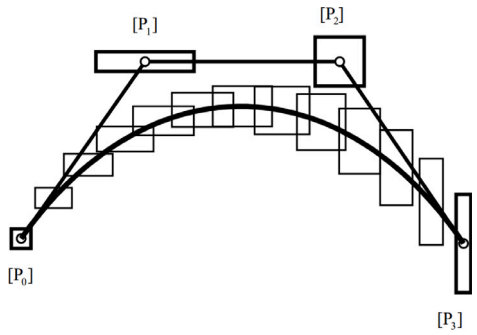

Fig. 7. The 4-point Bezier curve example [34].

Researchers are generally involved in studies that will increase the performance values of these methods. But there are basically 2 low-level MAV controls for MAVs to follow local plans: Waypoint-oriented and trajectory-oriented approaches. A*, Dijkstra and etc. are used to generate a navigation path for waypoint-oriented low-level MAV controls.

While global path calculation times are typically fast using such methods, it is worth noting that MAVs can face challenges when following waypoint-oriented paths, as they can be slow. To address this, in this study, Bezier curves are employed for trajectory-oriented low-level MAV control, allowing for high-speed tracking. It is important to recognize that while slow navigation is necessary for mapping, most of the MAV's navigation time is spent within areas they have already mapped. Thus, a global plan structured in the form of a curve can generate motion commands that better align with the MAV's driving dynamics.

Let P_0 and P_1 derive distinct points, the Bezier curve function in quadratic form can be given in Eq. (6).

$$B(t) = P_0 + t(P_1 - P_0) = (1-t)P_0 + tP_1, \qquad 0 \le t \le 1 \tag{6}$$

Since 3D path planning is needed for MAVs, the cubic form of the Bezier curve must be used. The explicit form of the cubic Bezier curve is given in Eq. (7).

$$B(t) = (1-t)^3 P_0 + 3(1-t)^2 tP1 + 3(1-t)t^2 P_2 + t^3 P_3, \qquad 0 \le t \le 1 \tag{7}$$

P_0 represents the position of MAV and P_3 represents navigation goal point. Points P_2 and P_3 are searched in free cells on the OctoMap using the 3D binary search method. After the optimum points are determined, those that do not intersect with the costmaps are scored according to their lengths. Points P_2 and P_3 that get the lowest score are determined. The 4-point Bezier curve visualization is given in Fig. 7.

As explained by the formulas above, Bezier curves align well with the trajectory-oriented movement capabilities of MAVs. In other words, in an obstacle-free environment, a MAV can efficiently follow a path generated by a Bezier curve with a trajectory-oriented movement ability, which directly matches MAVs' mobility. This leads to faster reaching of the destination. However, when MAVs navigate in environments containing obstacles, route planning is essential to avoid these obstacles. If a Bezier curve, as detailed below in its generation process, intersects with an obstacle in the environment, that curve is deemed unfeasible for MAVs to follow. In such cases, a new curve must be calculated in the space, and it needs to be verified whether this new curve avoids the obstacle. This process is performed iteratively until an appropriate Bezier curve is found for navigation. The determination of Bezier curve samples will be thoroughly discussed in the following sections.

2.4. Local plan extraction based on multi-hop Bezier curves

Drawing a single curve from the starting point to the designated target may not always be feasible. In situations where a continuous path cannot be established with a single curve, research has been conducted to explore the possibility of creating a multi-hop path to reach the target. The objective is to reach the target using multiple hops, necessitating the identification of intermediate hop points. The number of hop points has been predetermined through empirical methods. In other words, efforts have been made to predefine a navigation route to the destination using a specific number of hops, such as 2-hop or 3-hop. The determination of waypoints becomes a crucial consideration when aiming to reach the destination using 2, 3, 4, or any other number of hops.

Although various approaches were explored to determine the hop points, the method described below, which demonstrated the best performance, was developed and employed as the approach for finding hop points. In scenarios where a 2-hop path is planned for a given navigation destination, the following method is used:

- The shortest path from the starting point to the target point is determined as a Euclidean line.
- Circles with the minimum diameter are drawn from the midpoint of this Euclidean path (the Euclidean midpoint).
- Six points, closely spaced, are identified along the drawn circle where it does not intersect with filled map cells.
- For each of these points, two different Bezier curves are generated for the starting and target positions.
- If the destination can be reached using these 2 Bezier curves, this route is selected as the navigation path.
- However, if a valid navigation path cannot be established using these 2 Bezier curves without encountering obstacles, two new midpoints are identified by moving a certain distance from the Euclidean midpoint towards the target and starting points.
- The calculations described in the first case are then applied to these new midpoints to find a valid navigation path.
- These operations are repeated until a valid navigation path is found or a certain time limit is reached.

Since the calculation of multiple hop points lends itself to parallel processing compared to single hop calculations, a performance boost is achieved by using separate threads to calculate for each candidate point.

The 3-hop approach is executed in a similar manner to the 2-hop approach. Calculations are performed by determining two starting points at equal intervals along the Euclidean line. The curves that reach local maxima in the search space are accepted, without attempting to find the global maximum curve. This approach is favored because there is negligible time difference between the best curve and any curve found suitable for MAV tracking. Additionally, the time required to find the best global curve can be considerably long.

2.5. Local plan switching through environment complexity

As previously discussed in earlier chapters, the multi-hop approach developed using Bezier curves may not yield effective results, particularly in tight and intricate environments. The extensive search space often leads to incomplete searches within a reasonable timeframe, potentially diminishing the effectiveness compared to creating a local plan with waypoints

Considering this situation, it becomes evident that employing a method capable of transitioning between local plans based on the complexity of the environment is more effective. Put simply, in environments characterized by wide openings, considerable distances between objects, and situations where a Bezier curve can be drawn with a single hop whenever feasible, local planning over Bezier curves with

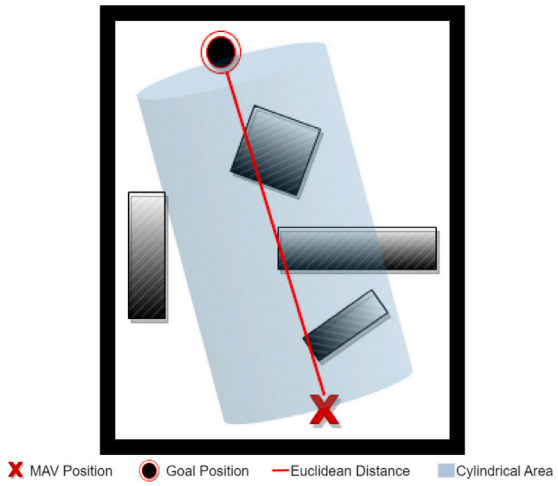

Fig. 8. Cylindrical search space.

multiple hops is expected to be more effective. Conversely, waypoint-based navigation is likely to yield better results in environments with closely spaced objects, limited open spaces, and narrow corridors.

To address this, it is crucial to determine the complexity of the environment using the available data. For this determination, the resources at hand include point cloud data obtained in real-time from MAVs, the 3D environment map generated by MAVs, and cost maps derived from these maps. When examining these sources individually, it is evident that point cloud data, specifically depth data, is confined to the MAVs' viewpoints, and the target to be reached is typically located in a position not directly visible to the MAV. Therefore, the use of point cloud data is often deemed ineffective.

Considering the 3D maps and cost maps among the data that can be utilized, they contain similar information, as costmaps are calculated based on the 3D maps generated. Thus, the primary question becomes how to calculate environment complexity from 3D maps. To tackle this challenge, two distinct approaches have been explored within the scope of this study

Firstly, as previously mentioned in this study, the generated environment map is based on OctoMap, and the OctoMap mapping method operates on the octree data structure. This data structure includes both empty and occupied voxels. If an empty voxel has other adjacent empty voxels, it is represented as a larger voxel to encompass those surrounding empty areas. A similar principle applies to occupied voxels. Essentially, the volumetric size of the voxels serves as a determining factor for the occupancy status of the coordinate where the voxel is located. This concept is illustrated in Fig. 1.

To calculate the environment complexity, a 3D Euclidean distance is established between the MAV's position and the given target location for navigation. This Euclidean distance is then processed through a voxel detection filter, which expands the line into a cylindrical volume with a specified parameter. The expansion parameter, in this case, was determined empirically to be 5, 7, and 9 times the width of the MAV used. This approach allows for the examination of voxels within a cylindrical area, taking advantage of their organization within an octree structure.

If the count of large voxels within this area exceeds empirically determined threshold values, the decision is made to utilize a Bezier curves-based planner for path planning. Otherwise, a waypoint-based planner approach is adopted. Notably, the calculations involved in determining the Euclidean line and iterating through the octree data structure are exceptionally fast, rendering the time taken for these operations negligible. The representation of this cylindrical area is provided in Fig. 8.

In this second method developed, akin to the previous method, a cylindrical volume is formed through a 3D Euclidean line connecting the MAV's position with the designated target. Unlike the first method, where the size of voxel dimensions plays a critical role, this method does not consider the voxel sizes within the cylinder area. Instead, it focuses on calculating the empty and occupied grids within the cylinder area in square meters and proportioning them to each other.

This approach derives a ratio of empty and occupied cells within the cylindrical area. Subsequently, this ratio is subjected to a filtering process using empirically determined threshold values to assess the environment's complexity. If the environment is determined to be densely populated, our approach opts for the Bezier curves planner, whereas if it is less crowded, a waypoint-based planner is employed.

A virtual cylindrical area, depicted in Fig. 8, is delineated from the MAV's location to the target point, and the calculation of both occupied and empty cells within this region is conducted. Just as in the previous experiment, various tests were conducted with cylinder sizes ranging from 5 to 9 times the width of the MAV. This approach necessitates a 3D matrix-based search, in contrast to the iterative search on the Octree data structure, which exhibits a high time complexity and a noticeable computational overhead. This additional computation is incorporated into the overall path calculation time, ultimately determining the total time required to navigate to a specified target point.

3. Experimental results and discussion

This section provides a detailed examination of the experimental results concerning the OctoMap mapping technique and the developed navigation method.

3.1. The experimental results of multi-robot navigation module

MAV located at the simulation environment take off in an autonomous way and start to spin around itselves. PointCloud data is taken from the depth cameras simulated with the extension in the Gazebo. The PointCloud data obtained during the time the MAV is in the air and the OctoMap with occupancy grids as shown in Fig. 9 is extracted via the OctoMap Server node. In addition, global and local costmaps are generated to avoid obstacles.

In the simulation environment, the MAV autonomously takes off and initiates self-rotation. PointCloud data is collected from the simulated depth cameras using the Gazebo extension. This PointCloud data is acquired while the MAV is airborne, and the OctoMap with occupancy grids, as depicted in Fig. 9, is generated through the OctoMap Server node. Furthermore, global and local costmaps are created to facilitate obstacle avoidance.

The user visualizes the OctoMap, which the system generates, using RViz, a 3D visualization tool. The user, with the assistance of interactive markers, directs the vehicle by determining its navigation position from the RViz screen. To ensure a safe and precise journey to the predefined destination, the most efficient route on the map is selected via global and local costmaps. Throughout this process, the path is continually updated to safeguard the MAV from obstacles and identify the correct route.

Costmaps employed for navigation during operation are presented to the user through interactive markers, as illustrated in Fig. 10. The yellow area on the OctoMap represents the costmaps, designed as a sliding window, while the red line depicts the waypoint-based route to the designated destination.

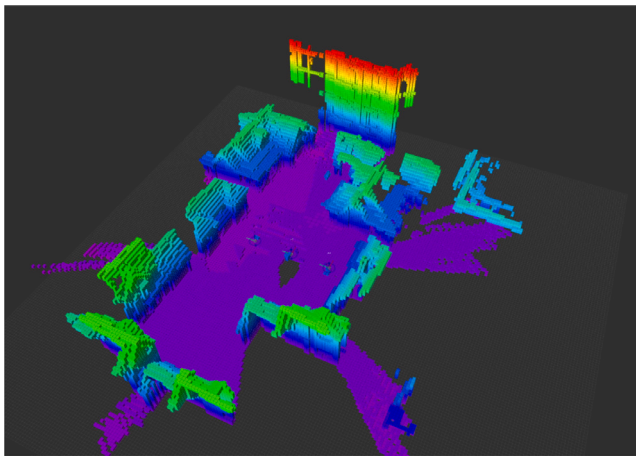

Fig. 9. A generated OctoMap.

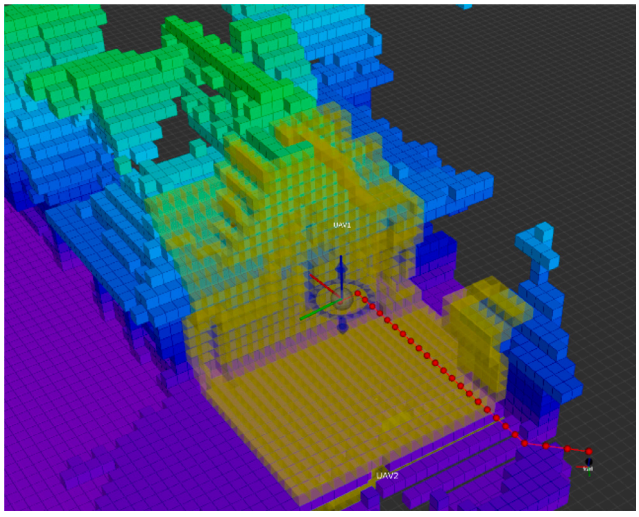

Fig. 10. MAV navigation interactive marker costmaps and paths.

3.2. The experimental environment and results for local plan extraction based on Bezier curves

Global plan calculations and executions were carried out using the Dijkstra, A* and Bezier curves method in 4 different scenarios prepared in the Gazebo environment. The 2D sketch of these experimental environments is given in Fig. 11. Since the paths produced by the Dijkstra and A* methods are very similar, they are visualized as the same.

The calculations and executions of global plans were conducted using both the Dijkstra, A* and Bezier curves methods within four distinct scenarios created in the Gazebo environment. An illustrative representation of these experimental environments in 2D can be found in Fig. 11. Given the high similarity of paths generated by the Dijkstra and A* methods, they are visualized as a single entity.

In Fig. 12, the Bezier curve for the 3rd scenario is given. It can be seen that the waypoints are organized to pass through a Bezier curve.

Fig. 12 illustrates the Bezier curve for the third scenario, showing the waypoints arranged along the path of the Bezier curve.

The results, as presented in Table 1, indicate that while the Dijkstra and A* methods exhibit better path calculation times, their path execution times are notably longer compared to the results generated by the Bezier curve. Consequently, for a given navigation target, it has been observed that the arrival time (total time) at the target is more efficient on paths produced using the Bezier curve across all scenarios.

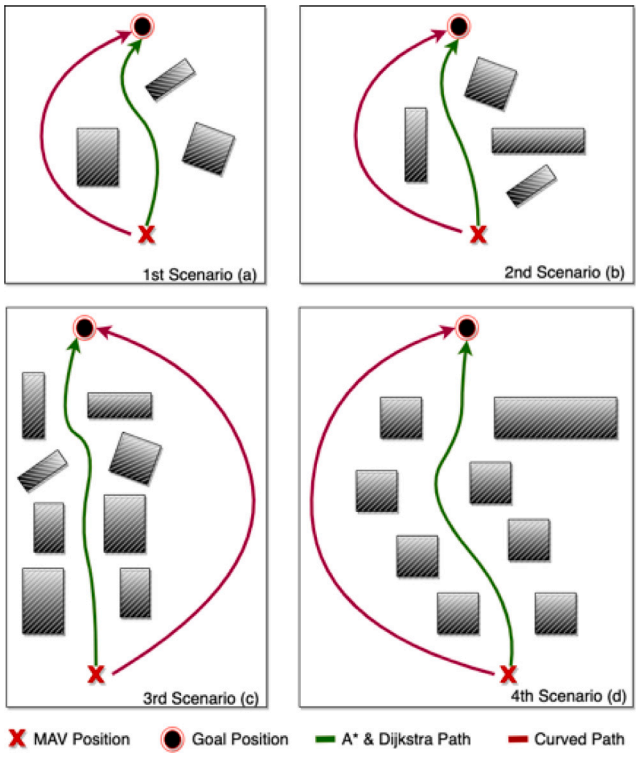

Fig. 11. 4 different scenarios created for performance comparison of Bezier-based local plan.

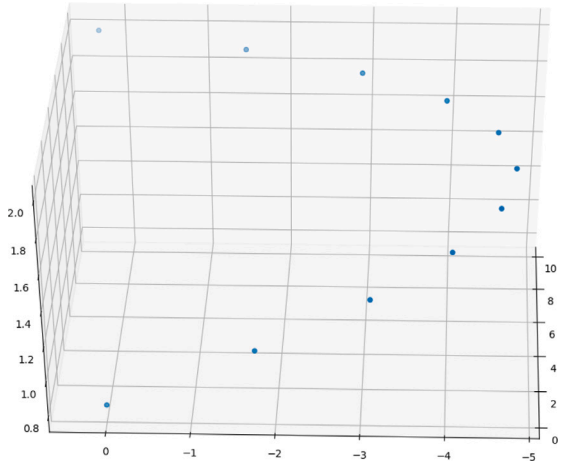

Fig. 12. Bezier curve for 3rd scenario.

In summary, for MAV platforms, which allocate a significant portion of their energy to remain airborne, expediting missions, even by a few seconds, is of utmost importance.

3.3. The experimental environment and results for local plan extraction based on multi-hop Bezier curves

The same fields produced for the single-hop Bezier curve tests were used to test the multi-hop Bezier curves. As a difference, walls were built around the areas used for single-hop experiments, making it impossible to find a single-hop solution (Fig. 13). The results obtained after performing the experiments in the environment given in Fig. 13 are given in Table 2.

Table 1
Path calculation, path implementation and total time comparison table of A*, Dijkstra and Bezier curve methods used as local planners (Times are in seconds).

Scenario #	Dijkstra path			A* path			Bezier curve path		
	Calc. Time	Exec. Time	Total time	Calc. Time	Exec. Time	Total time	Calc. Time	Exec. Time	Total time
1st (a)	3,6	12,3	**15,9**	2,3	12,3	**14,6**	6,7	3,8	**10,5**
2nd (b)	4,2	16,1	**20,3**	2,4	16,1	**18,5**	6,6	4,2	**10,8**
3rd (c)	8,1	37,1	**45,2**	5,3	37,1	**42,4**	21,3	6,2	**27,5**
4th (d)	11,2	38,6	**49,8**	9,1	38,6	**47,7**	24,4	7,4	**31,8**

Table 2
Path calculation, path implementation and total time comparison table of multi-hop Bezier curve methods used as global planners (Times are in seconds).

Scenario		1st (a)	2nd (b)	3rd (c)	4th (d)
1-Hop Bezier curve	Calc. Time	10,6	11,2	24,3	45,0
	Exec. Time	–	–	–	–
	Total time	–	–	–	–
2-Hop Bezier curves	Calc. Time	10,1	10,8	17,8	40,4
	Exec. Time	4,3	4,4	–	–
	Total time (1+2)	**29,3**	**32,4**	–	–
	Total time (Only 2)	**14,4**	**15,2**	–	–
3-Hop Bezier curves	Calc. Time	13,1	13,0	37,7	29,2
	Exec. Time	**5,2**	**5,1**	–	**14,9**
	Total time (1+2+3)	–	–	–	**129,5**
	Total time (Only 3)	**18,3**	**18,1**	**79,8**	**44,1**

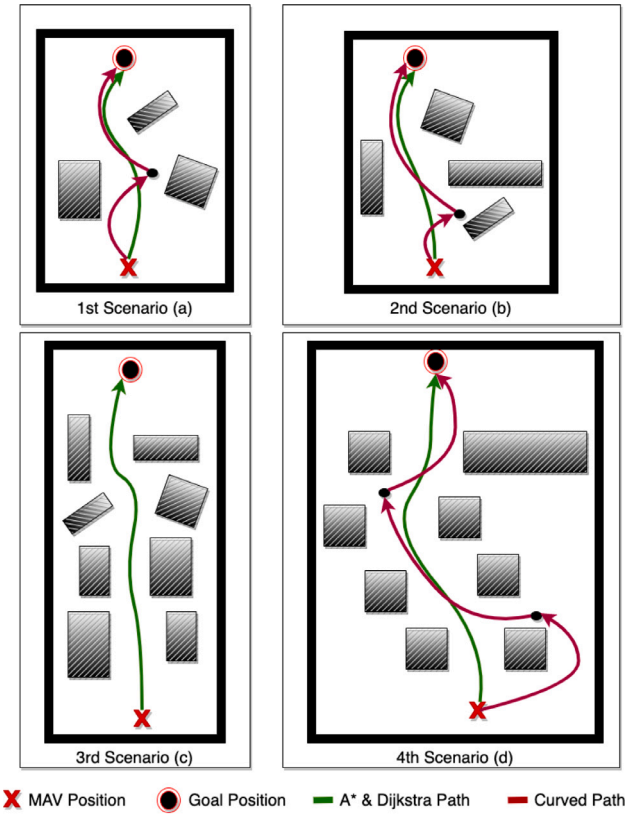

Fig. 13. 4 different scenarios created for multi-hop Bezier curve performance comparison.

The results in Table 2 are presented with two distinct evaluation criteria. The first set of results reflects instances where no paths for (n-1) hops were found when the navigation path was calculated with N hops. In these cases, when examining the results produced with 1 hop, the elapsed times are indicated in red, signifying that no navigation path could be established from the starting point to the destination with a single hop for all scenarios.

When examining the results produced by 2-hop navigation, it was found that a navigation path could be calculated in Scenario 1 and Scenario 2, but no navigation path could be calculated in Scenario 3 and Scenario 4. In Scenario 3 and Scenario 4, using 1 and 2 hops, the path was not found, and the time taken to find the path is marked in red.

One notable observation is that in Scenario 3, while no path was found after 24.3 s using 1-hop, a path was found in 17.8 s with 2-hop. This can be attributed to the narrowing of the search space of the method used to determine hop points. However, this occurrence is rare and is typically observed in situations with minimal obstacles in the environment.

In the 1st and 2nd scenarios, routes were found with 2-hop, and the times taken to find the routes, the times taken to follow these navigation routes with a micro air vehicle, and two different total elapsed times are provided. In cases where the attempt to find the navigation route with 2-hop was conducted after failing to find the navigation path with 1-hop, a total time of 29.3 s was reached. On the other hand, the time taken to find the way using only 2-hop and reach the destination by following the path found was calculated as 14.4 s.

When comparing these times, the 29.3 s navigation duration was notably longer than the durations of 15.9 s with Dijkstra and 14.6 s with A*, both of which did not involve using Bezier curves. Conversely, it was observed that navigating using only a 2-hop search took 14.4 s, which is significantly faster, denoted in green as it outperformed the Dijkstra and A* methods.

When examining the results produced by 3-hop navigation, it is worth noting that there are no results for Scenario 3, but results are available for Scenario 1, 2, and 4. In the cases of Scenario 1 and 2, where results could already be obtained with 2-hop, only the results with 3-hop are provided, and the times are given individually.

For Scenario 4, where a 1-hop path could not be found initially, the total time spent on this attempt was 45 s. In the case of 2-hop, when a path could not be found, the total time was observed to be 40.4 s. A total of 29.2 s was taken to find the path with 3-hop. Additionally, it took 14.9 s to execute the path found with 3-hop. When summing up all these times, a total time of 129.5 s was observed, significantly longer than the results produced by the Dijkstra method in 49.8 s and the A* method in 47.7 s.

In Scenario 4, when searching solely with 3-hop and reaching the destination using the navigation path found, the elapsed time was measured at 44.1 s. This time is shorter than the times of both the Dijkstra and A* methods. While creating and following the navigation path using only 3-hop outperformed the Dijkstra and A* methods in the 2nd scenario, the times in the 1st scenario were also better than the Dijkstra and A* methods

When examining the results from these tables, it becomes evident that effective navigation strategies should be tailored to the specific environment. To harness the potential of Bezier curves, it is crucial to consider the environment's congestion and determine the appropriate number of hops and the navigation path creation approach.

It is noteworthy that notably better results are achieved compared to using Dijkstra and A* methods when using Bezier curves with a fixed

Table 3
Environment complexity calculation results using different cylinder sizes over voxel volume sizes.

Sce. #	Cylinder width	Multi-Hop Bezier		Waypoint	
		Expected	Calculated	Expected	Calculated
1	x5	✓	✓		
	x7	✓	✓		
	x9	✓			✓
2	x5	✓	✓		
	x7	✓	✓		
	x9	✓	✓		
3	x5			✓	✓
	x7			✓	✓
	x9			✓	✓
4	x5	✓			✓
	x7	✓			✓
	x9	✓	✓		

Table 4
Environment complexity calculation results using different cylinder sizes over voxel counts.

Scenario #	Cylinder width	Common result	Calculation time
1	x5	✓	0.12
	x7	✓	0.15
	x9	✓	0.31
2	x5	✓	0.21
	x7	✓	0.29
	x9	✓	0.41
3	x5	✓	0.61
	x7	✓	0.93
	x9	✓	1.13
4	x5		0.59
	x7		0.91
	x9	✓	1.17

number of hops. It is essential to understand that simply increasing the number of hops does not necessarily yield improved results, and in cases where the navigation path is not found, it is challenging to surpass the performance of Dijkstra and A* methods.

In some experiments, results were attempted using 4-hop. However, these results took longer to compute than the other two methods and were, therefore, omitted. The pathfinding process becomes considerably more time-consuming when the Euclidean distance to the target exceeds 11 m. Therefore, test results beyond this distance were excluded from the analysis.

3.3.1. Evaluation of results for local plan extraction based on multi-hop Bezier curves

When a navigation path can be successfully generated using Bezier curves with a single hop, it consistently outperforms the Dijkstra and A* methods in various scenarios. In cases where a path is not found, the total time taken to search for results with 2-hop and 3-hop is generally less than the total navigation time required by the Dijkstra and A* methods. Notably, the navigation paths obtained with just 2-hop and 3-hop often exhibit greater efficiency than those produced by the Dijkstra and A* methods in many instances.

3.4. The experimental results for local plan switching through environmental complexity

The developed method was tested in the area depicted in Fig. 13, with the MAV's width being 66 cm. Voxel search space tests were conducted with cylinder diameters ranging from 5 to 9 times the size of the MAV. These values were determined while taking into account the characteristics of Bezier curves. It is important to note that Bezier curves cannot be extracted when the cylinder width exceeds 9 times that of the MAV. Fortunately, the method's runtime was consistently under 100 ms for each case, rendering it negligible for comparison purposes.

When examining the table, it becomes evident that optimal choices can be made for the 2nd and 3rd scenarios. In the 1st scenario, successful detection of empty areas was achieved in tests with cylinder sizes of 5x and 7x. Conversely, in the 4th scenario, successful detection of empty areas was observed in the test with the largest cylinder size.

In certain instances, the correct method is chosen when a smaller cylinder area is determined, while in others, the correct method may be selected when a larger cylinder area is defined. Overall, it has been observed that this method yields an accuracy rate of 75% in correctly assessing the complexity of the environment.

In the first method, the size and dimensions of the voxels are of importance. In the second method developed, the size of the voxels within the cylinder area becomes less significant. Instead, the calculation is based on the ratio of empty to occupied areas within the cylinder area, measured in m^2. Complexity of the environment is then determined by filtering this ratio using empirically determined threshold values, which in turn helps in choosing between Bezier curves or a waypoint-based local planner.

A virtual cylindrical area is defined extending from the MAV's location to the target point, as illustrated in Fig. 8. The occupancy and voids within this area were evaluated through experiments for different cylinder sizes (5, 7, and 9 times). While the first method's iterative search on the Octree data structure has a low time complexity, the second method required a 3D matrix-based search, resulting in non-negligible computation times. The test results in terms of computation times are presented in the following table.

The results are illustrated in Table 4, where expected and calculated outcomes for each method are provided separately, as shown in Table 3. In this table, all expected situations are presented. A checkmark was assigned when the expected and calculated results in Table 4 matched, and no checkmark was used when they did not align. Furthermore, computation times in seconds are included in this table.

Upon examining the table, it becomes evident that as the distance to the target and the cylinder diameter increase, the computation times also rise. However, this increase in computational cost was accompanied by improved performance, jumping from 75% to 83%

In conclusion, two distinct systems have been developed, allowing for the seamless transition between Bezier curves and waypoints based on the environmental complexity. These systems exhibit similar success rates. To assess the method switching capabilities in a larger environment, a simulated area, as shown in Fig. 14, was constructed within the Gazebo simulation environment

In this context, the OctoMap of the environment constructed before the test is depicted in Fig. 15. These images showcase both a top-down view of the map and a side profile. Fig. 14 provides insight into the MAV's navigation journey. The MAV receives commands to traverse from the initial position denoted by the red X to the destination indicated by the green X. Intermediate waypoints along this path are identified by figures annotated with a black dot and encircled by a red boundary. The MAV's task involves navigating to each designated waypoint. It is noteworthy that, before commencing the navigation, a crucial decision must be made regarding the choice of the local planner. The method outlined above was employed to determine the appropriate local planner for this purpose.

Fig. 14 provides a comprehensive overview of the selected method for each intermediate target. These intermediate target points are numbered in blue and span a total of 7 consecutive targets. The paths highlighted in green indicate the utilization of a waypoint-based local planner, while the paths in red represent navigation with a local planner based on Bezier curves. In cases where Bezier curves are employed as the local planner and it is determined that reaching the target necessitates more than one hop, the hop zones are visualized by black dots.

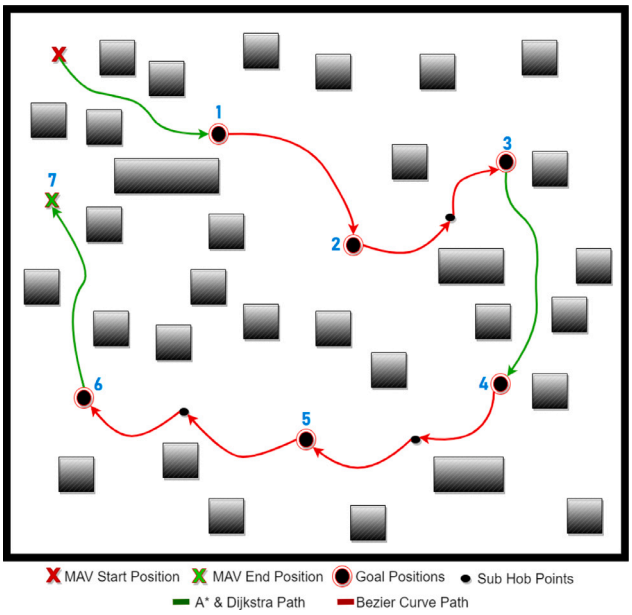

Fig. 14. Bezier and waypoint based local plan transition test area illustration over environment complexity.

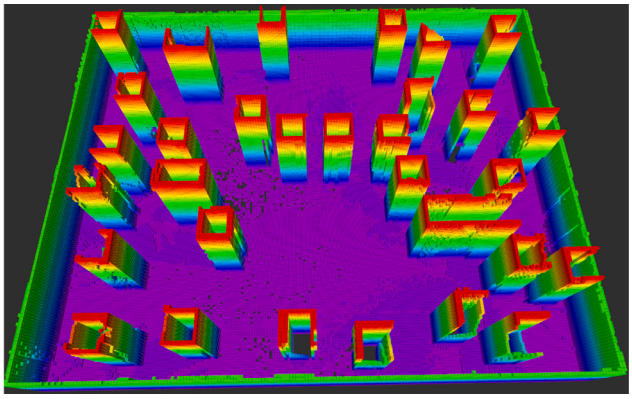

Fig. 15. Bezier and waypoint based local plan transition test area OctoMap over environment complexity.

Upon closer examination of the results obtained with the developed method, a waypoint-based local planner was employed for navigating to the 1st target. When transitioning from the 1st to the 2nd target, the environment was assessed as having low complexity, and thus, single-hop Bezier curves were used. Navigating from the 2nd to the 3rd target involved the use of 2-hop Bezier curves. Subsequently, a waypoint-based local planner was utilized for the journey from the 3rd target to the 4th target. For the routes spanning from the 4th target to the 5th and from the 5th to the 6th, the method selected 2-hop Bezier curves as the preferred local planner. Lastly, the analysis indicates that the method chose a waypoint-based local planner while navigating from the 6th to the 7th destination.

Furthermore, when the intention was to reach all sub-goal points exclusively by utilizing Bezier curves, it was observed that a maximum 3-hop path could not be established from the initial state to the 1st target and from the 6th to the 7th target. Conversely, when the preference was for a waypoint-based approach, all sub-goal points were successfully reached. The cumulative elapsed times for each intermediate target in the experiments are documented in Table 5.

The times provided in this table are in seconds. When exclusively employing the waypoint-based local planner, all sub-goal points were

Table 5
Suggested method, waypoint based local planner only, multi-hop based local planner only and navigation result time comparisons.

Route	With waypoint local planner	With Multi-Hop Bezier curves	Proposed method with complexity switch
Start to 1	12.2	–	12.2
1–2	13.7	7.1	7.1
2–3	26.5	18.2	18.2
3–4	21.3	19.1	21.3
4–5	21.9	17.6	17.6
5–6	23.0	18.4	18.4
6–7	28.1	–	28.1
Total	**146.7**	**Could not complete**	**122.9**

reached in a total of 146.7 s. In contrast, when solely using Multi-hop Bezier curves, it is not possible to provide a total time because certain sections of the route could not be covered using Bezier curves. In the proposed method, the MAV first determines which local planner to utilize, and as a result, it reaches the intermediate points with the selected local planner. The entire route was navigated in a total of 122.9 s. In other words, it was able to complete the route 23.8 s faster than when exclusively using the waypoint-based local planner, representing a 16% difference. Efficiently managing battery usage in MAVs is of utmost importance, and a 16% reduction is a considerable improvement.

The study's results highlight the importance of autonomous software developers having a strong understanding of MAV flight dynamics to enhance performance. The research demonstrates that curvilinear path planning with Bezier curves can significantly improve navigation performance. The study also provides a detailed explanation of how multiple Bezier curves can be used sequentially, depending on the complexity of the environment. This approach has been shown to reduce flight times, increase computation times, and ultimately shorten the total mission execution time. This indicates that well-informed and innovative approaches to MAV navigation can lead to more efficient and effective autonomous systems.

The study's findings suggest that shorter navigation times might be sufficient for MAVs to complete their tasks effectively. The study determined the complexity of the environment based on the MAV's map without prior knowledge. However, future studies could benefit from operators providing predefined values for the environment's complexity. This could help reduce the time required to calculate complexity and improve overall navigation efficiency.

Plans were made assuming that once a navigation target was given, the MAV would never land. In waypoint-based planners, since MAVs move very slowly, in most cases waiting for them to land will waste time. However, there is a significant calculation time for the developed Bezier curves-based planners. During this period, techniques can be developed such as the MAV landing on the ground and making calculations as if it were in the air, and when it completes the calculation, it takes off and follows the curve. This situation can be compared to automatic engine START–STOP techniques used for automobiles. Of course, in addition to this, various methods will need to be used to check whether the MAV is suitable for landing on the ground.

In addition to all these, blind search techniques were used to determine the most appropriate Bezier curve to follow. Instead, if optimization is done with heuristic-based optimization approaches, it may be possible to shorten the calculation time and therefore reduce the total flight time. Studies on this are being carried out and are considered as future studies.

4. Conclusions

Within the scope of this study, a navigation algorithm suitable for driving dynamics has been developed for micro air vehicle. The

necessity of a navigation method after the production of costmaps is the preparation of global and local plans. 3D A* and Dijkstra methods have been developed for global plan extraction. In order for the robot to move from its current location to the given target location, first a global plan was drawn up, and then this global plan was divided into waypoints and these waypoints were given to MAVs moving with position control. A method suitable for the driving dynamics of MAVs has been developed as a result of the slow movement of MAVs that move over waypoints with position control, especially in uncomplicated environments. While MAVs can move based on waypoints with position control, they can also follow a given curve. For this reason, it has been investigated whether a curve can be produced for a given target using Bezier curves, and it has been observed that curves can be produced to follow MAVs over Bezier curves by performing tests in the simulation environment. With this new technique developed, MAVs can reach the target more quickly over curves, especially in environments with relatively low complexity.

However, it may not always be possible to move to the target using a single curve, especially in complex environments. Considering this situation, multi-hop Bezier curves approach has been developed in order to reach the target position with more than one curve tracking. If it is not possible to reach the given destination with a single hop, curves are produced to reach the given destination with 2 hops, and if it cannot be reached with 3 hops. In many cases, it takes longer to generate curves with 4 hops than to track waypoints generated with A*, so calculations are made with a maximum of 3 hops. In cases where the environment complexity is very high, the target point may not be found with 3 hops. In such cases, since it takes a long time to generate waypoints by calculating the path with A* again, a switchable method has been developed to use a local planner based on waypoints and Bezier curves, depending on the complexity of the environment. The cumulative cost is calculated over the maps with the complexity of the environment with the help of an equation, and if a target is given in a high complexity environment, waypoint-based targets are produced and navigation of the MAVs is provided by position control, while in environments with low complexity, multi-hop Bezier curves are calculated and the MAVs are calculated over the curves achieved to reach the given target.

As a result, new approaches have been brought to the literature on navigation approaches on micro air vehicles. In addition to this study, in the future, in the calculation of multi-hop curves, searching for the positions of the intermediate hops with an intuitive method can make the developed method work faster, so this point can be focused on. In addition, costmaps used to calculate environment complexity may not always produce the correct result, with a low probability. The discovery of new methods to calculate this complexity can be listed among future studies.

CRediT authorship contribution statement

Furkan Cakmak: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Writing – original draft, Visualization. **Sirma Yavuz:** Supervision and modification for the final layout.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was supported by the Turkish Scientific and Technical Research Council (Project no: 118E215) and Yildiz Technical University Scientific Research Projects Coordination Department (Project no. FDK-2017-3044).

The authors would like to thank the editors and anonymous reviewers for providing insightful suggestions and comments to improve the quality of research paper.

References

[1] P. Yang, K. Tang, J.A. Lozano, X. Cao, Path planning for single unmanned aerial vehicle by separately evolving waypoints, IEEE Trans. Robot. 31 (5) (2015) 1130–1146, http://dx.doi.org/10.1109/TRO.2015.2459812.

[2] H. xia Xia, Y. en Chen, Y. fei Miao, R. qi Luo, A quality assessment model for unmanned aerial vehicle path planning based on Bayesian networks, in: 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2015, pp. 849–853, http://dx.doi.org/10.1109/FSKD.2015.7382053.

[3] F.H. Tseng, T.T. Liang, C.H. Lee, L.D. Chou, H.C. Chao, A star search algorithm for civil UAV path planning with 3G communication, in: 2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2014, pp. 942–945, http://dx.doi.org/10.1109/IIH-MSP.2014.236.

[4] I. Lluvia, E. Lazkano, A. Ansuategi, Active mapping and robot exploration: A survey, Sensors 21 (7) (2021) 2445, http://dx.doi.org/10.3390/s21072445.

[5] S. Shen, N. Michael, V. Kumar, Autonomous multi-floor indoor navigation with a computationally constrained MAV, in: 2011 IEEE International Conference on Robotics and Automation, 2011, pp. 20–25, http://dx.doi.org/10.1109/ICRA.2011.5980357.

[6] S. Liu, M. Watterson, S. Tang, V. Kumar, High speed navigation for quadrotors with limited onboard sensing, in: 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 1484–1491, http://dx.doi.org/10.1109/ICRA.2016.7487284.

[7] S. Weiss, D. Scaramuzza, R. Siegwart, Monocular-SLAM-based navigation for autonomous micro helicopters in GPS-denied environments, J. Field Robot. 28 (2011) 854–874, http://dx.doi.org/10.1002/rob.20412.

[8] G. Loianno, J. Thomas, V. Kumar, Cooperative localization and mapping of MAVs using RGB-D sensors, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 4021–4028, http://dx.doi.org/10.1109/ICRA.2015.7139761.

[9] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I.L. Grixa, F. Ruess, M. Suppa, D. Burschka, Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue, IEEE Robot. Autom. Mag. 19 (3) (2012) 46–56, http://dx.doi.org/10.1109/MRA.2012.2206473.

[10] E. Maset, L. Scalera, A. Beinat, D. Visintini, A. Gasparetto, Performance investigation and repeatability assessment of a mobile robotic system for 3D mapping, Robotics 11 (3) (2022) http://dx.doi.org/10.3390/robotics11030054.

[11] B. Guo, H. Dai, Z. Li, W. Huang, Efficient planar surface-based 3D mapping method for mobile robots using stereo vision, IEEE Access 7 (2019) 73593–73601, http://dx.doi.org/10.1109/ACCESS.2019.2920511.

[12] D. Wang, T. Fan, T. Han, J. Pan, A two-stage reinforcement learning approach for multi-UAV collision avoidance under imperfect sensing, IEEE Robot. Autom. Lett. 5 (2) (2020) 3098–3105, http://dx.doi.org/10.1109/LRA.2020.2974648.

[13] Y. Chang, H. Zhou, L. Shen, Q. Fang, T. Hu, Multi-UAV binocular intersection with one-shot communication: Modeling and algorithms, IEEE Access 7 (2019) 124902–124913, http://dx.doi.org/10.1109/ACCESS.2019.2928596.

[14] A. Bahabry, X. Wan, H. Ghazzai, H. Menouar, G. Vesonder, Y. Massoud, Low-altitude navigation for multi-rotor drones in urban areas, IEEE Access 7 (2019) 87716–87731, http://dx.doi.org/10.1109/ACCESS.2019.2925531.

[15] H. Deng, Q. Fu, Q. Quan, K. Yang, K.-Y. Cai, Indoor multi-camera-based testbed for 3-D tracking and control of UAVs, IEEE Trans. Instrum. Meas. 69 (6) (2020) 3139–3156, http://dx.doi.org/10.1109/TIM.2019.2928615.

[16] V. Walter, N. Staub, A. Franchi, M. Saska, UVDAR system for visual relative localization with application to leader–follower formations of multirotor UAVs, IEEE Robot. Autom. Lett. 4 (3) (2019) 2637–2644, http://dx.doi.org/10.1109/LRA.2019.2901683.

[17] Y. Tang, Y. Hu, J. Cui, F. Liao, M. Lao, F. Lin, R.S.H. Teo, Vision-aided multi-UAV autonomous flocking in GPS-denied environment, IEEE Trans. Ind. Electron. 66 (1) (2019) 616–626, http://dx.doi.org/10.1109/TIE.2018.2824766.

[18] O. Saif, I. Fantoni, A. Zavala, Distributed integral control of multiple UAVs, precise flocking and navigation, IET Control Theory Appl. 13 (2019) http://dx.doi.org/10.1049/iet-cta.2018.5684.

[19] OctoMap - 3D occupancy mapping. URL: https://octomap.github.io/.

[20] octomap - ROS wiki. URL: http://wiki.ros.org/octomap.

[21] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, W. Burgard, OctoMap: An efficient probabilistic 3D mapping framework based on octrees, Auton. Robots 34 (2013) http://dx.doi.org/10.1007/s10514-012-9321-0.

[22] X. Wang, Y. Mizukami, M. Tada, F. Matsuno, Navigation of a mobile robot in a dynamic environment using a point cloud map, Artif. Life Robot. 26 (2020) http://dx.doi.org/10.1007/s10015-020-00617-3.

[23] L. Chang, L. Shan, C. Jiang, Y. Dai, Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment, Auton. Robots 45 (2021) http://dx.doi.org/10.1007/s10514-020-09947-4.

[24] C. Wang, X. Chen, C. Li, R. Song, Y. Li, M.Q.-H. Meng, Chase and track: Toward safe and smooth trajectory planning for robotic navigation in dynamic environments, IEEE Trans. Ind. Electron. 70 (1) (2023) 604–613, http://dx.doi.org/10.1109/TIE.2022.3148753.

[25] H. Kivrak, F. Çakmak, H. Kose, S.r. Yavuz, Waypoint based path planner for socially aware robot navigation, Cluster Comput. 25 (2022) http://dx.doi.org/10.1007/s10586-021-03479-x.

[26] R. Ramamurti, W. Sandberg, R. Lohner, Simulation of the dynamics of micro air vehicles, in: 38th Aerospace Sciences Meeting and Exhibit, 2000, http://dx.doi.org/10.2514/6.2000-896.

[27] H. Li, Y. Luo, J. Wu, Collision-free path planning for intelligent vehicles based on Bézier curve, IEEE Access 7 (2019) 123334–123340, http://dx.doi.org/10.1109/ACCESS.2019.2938179.

[28] L. Chen, D. Qin, X. Xu, Y. Cai, J. Xie, A path and velocity planning method for lane changing collision avoidance of intelligent vehicle based on cubic 3-D Bezier curve, Adv. Eng. Softw. 132 (2019) 65–73, http://dx.doi.org/10.1016/j.advengsoft.2019.03.007.

[29] S. Feng, A. Abuaish, P.A. Vela, Safer gap: A gap-based local planner for safe navigation with nonholonomic mobile robots, 2023, arXiv:2303.08243.

[30] I.-A. Somitca, S. Brad, V. Florian, S.-E. Deaconu, Improving path accuracy of mobile robots in uncertain environments by adapted Bezier curves, Electronics 11 (2022) http://dx.doi.org/10.3390/electronics11213568.

[31] Y.S. Aljamali, M.J.A. Safar, Smooth and collision-free path planning for holonomic mobile robot based RRT-connect, J. Phys. Conf. Ser. 2550 (1) (2023) 012032, http://dx.doi.org/10.1088/1742-6596/2550/1/012032.

[32] B. Siciliano, O. Khatib, Springer Handbook of Robotics, Springer-Verlag, Berlin, Heidelberg, 2007.

[33] D. Steffi, S. Mehta, V. K. a, S. Dasari, Robot path planning prediction, a multidisciplinary platform, a survey, 2021, pp. 211–219.

[34] T.W. Sederberg, R.T. Farouki, Approximation by interval Bézier curves, IEEE Comput. Graph. Appl. 12 (5) (1992) 87–95.