ENHANCING UNMANNED ROTORCRAFT GUIDANCE WITH LIDAR AND ADS-B INTEGRATED PGFLOW ALGORITHM

Jan Rudzki *
Technical University of Munich
Munich, Germany

Zeynep Bilgin †
Technical University of Munich
Munich, Germany

İlkay Yavrucuk ‡
Technical University of Munich
Munich, Germany

This paper details the development of a Detect and Avoid (DAA) system for UAVs using potential flow theory to avoid static and dynamic obstacles. The research features the integration of an existing Potential Guidance Flow algorithm into a ROS2-based DAA system with SLAM-based environment mapping and PX4 Autopilot. Utilizing LiDAR and ADS-B data, the system detects collision risks and generates avoidance commands, applicable to various UAV platforms and supporting real-world experiments. The system's effectiveness is validated through Software-in-the-Loop simulations in Gazebo, which realistically models sensor noise, measurement uncertainties, and vehicle dynamics. The evaluation includes diverse scenarios, from navigation around convex obstacles to navigation in complex urban environments. Experiments test system responsiveness, obstacle detection accuracy, and maneuver reliability at different speeds. Results show that the PGFlow algorithm, combined with SLAM-based mapping, allows multicopter UAVs to effectively avoid obstacles in unknown environments. The mapping function accurately represents static obstacles and identifies collision risks in time to initiate avoidance maneuvers. The ROS2 framework efficiently manages computational demands, ensuring low latency in command execution.

1 INTRODUCTION

The advent of unmanned rotorcraft, or unmanned aerial vehicles (UAVs) in general, marks a transformative era in modern logistics and aerial mobility. Recently, in Germany, for example, a cargo drone system received nationwide operational authorization for the commercial transport of goods along predefined routes in February 2024 [1]. This highlights the potential of unmanned rotorcraft to navigate infrastructure challenges, playing a pivotal role in enhancing the resilience and efficiency of supply chains. Apart from the classical transportation use cases, there are numerous other scenarios where UAVs come into play. These include, but are not limited to, search and rescue [2, 3], precision agriculture [4], construction [5], inspection [6], remote sensing [7], and surveillance [8]. Innovations in the fields of battery technology, sensing equipment, and flight control constantly broaden the spectrum of possible applications [9, 10]. At the center of this technological leap is autonomy, playing a critical role not only in making operations more efficient but also in ensuring safety and reliability in increasingly congested airspace [9, 11]. In this context, autonomous navigation that provides the necessary functionality to detect and avoid static and dynamic obstacles is essential to exploit the full potential of UAVs in civil applications and to minimize the need for human supervision [12].

Neither the Organization (ICAO) nor the European Union Aviation Safety Agency (EASA) stipulate how the capabilities of DAA systems must be implemented

* Graduate Student, jan.rudzki@tum.de
† PhD Candidate, zeynep.bilgin@tum.de
‡ Professor, ilkay.yavrucuk@tum.de

directly [13, 14]. For low-risk applications, developers are only obliged to demonstrate compliance with EU regulations by disclosing their DAA method [14]. Hence, there are many approaches presented in literature that solve the problem differently. However, researchers are unanimous about the three primary functions of sensing, detection, and avoidance which constitute a DAA system [13, 15–18]. This paper addresses the challenge by integrating a potential flow theory-based guidance algorithm into a fully functional detect and avoid (DAA) system for UAVs. The so-called potential guidance flow algorithm (PGFlow), originally developed by Bilgin et al. [19], approximates the fluid flow around static and dynamic obstacles in the UAV's operational space and yields a velocity vector that serves as input for a low-level flight controller. Its performance was proven in idealized offline simulations [20] and also real-world experiments with a small multicopter [20] as well as a fixed-wing micro UAV [21]. However, it was never combined with a sensing and detection function and is therefore only applicable to known environments yet.

PGFlow is among the artificial potential field methods towards DAA. Given the high relevance for unmanned rotorcraft, there are numerous other approaches presented in literature [18]. For instance, optimization-based methods transform the DAA functionality into an optimal control problem that generates, once solved, an avoidance trajectory [18]. Conversely, geometric methods adopt a more straightforward approach by only optimizing pre-defined evasive maneuvers based on the estimate of the flight paths of intruder aircraft or the position of static obstacles en route [22].

This research aims to extend the functionality of the existing PGFlow algorithm [23, 24] proposed by Bilgin et al. by adapting it to process sensor data for enhanced obstacle detection, making obstacle databases obsolete. While the use of LiDAR (Light Detection and Ranging) sensors for static obstacle detection is a known approach in terrestrial robotics [25], its adaptation to airborne systems poses unique challenges [26]. Similarly, the integration of ADS-B (Automatic Dependent Surveillance-Broadcast) for the identification of intruder aircraft extends the capabilities of existing potential flow theory-based DAA systems such as presented by Cruz and Garcia [27], marking a significant contribution to navigation safety for unmanned rotorcraft. For the integration to a DAA system, the original PGFlow algorithm is adapted and implemented as ROS2 system. By interfacing this framework with the PX4 Autopilot [28], it becomes readily available for real-world applications on different UAV platforms. Further, it enables software in the loop simulations (SITL) that streamline the entire development process and can be used to validate the performance in different scenarios. In this manner, the guidance logic with special focus on the ADS-B signal processing is transferred to ROS2 [29, 30], maintaining its real-time capability, and a supervisory flight-management system is developed to enable the use of the PGFlow guidance on go-to-target missions similar to drone delivery use cases.

2 METHODOLOGY

Integrating PGFlow to an overarching DAA framework requires various software modules and a method for evaluating the results. Firstly, the guidance algorithm must be embedded into a system that can process sensor readings and connect with a Software-in-the-loop (SITL) simulation environment. Further, a low-level flight controller is required to track the PGFlow's commands. The proposed architecture that fulfills these criteria is illustrated in Figure 1. Although simplified, it features all software modules that are part of the DAA system and the simulation environment. The symbol within the high-level control functions indicates that they are implemented in ROS2. This approach offers several advantages: Firstly, the extensive provision of different libraries or ROS2 packages that implement common functionalities such as SLAM accelerates the development progress. Another important advantage that results from offboard control with ROS2 is the separation of the flight controller firmware from computationally intensive tasks such as path planning. This is relevant for real-world applications, where the computational resources of the onboard flight controller are limited. The ROS2 system can run in parallel on a separate computer that is not subject to any weight or size limitations as it is only connected to the UAV via a command and control link. The final motivating factor for the distributed ROS2 approach is the ROS-client library which supports Python. This ensures consistency as the original PGFlow algorithm is also implemented in Python.

For executing guidance commands on a multicopter UAV, PX4 Autopilot was chosen as a low-level flight controller. Thanks to its peer-to-peer communication paradigm, PX4 connects well with ROS2 systems via a μXRDC-DDS agent and also features a SITL mode. Hence, this framework allows to test the DAA system in realistic conditions.

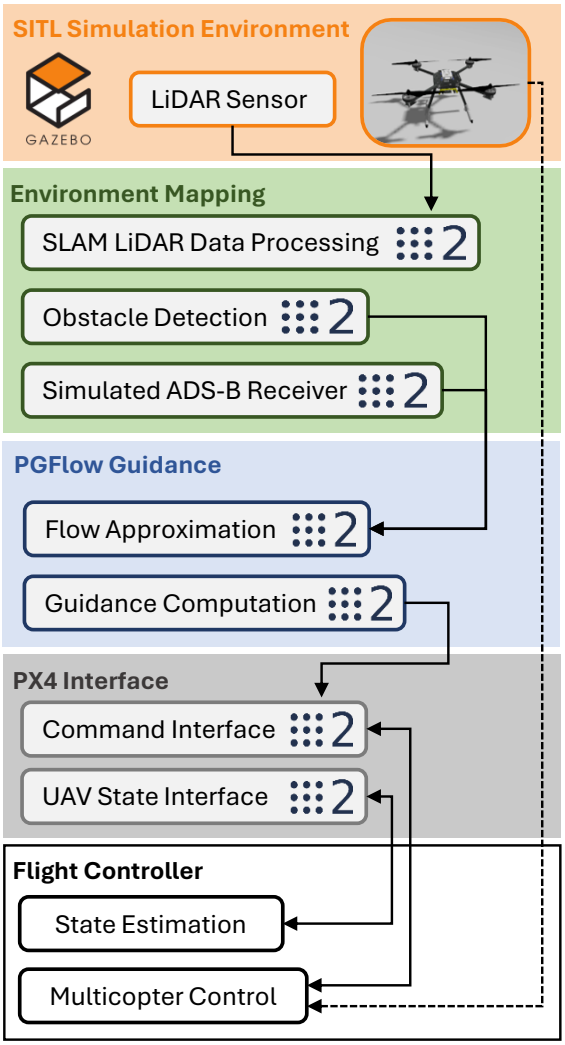


Fig. 1: Simplified System Architecture of the Proposed DAA Framework

This structure enables to assess the performance of the PGFlow algorithm when subject to an incomplete but continuously evolving environment mapping. Further, the x500 Gazebo UAV model provides realistic 6 DOF vehicle dynamics and helps to examine its influence on the guidance framework. According to the general structure of a DAA system, the following sections detail how the three functions, sensing, detection, and avoidance are implemented in this study.

3 DETECT AND AVOID SYSTEM

Neither the ICAO nor the EASA stipulates how the capabilities of DAA systems must be implemented [13, 14]. For low-risk applications, developers are merely obliged to prove compliance with EU regulations by disclosing their DAA method [14]. Thus, many approaches in the literature feature different techniques to ensure safe separation from obstacles and air traffic participants. However, researchers are unanimous about the three primary functions constituting a DAA system [13, 15–18]: sensing, detection, and avoidance.

3.1 PERCEPTION AND SENSING

The variety of suitable equipment for this task is large and heavily dependent on the use case and environmental conditions that the UAV will face [18]. Determining factors for selecting appropriate sensors are range, FOV, accuracy, measurement rates, and integrity [31]. The DAA system requires equipment for detecting other traffic participants, as well as static collision risks. For the detection of other aircraft, the system relies on ADS-B, which is a surveillance technology mandated by the ICAO for all aircraft and larger UAVs by 2020 [13]. It is based on transmitting the aircraft's state vector, such as position, velocity, and altitude, to other aircraft and ground stations [17]. ADS-B is a popular sensor for UAVs, as it is relatively cost-efficient, light, easy to implement, and has a long range of up to 240 km [15, 17].

To detect static obstacles, the system employs a 360° 2D LiDAR sensor. Continuous advancements have made these sensors more compact, lightweight, and affordable compared to RADAR systems. Additionally, their high resolution makes them ideal for identifying small obstacles [32]. However, LiDAR sensors are sensitive to weather conditions and cannot detect transparent objects like glass. Despite these limitations, the LiDAR sensor is the most suitable choice for this research's DAA system and can be effectively simulated in Gazebo.

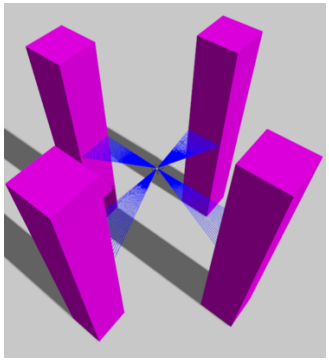
3.2 DETECTION

The detection function processes the sensor data generated in the previous step and extracts useful information to classify potential collision threads. The performance of these algorithms is vital for the entire system, as the accuracy of the subsequent path planning directly depends on it [15].

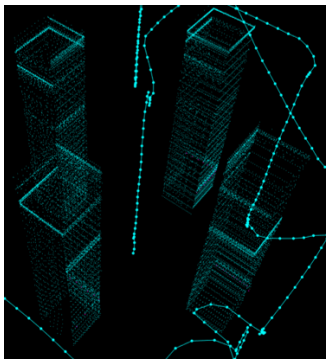
Processing the ADS-B signal is straightforward, as it directly provides the position information of other aircraft to the PGFlow algorithm without needing further processing. However, integrating LiDAR data into the avoidance function involves additional steps. For effective navigation in narrow, unknown environments, DAA systems using potential flow

theory require a detailed map of their surroundings to generate a guidance potential. Specifically, the PGFlow algorithm used in this research needs precise 2D outlines of obstacles. Therefore, the environment mapping function must transform sensor data into a format accessible by the DAA algorithm. Artificial potential field methods can include all obstacles in the potential field without evaluating collision risks. However, pruning static obstacles based on their relevance for the guidance computation can alleviate the computational demands.

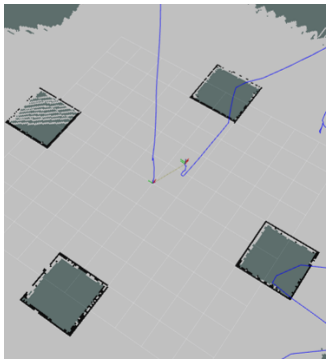
Generating a map of an unknown and cluttered environment is an integral task in the field of robotics [33, 34]. It is essential for various applications such as remote sensing [35], autonomous navigation [34], and environmental monitoring, i.e. 3D reconstruction [36]. This function spans its own research domain and is considered to be one of the most important topics in robotics [37]. It is commonly referred to as SLAM, which describes the process of creating a map of an unknown environment while simultaneously determining the robot's position relative to it [37]. In the context of this research, it is mainly employed for processing sensor measurements to generate a map of the environment. However, the localization function extends the applicability of the proposed DAA system to GPS-denied environments such as warehouses or production facilities [38]. Implementing an environment mapping function does not lie within the scope of this work. Fortunately, the ROS2 ecosystem provides library functions that implement a multitude of distinct SLAM approaches. First successful results are achieved with the RTAB-map ROS2 package [39, 40]. It is a versatile open-source library, maintained by Mathieu Labbé and François Michaud from the IntRoLab at Université de Sherbrooke. It features a graph-based SLAM approach and integrates various sensors to create a 3D point cloud map of the environment. To prevent it from exceeding real-time constraints, the framework includes a memory management approach that limits the map graph from uncontrolled growing, even in large-scale SLAM applications. For a detailed description of the methods implemented to RTAB-map, the reader is referred to the corresponding publication [39]. The integration into ROS2 offboard control is similar to both other frameworks presented above. RTAB-map directly subscribes to the */scan* topic created by the LiDAR sensor and receives odometry information from the PX4 interface module. Depending on its detection rate, the algorithm then updates a 3D point cloud map of the environment. Simultaneously, it creates a 2D occupancy grid that considers all points that are within a certain height range. In this manner, the 2D representation also includes 3D information that is either observed with a 3D LiDAR sensor or inferred from the vehicle's attitude. For a better understanding,



(a) Gazebo Simulation Environment



(b) RTAB-Map 3D Point Cloud



(c) RTAB-Map 2D Occupancy Grid

Fig. 2: Environment mapping with RTAB-Map in Gazebo: (a) Gazebo Simulation Environment, (b) RTAB-Map 3D Point Cloud, (c) RTAB-Map 2D Occupancy Grid.

Figure 2 presents the outcome of the RTAB-map applied in a Gazebo SITL simulation. The left image, Figure 2a, displays a simple simulation scenario with four magenta, box-shaped static obstacles specifically designed for testing the mapping function. The image shows an x500 UAV hovering right in the center while

capturing its environment with a LiDAR sensor that outputs 300 samples. Laser rays that are reflected from obstacles are highlighted as blue lines. The middle image, Figure 2b, shows the resulting 3D point cloud map after flying through the test scenario for a few seconds. The mapping is accurate and clearly identifies the four box-shaped obstacles. Additionally, the point cloud contains areas with increased density, appearing as lines within the outer walls of the obstacle. These occur when the UAV hovers at the same position for an extended period. Then, the laser rays hit the obstacles for multiple scan times at the same position, and the 3D point cloud map is updated with similar measurements repeatedly. If this period is long enough, the point cloud becomes so dense that it appears to be a 2D line. The blue, dotted curve in Figure 2b represents the vehicle's flight path, with each dot marking a location where the map was updated. The right image, Figure 2c, is a screenshot from rViz2, showing the simultaneously generated 2D occupancy grid. It uses three colors: light gray for free space, dark gray for unknown areas, and black for cells occupied by obstacles. As the colors are defined by values from 0 to 100, RTAB-map enables an even finer distinction. However, compared to a depth camera, LiDAR sensors provide reliable measurements, wherefore the results are quite clear [41]. The grid updates at the same rate as the 3D point cloud and is accessible to other ROS2 programs.

Further processing is necessary to provide the PGFlow algorithm with its desired input of obstacle vertices in 2D. Figure 3 visualizes the entire environment mapping pipeline, including the functions that interpret the 2D occupancy gird. More specifically, before clustering, the occupancy grid is converted into an array of 2D coordinates based on the grid size and resolution. Following the recommendations of Lohani and Ghosh [32], these points are clustered using the DBSCAN algorithm [42]. It identifies high-density regions and expands clusters from these areas [43]. A specified distance threshold is used to exclude outliers from the clusters [42]. The Euclidean distance is employed to measure sample density, with a minimum of five samples required to form a cluster. Compared to similar methods like OPTICS, DBSCAN is faster and has been shown to produce more accurate results in processing LiDAR data, as noted by other researchers [32]. Initial tests in this research corroborate these findings. Subsequently, each cluster is processed with the *ConvexHull* algorithm [44] to compute the vertices of its outline. This step is essential

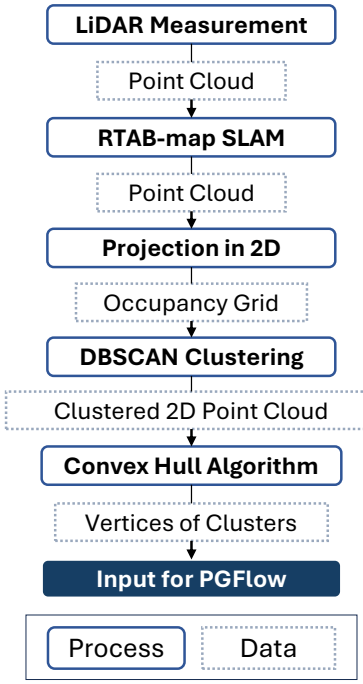


Fig. 3: Environment Mapping Pipeline

because the PGFlow algorithm requires obstacles to be represented as closed polygonal shapes. Additionally, this process reduces the number of coordinate points associated with an obstacle. To further relieve the computational load by neglecting irrelevant static obstacles, a distance-based pruning function is embedded in the detection function.

3.3 POTENTIAL GUIDANCE FLOW

In this paper the existing Potential Guidance Flow (PGFlow) algorithm is employed [23, 24, 45]. The PGFlow algorithm uses potential flow theory in fluid mechanics to generate a guidance vector field that mimics fluid flow. To calculate the vector field, the algorithm takes obstacle positions as input and divides the obstacle boundary into small line segments. Each segment is assigned a point

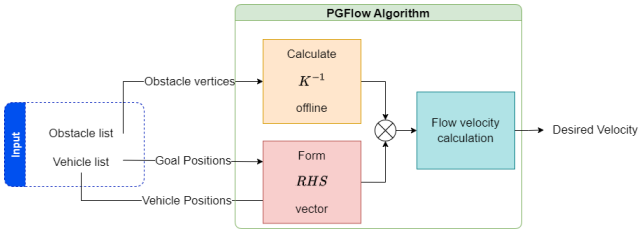


Fig. 4: Block diagram for PGFlow method. Obstacle and vehicle positions are fed to the guidance algorithm as input, and desired velocity vector that guarantees collision avoidance is generated.

vortex element with unknown strength which can be calculated by solving a system of linear equations in Equation 1.

$$K_{ij}\gamma_j = RHS_i \tag{1}$$

Here, K is a coefficient matrix and γ_j is the unknown vortex strength on element j. Components of K_{ij} are the normal velocity induced by jth element on ith element. The RHS vector contains the known flow components in the arena. These known flow components are point source elements attached to all vehicles to prevent collisions and a point sink element to attract the vehicle to the destination.
After solving Equation 1 for unknown vortex strengths, the flow velocity at any point can be calculated as in Equation 2 and Equation 3

$$u = \frac{-\sigma_g}{2\pi}\frac{x-x_g}{r_g^2} + \sum_{n=1}^{N}\frac{\sigma_n}{2\pi}\frac{x-x_n}{r_n^2} + \sum_{s=1}^{S}\sum_{k=1}^{K}\frac{\gamma_{sk}}{2\pi}\frac{x-x_{sk}}{r_{sk}^2} \tag{2}$$

$$v = \frac{-\sigma_g}{2\pi}\frac{y-y_g}{r_g^2} + \sum_{n=1}^{N}\frac{\sigma_n}{2\pi}\frac{y-y_n}{r_n^2} + \sum_{s=1}^{S}\sum_{k=1}^{K}\frac{-\gamma_{sk}}{2\pi}\frac{y-y_{sk}}{r_{sk}^2} \tag{3}$$

Here, S and K are number of obstacles(buildings) and number of elements on obstacle boundaries respectively. Position of a vortex element on a given obstacle is denoted as (x_{sk}, y_{sk}) and associated vortex strength is γ_{sk}. N is the total number of vehicles in the airspace and σ_n is the source strength of each vehicle to prevent collisions. Position of vehicles are denoted as $\boldsymbol{P}_n = (x_n, y_n)^T$. Last but not least, the goal position is denoted by $\boldsymbol{P}_g = (x_g, y_g)^T$ and modeled by a point sink element with strength σ_g.
A block diagram for the proposed guidance method is presented in Figure 4. The algorithm takes obstacle and vehicle positions as input and calculates the flow velocity at a point of interest. The flow velocity is calculated at each time-step and then fed into automatic flight control system as desired velocity.
Reader may refer to [19, 23, 24] for detailed derivation of method.

4 EXPERIMENTAL SETUP

To validate the proposed framework, two simulation scenarios of distinct complexity and scale were designed. From a simple environment with small static obstacles to a realistic scenario that models the skyline of Dubai, the setups are intended to test the DAA system under various conditions. The experimental procedure for each scenario is methodically designed to ensure consistency and reliability in the results. Each test run starts with initializing the Gazebo simulation environment and launching the PX4 instances in SITL mode. Following this, the ROS2 nodes for the PGFlow algorithm and environment mapping are activated. The initial positions and velocities of the guided and intruder vehicles are then set according to the specific parameters defined for each scenario.

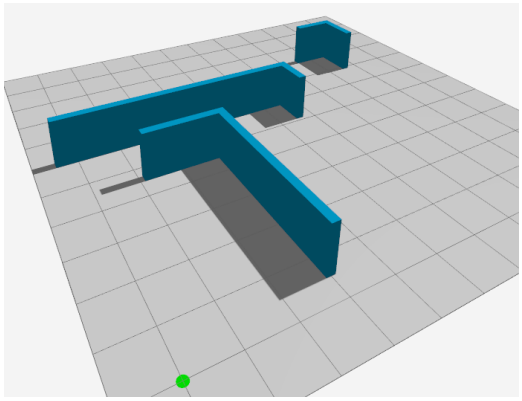


Fig. 5: Gazebo Simulation of Scenario A

During the simulation, the performance of the DAA system is closely monitored and logged. Key metrics tracked include the precision of obstacle detection, the accuracy of tracking velocity vector setpoints, and the frequency of guidance commands. The trajectory of the guided vehicle is also analyzed to evaluate its capability to navigate around obstacles and avoid collisions with other aircraft. Multiple runs are conducted for each scenario at different vehicle speeds to ensure robustness. This approach helps identify any inconsistencies or potential weaknesses.

Scenario A For a better impression, Figure 5 shows the Gazebo simulation environment A in 3D and Figure 6 displays its projection in 2D. It features three large concave obstacles which create a labyrinth-like structure for the UAV to pass through. The obstacles consist of basic box shapes that are defined in the SDF-file, which is parsed for loading the environment in SITL simulations. The environment does not correspond to any real-world scenario and is designed to test the DAA system's ability to deal with concavely shaped obstacles that are larger than the LiDAR sensor's range of 20 m. The initial position and the goal position are marked with a dot. Other parameters include the obstacle pruning distance of 30 m and the desired panel length for discretizing the static obstacles set at 0.3 m. For evaluating the avoidance of other UAVs, a second multicopter spawns and travels with a constant velocity vector that likely leads

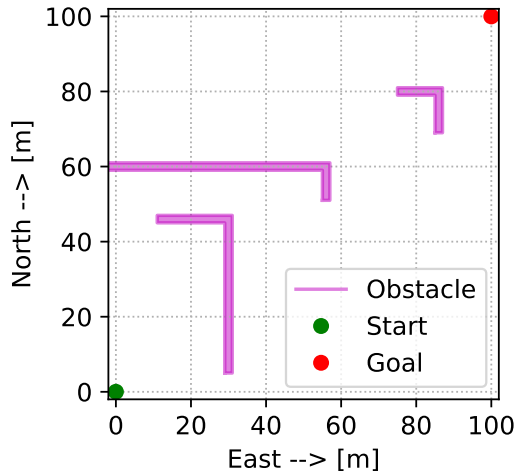


Fig. 6: 2D Map of Scenario A

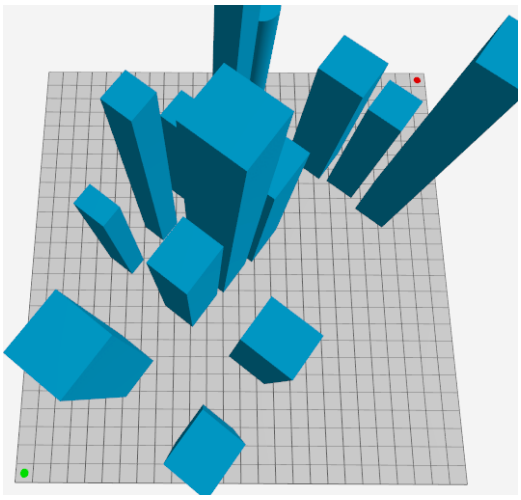


Fig. 7: 2D Map of Scenario B

to a collision.

Scenario B In contrast to scenario A, scenario B corresponds to a real-world environment. The Dubai Marina district is perfect for this analysis as the prevailing rectangular or circular foundation of the buildings simplifies modeling since they can be approximated as box shapes. Figure 7 illustrates the 2D map alongside its 3D simulation in Gazebo. This scenario includes high-rise buildings, narrow alleyways, and open spaces, presenting a variety of navigation challenges. The obstacles in this scenario are predominantly convex and of similar size, making it less complex in terms of individual obstacle difficulty compared to the previous scenario. However, the expansive environment introduces new challenges. The desired panel length must be carefully calibrated to balance accurate flow condition approximation with manageable computational demands. For example, using the same parameter settings (l = 0.3 m) as the challenging scenario during integration tests on

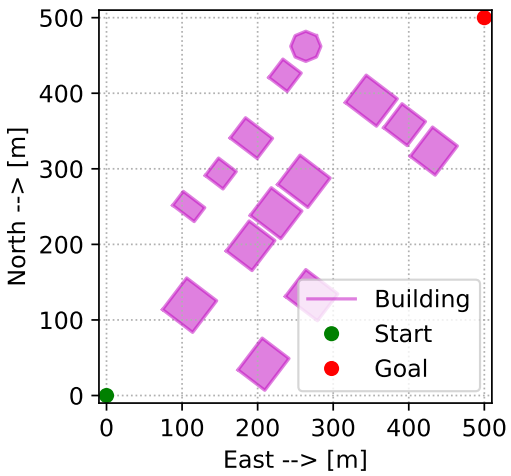


Fig. 8: Gazebo Simulation of Scenario B

the Dubai map led to a significant drop in the update frequency of the flow approximation. This evaluation is crucial for assessing the DAA system's sensitivity to guidance parameters and identifying the optimal configuration for operation in realistic environments. Additionally, this scenario allows for evaluating the effectiveness of the chosen LiDAR sensor in a practical setting. The performance of environment mapping can provide valuable insights into the suitability of the sensor for such applications, and the necessary range for the LiDAR sensor to ensure the DAA system's functionality in densely populated urban areas can be determined. To account for the larger scale, the LiDAR sensor's range is increased to 40 m, the desired panel length to 0.45 m, and the obstacle pruning distance to 60 m.

5 RESULTS AND DISCUSSION

The initial evaluation of static obstacle avoidance is conducted in a single-vehicle simulation. Key metrics such as velocity, map update frequency, and the number of obstacles and panels considered for path planning are recorded to assess the DAA system's performance. Subsequently, the framework is tested in a multi-vehicle simulation, focusing on avoidance behavior with other UAVs, thereby completing the assessment.

5.1 SIMULATION SCENARIO A

Figure 9 displays the results of two single-vehicle SITL simulations conducted for scenario A. The detected obstacles and UAV trajectories are plotted relative to a local NED frame, with the origin at the UAV's starting position $[0,0]$. In the first simulation, the velocity vector setpoint is set to 2.0 m/s, and the

resulting flight path is represented by the orange line in the upper image. In the second simulation, the speed is increased to 4.0 m/s, with the flight path also shown as an orange line in the lower image of Figure 9. The UAV's state is recorded every 0.8 seconds. Despite the large scale and concave

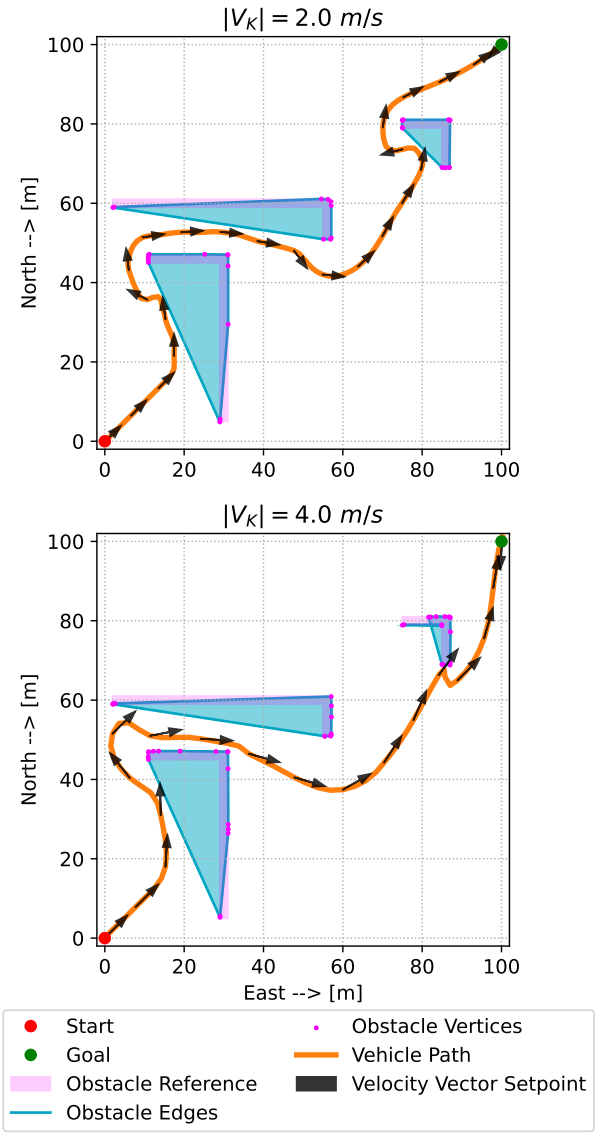


Fig. 9: SITL simulation results for scenario A at 2.0 m/s (upper) and 4.0 m/s (lower): The slower vehicle completes the course in 106.1 seconds, while the faster one finishes in 54.7 seconds. Both vehicles successfully avoid collisions with obstacles and reach their goal position. Except for the wall in the upper right corner, the mapping function transforms the concave walls into larger convex obstacles. This approach prevents PGFlow from encountering the inconsistent flow conditions typically found in corners.

shape of the obstacles, the DAA system successfully avoids them and reaches the goal position at both speed settings. As expected, the difference in velocity produces distinguishable paths: the slower x500 maintains a greater distance from obstacles and follows a smoother path, while the faster vehicle gets closer to obstacles and displays aggressive avoidance behavior with sharp turns and near misses. However, the higher velocity vector setpoint allows the UAV to reach the destination almost twice as quickly, completing the course in 54.3 seconds compared to 106.1 seconds at 2 m/s. To further evaluate the system's performance at higher speeds, the velocity vector setpoint is increased to 6.0 m/s. However, even without the additional computational demands from auxiliary functions for plotting, the simulated UAV often collides with obstacles from velocity vector setpoint magnitudes above 5.5 m/s.

The magenta dots in Figure 9 depict the outcome of the environment mapping function developed in this study. As outlined in Section 3.2, the DBSCAN clustering algorithm groups LiDAR measurements into obstacles, which are then converted into convex hulls. These hulls, highlighted in blue, represent how obstacles are perceived by the PGFlow algorithm. Consequently, the hulls do not precisely mirror the actual obstacle shapes, as the environment mapping module transforms concave walls into larger, convex obstacles, appearing as triangles in 2D. This deliberate simplification prevents the vehicle from entering concave regions where the approximated flow conditions are often inconsistent. If the obstacles were mapped accurately, the gradual environment mapping process, which doesn't capture the entire shape of large obstacles at once, would guide the UAV along the original shapes, exposing the algorithm to corner flow conditions and increasing the path length. This method is effective for the concave walls in the lower left corner of Figure 9. However, the third wall in the upper right corner is not transformed as expected due to an inaccurate DBSCAN clustering that splits the wall into two separate obstacles. Still, both fragments are considered for path planning, and the PGFlow algorithm navigates around them.

Despite the reliable avoidance performance in scenario A, the method has its limitations. For example, if the vehicle gets too close to the corner of a concave shape, it may end up enclosed by the convex hull. From the perspective of the PGFlow algorithm, the UAV would then be considered inside the obstacle, rendering the computed velocity vector setpoint ineffective. However, in scenario A, these concerns do not materialize, as the environment

mapping function detects the concave walls early enough to prevent the vehicle from entering the corners with a LiDAR range of 20 m.

Performance The evaluation proceeds with an emphasis on the performance of the DAA system. Figure 15 illustrates the frequency of environment mapping updates, i.e. gamma updates and the number of static obstacles and corresponding discretization panels employed by the PGFlow to generate a velocity vector setpoint. These results are derived from the simulation at 4.0 m/s, as previously examined in Figure 9. Contrary to expectations,

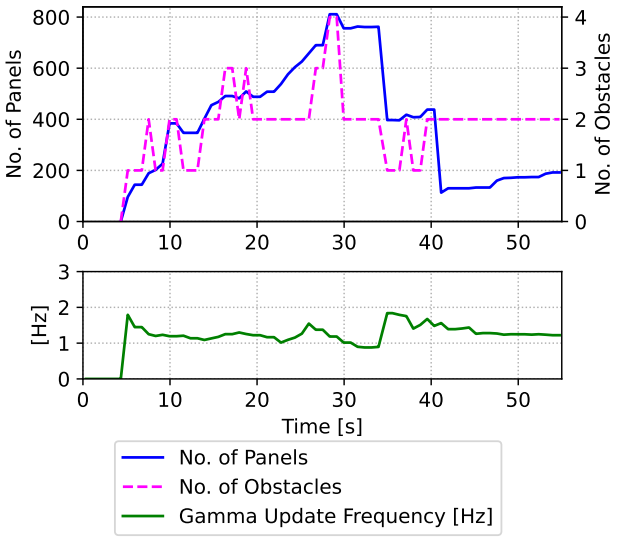


Fig. 10: The number of obstacles and panels considered for path planning in the challenging scenario is shown. The pruning strategy omits irrelevant obstacles, thereby reducing the size of the linear system to be solved for gamma. The decline in the green line after 25 seconds corresponds with the increasing number of panels. Consequently, a finer discretization of the obstacles adversely affects the DAA system's reaction time.

the magenta line in Figure 10 fluctuates between one and two Hz during the first 13 seconds, despite only one large obstacle being visible to the LiDAR sensor. This behavior is likely due to the DBSCAN clustering, which produces imperfect clusters of a single obstacle. Similar behavior is observed later as additional obstacles come into view for the first time. Generally, this inaccuracy does not impact the guidance, as the PGFlow algorithm treats all obstacles equally for path planning. The blue line, which corresponds to the number of obstacles, peaks at 806 panels at 28 seconds. Simultaneously, the frequency of gamma updates drops to a minimum of 0.9 Hz at 35 seconds. This correlation supports the assumption that the high number of panels involved in path planning causes delays in gamma calculations. When the number of panels drops suddenly, the update frequency recovers and remains stable for the rest of the simulation. As mentioned in Section 4, the desired panel size is set to 0.3 m. For reference, simulations with a smaller length of 0.15 m failed because the environment mapping algorithm did not update frequently enough to inform the flow calculation. Hence, these results also highlight the importance of the obstacle pruning mechanism, as reducing computational load improves the DAA system's reaction time.

Vehicle Dynamics As outlined in Section 3.3, the PGFlow algorithm does not account for vehicle dynamics and issues a velocity setpoint of constant magnitude. To assess the system's response to guidance commands in scenario A, the kinematic velocity relative to the NED frame is measured and compared to the setpoint, as shown in Figure 11. The tracking error illustrates how effectively the low-level multicopter flight controller manages the inputs. Additionally, sudden changes in velocity indicate higher accelerations and suggest aggressive avoidance behavior, while consistent velocity time series point to smoother navigation and better control. Figure 11 presents these values for the x500 UAV navigating through the obstacle course at 4.0 m/s. For most of the time, PX4 accurately tracks the setpoint. However, a significant error occurs at 43 seconds, attributed to the aggressive avoidance maneuver at $[83, 68]$ in Figure 9. At 4.0 m/s, the x500 approaches a wall-shaped obstacle and turns just before crashing. This suggests that the environment mapping function did not provide sufficient information for the PGFlow algorithm to initiate an earlier avoidance maneuver. Several smaller deviations are observed where the velocity setpoints oscillate at a high frequency. For instance, between 5 and 14 seconds, PGFlow publishes unstable commands in the y-direction of the NED system. These oscillations align with the frequency of gamma updates, indicating that the stepwise mapping of the environment causes this behavior. New obstacle information alters the gamma values between adjacent velocity vector setpoint calculations, leading to abrupt changes in PGFlow's output. Fortunately, the UAV's dynamics filter out these oscillations, resulting in smooth flight behavior.

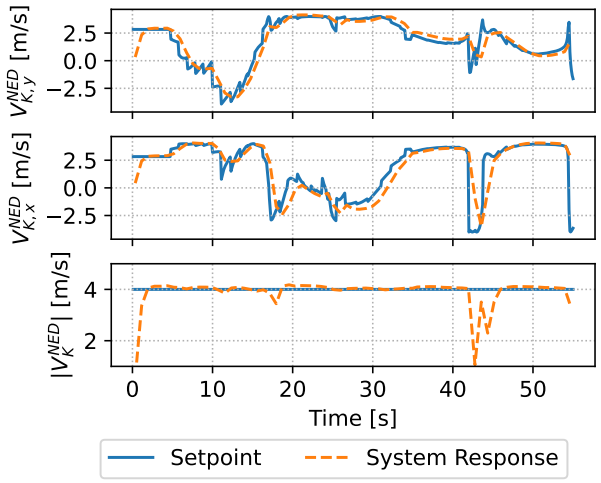


Fig. 11: Velocity setpoint and system response for the challenging scenario at 4.0 m/s are depicted. Throughout most of the simulation, PX4 accurately tracks the desired velocity vector setpoint. However, instances with larger tracking errors are observed during "close calls," where PGFlow generates infeasible commands to avoid obstacles just before a potential crash.

Comparison with Offline Simulation The final analysis of the single-vehicle simulation in scenario A compares the flight path generated during Gazebo SITL with the results of the offline simulations. The difference between the two approaches is significant, as the offline simulation assumes perfect execution of the velocity vector setpoints, resulting in deterministic behavior. In contrast, SITL introduces noise and asynchronous interactions between ROS2 nodes, leading to non-deterministic behavior. Crucially, the PGFlow algorithm in the offline simulation has complete knowledge of the obstacle environment before navigation begins. This comparison aims to address two key questions: Does stepwise mapping and limited environmental knowledge impact the guidance? And how do dynamic constraints and tracking errors affect avoidance behavior? Both simulations demonstrate the PGFlow's ability to avoid obstacles in scenario A while traveling to the destination. However, notable differences in navigation strategies emerge: the offline simulation, with complete obstacle knowledge, avoids narrow passages between concave walls by taking a wide curve, steering clear of the large obstacles in the lower left corner of the map. This behavior confirms that with full awareness of the obstacle environment, PGFlow prefers broader maneuvers around concave walls, avoiding tight passages entirely. This conservative approach contrasts with the SITL results, where the UAV, operating with limited obstacle information at

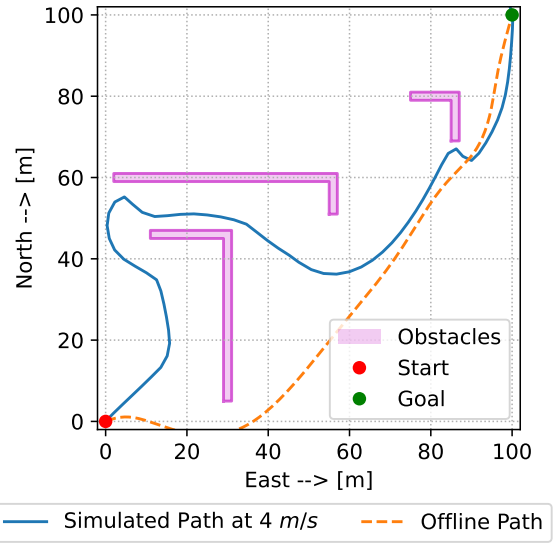


Fig. 12: Comparison between the path from SITL at 4.0 m/s and the path from the offline simulation in scenario A shows distinct differences. With complete knowledge of the obstacles, PGFlow in the offline simulation avoids navigating through narrow passages and instead flies a wide curve around the concave walls.

first, attempts more direct routes. This comparison highlights the adaptability of the PGFlow algorithm in SITL under partial information, effectively handling constraints and navigating through tighter spaces that the offline simulations avoid. These differences underscore the impact of environment mapping on navigation strategies, with the DAA system in SITL adopting a more aggressive yet effective obstacle avoidance approach. Despite the discrepancies, the DAA system generates feasible paths in this scenario, demonstrating its utility in environments with limited data.

Multi-Vehicle Simulation Now, the DAA system is tested with another simulated x500 multicopter UAV while navigating through the obstacles in scenario A. Although the necessary functions are implemented, the other aircraft is not controlled by PGFlow and flies eastward in a straight line. Its initial position is set to likely intersect with the flight path of the guided x500. The results of the simulation at 1.0 m/s are shown in Figure 13, with the left image displaying the global path and the right plot zooming in on the intersection point. The black UAV symbols indicate the positions of both vehicles at a critical moment.

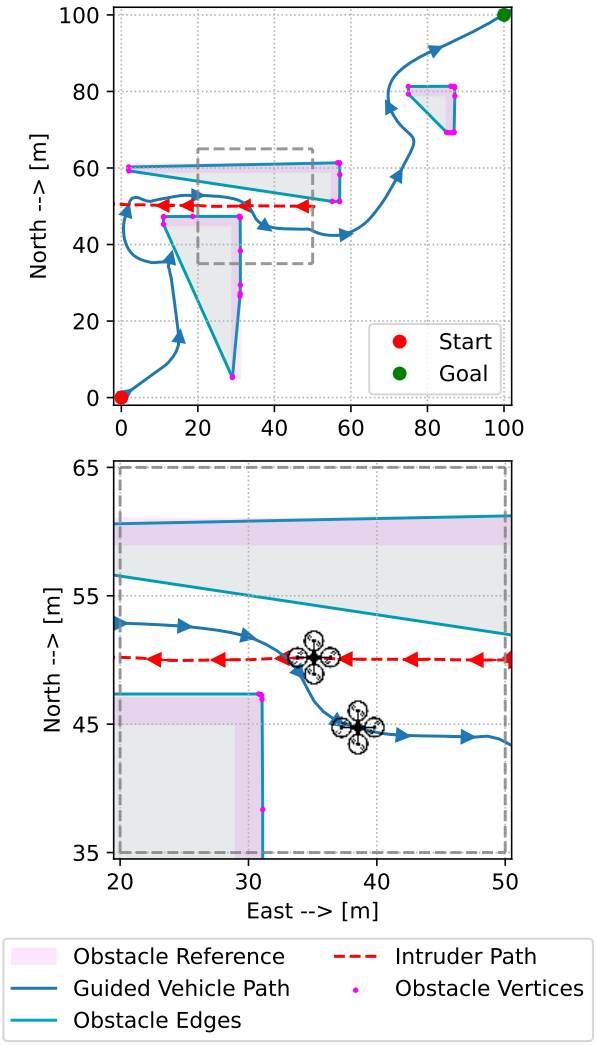


Fig. 13: Results of the multi-vehicle SITL simulation in scneario A at 1.0 m/s show that the DAA system avoids the other aircraft, even within the narrow passage between the concave walls. The system's behavior adheres to the right-of-way rule, confirming its ability to react to dynamic obstacles in a challenging environment.

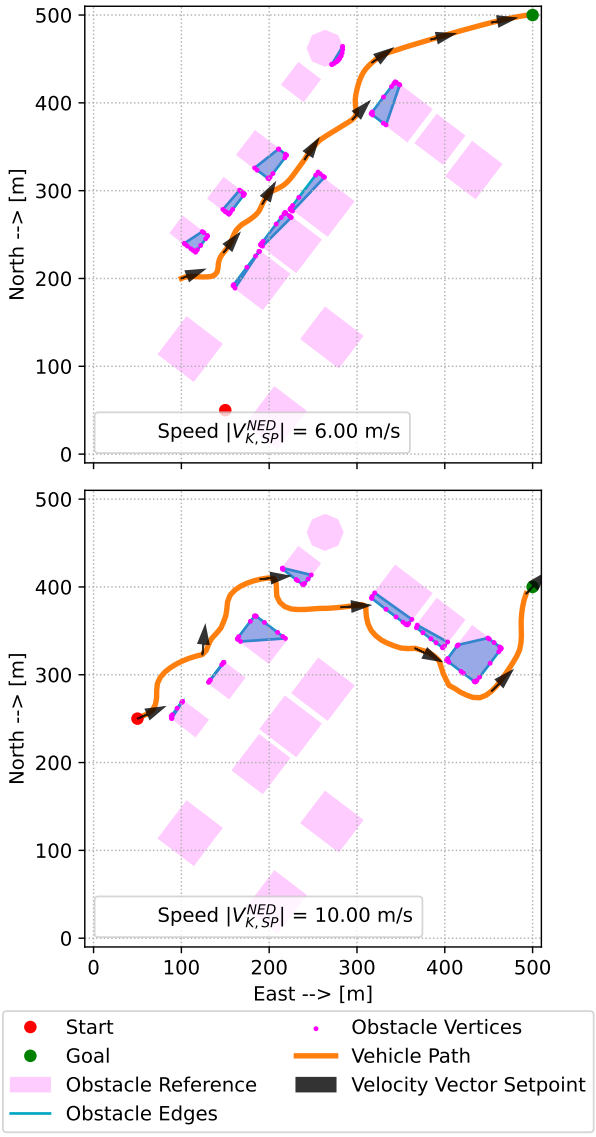


Fig. 14: SITL simulation results for scenario B at 6.0 m/s (upper) and 10.0 m/s (lower) demonstrate that, regardless of the start and end positions, the DAA system avoids all obstacles and guides the x500 to the goal. The environment mapping function converts the visible sections of skyscrapers into convex obstacles. Although these represent only small portions of the entire building layout, this information is sufficient for the PGFlow algorithm to generate feasible paths. With reference to the lower image, the high speed of 10.0 m/s does not negatively impact the avoidance behavior.

The results indicate that the DAA system successfully avoids the other aircraft in the narrowest passage of scenario A. The x500 yields right-of-way to the intruder by steering south. Globally, the path is similar to the previous simulation at 2.0 m/s. However, the environment mapping is more precise, with the concave obstacle in the upper right corner of the upper image being converted into a single larger triangular shape in 2D.

5.2 SIMULATION SCENARIO B

As outlined in Section 4, the Gazebo world of scenario B corresponds to a district of Dubai that features many skyscrapers in close vicinity. This simulation aims to

represent a real-world application of the DAA system, exposing PGFlow to a dense urban environment. Figure 14 presents the results of two simulations, each showing the flight path, several velocity vector setpoints, and the desired speed, similar to scenario A. Additionally, the plots illustrate the environment mapping at the end of the SITL run and the actual 2D shapes of the obstacles. To reduce computational load, the panel length is increased to 0.45 m, a necessary step to prevent the environment mapping and PGFlow from operating at a low frequency, as observed before. Furthermore, the LiDAR sensor's detection range is extended from 20.0 m to 40.0 m. However, successful results are also obtained with the shorter range.

The results confirm the DAA system's effectiveness in dense urban environments, as the x500 successfully avoids all obstacles and reaches the goal position. The larger scale of scenario B allows for higher velocity vector setpoints, which were not feasible in the previous obstacle course. Despite the increased speed, the avoidance behavior remains unimpaired. Unlike the smoother path in the upper image, the path of the x500 at 10.0 m/s features several sharp turns and is less smooth compared to the other simulation. Although the plots in Figure 14 might suggest small distances to the buildings, this is a false impression created by the large scale of the environment. Even at 10.0 m/s, the experiments do not involve close calls or crashes.

Figure 14 showcases the performance of the environment mapping module while providing PGFlow with the necessary obstacle information in scenario B. Unlike scenario A, the function not only maps the outer edges but also approximates the actual shape through convex hull calculation. In some cases, the proposed methodology captures the entire building based on incomplete LiDAR measurements, though most obstacles are represented by only small portions of the building layout. Despite this, the information is sufficient for PGFlow to generate feasible paths. The results from scenario B demonstrate the system's applicability to urban environments. For performance and tracking accuracy evaluations, the results of the 10.0 m/s simulation are analyzed in detail. In this run, the x500 takes 72.6 seconds to travel from start to finish, covering a distance of 763.2 meters.

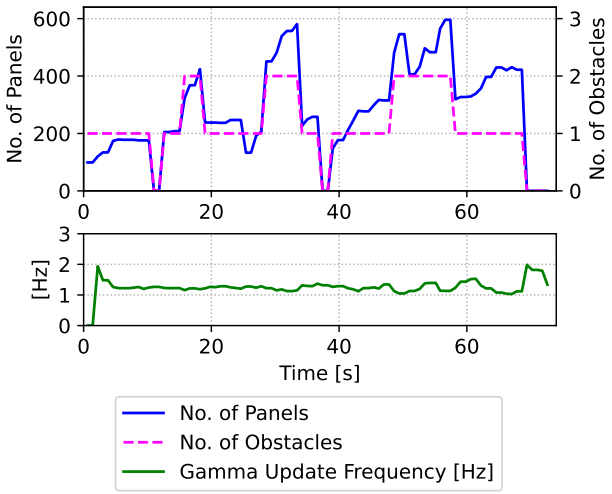


Fig. 15: Number of obstacles and panels considered for path planning in scenario B at 10.0 m/s. The pruning methodology proves effective, as no more than two of the seven identified obstacles are considered for path planning at the same time. The larger panel size reduces computational load and keeps the gamma update frequency from dropping below 1 Hz.

Performance Given the large scale of the obstacles in scenario B, it was anticipated that the required number of panels would surpass those in the previous analysis. However, Figure 15 reveals this concern to be unfounded, with the maximum number of panels never exceeding 600. Nonetheless, small drops in the gamma-update rate are directly linked to phases with a high number of panels. Additionally, the magenta line in Figure 15 demonstrates the effectiveness of the pruning methodology, as the number of obstacles influencing the PGFlow calculation never exceeds two. Without this strategy, the algorithm would consider seven obstacles and approximately 2100 panels for gamma and velocity vector setpoint calculations at the simulation's end. Longer missions would likely experience significant declines in computational efficiency and response time, potentially affecting the UAV's ability to navigate complex environments. The pruning strategy efficiently reduces computational load by concentrating on the most critical obstacles relevant to the local guidance process.

Vehicle Dynamics In scenario B, vehicle dynamics pose significant challenges for control precision and response adaptability at high velocities. Figure 16 shows the velocity setpoint and system response, illustrating how effectively the PX4 controller tracks

the guidance commands. This also allows for visual inspection of the consistency of PGFlow's velocity vector setpoints. Although the tracking error is larger than in other scenarios, PX4 generally follows the overall trend of the guidance commands. This is particularly noteworthy given the 10.0 m/s speed, which demands high responsiveness and stability from the system. Despite the PGFlow algorithm's disregard for vehicle dynamics, the tracking error remains within an acceptable range. The x500 maintains the desired speed with minimal deviations, highlighting the sophistication of the low-level flight controller. This performance suggests that, even without dynamic adjustments, the current setup can handle scenarios near the upper operational limits of the x500 multicopter. The velocity vector setpoints often exhibit fast, low-amplitude oscillations, likely due to continuous environment map updates used to calculate the guidance commands. The UAV cannot follow these oscillations precisely, but the vehicle dynamics naturally filter out these vibrations.

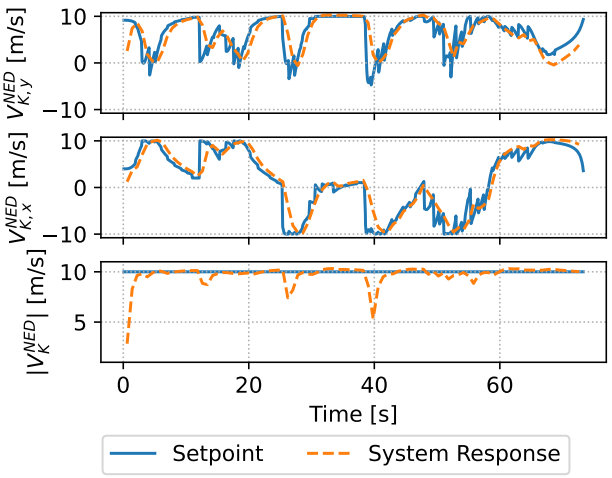


Fig. 16: Velocity setpoint and system response for scenario B at 10.0 m/s show a slightly higher tracking error compared to scenario A, but PX4 still allows the x500 to follow the overall trend of the guidance commands. Despite the high speed, the avoidance behavior is less aggressive than expected, even though PGFlow does not account for vehicle dynamics. With few exceptions, the UAV maintains the desired speed of 10.0 m/s.

Multi-Vehicle Simulation The DAA system successfully avoids the other x500 model in the narrow passage between buildings in scenario B. Including other aircraft in the guidance calculations does not hinder the avoidance of static obstacles. As demonstrated in previous experiments, the system adheres to the right-of-way rule. Although the UAVs appear close together in the lower image of Figure 17, they maintain a safe distance of at least 6.0 meters. These results align with observations from other multi-vehicle simulations, confirming the system's ability to react to dynamic obstacles in dense urban environments. The only difference from the experiments in scenario A is the increased desired speed. While the velocity vector setpoints were scaled to 1.0 m/s in the previous evaluation, the x500 in scenario B travels at 4.0 m/s. This higher speed does not challenge the guidance, as the avoidance behavior remains smooth near the intruder aircraft.

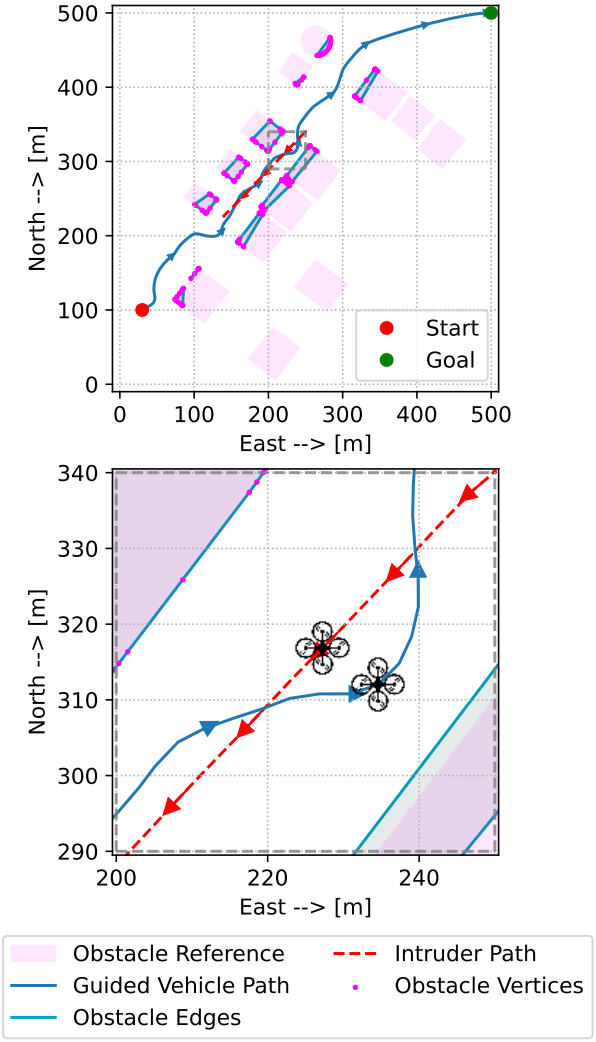


Fig. 17: Results of the multi-vehicle SITL simulation in scenario B at 4.0 m/s show that the DAA system successfully avoids the other aircraft despite the dense urban environment. The x500 yields the right-of-way to the intruder and executes an avoidance maneuver, resulting in a curved flight path. This experiment also confirms the DAA system's capability to avoid other aircraft at higher speeds.

6 CONCLUSIONS

This paper presents the development and evaluation of a DAA system for unmanned rotorcraft, which employs potential flow theory for the avoidance of static obstacles and other air traffic. The system is implemented as ROS2 framework and integrates the PGFlow algorithm developed by Bilgin et al. [19] for generating velocity vector setpoints that avoid any collision risk while traveling to a certain destination. As main contribution of this research, the algorithm is interfaced with a SLAM-based environment mapping approach which processes LiDAR measurements and ADS-B data. In this manner, the proposed framework extends the applicability of the PGFlow algorithm to previously unknown environments.

The results of the SITL simulations confirm the DAA system's general capability to avoid static and dynamic obstacles. Two different simulation scenarios are set up to test the system's performance under varying conditions. In scenario A, the system successfully navigates through narrow passages and concave obstacles, maintaining safe distances at speeds of up to 4.0 m/s. Scenario B, which is based on Dubai's marina district features larger-scale buildings and validates the system's applicability to real-world environments, achieving safe navigation even at higher speeds of up to 10.0 m/s.

The mapping approach with the RTAB-map at its core creates an accurate mapping of the static environment in all experiments. Although all subsequent processing and guidance computations are implemented with the ROS2 client library for Python, it runs fast enough to avoid any collision in the SITL simulations. The combination of density-based clustering and convex hull calculation enables to approximate the original shape of obstacles based on incomplete LiDAR scans. Further, it prevents the UAV from entering concave structures and shields the guidance computation from the irregular flow approximation as they occur in corners of such shapes in 2D. Approximating these flow patterns accurately enough to use the resulting velocity vector for guidance would require a fine discretization which in turn would increase the computational demand, limiting the real-time performance. In this context, the SITL simulations further enable the assessment of the PGFlow algorithm's performance when provided with a continuously expanding environment mapping. In general, it handles the uncertainty, but the resulting flight path from SITL differs from the idealized offline simulation.

Challenges encountered during this research include the system's sensitivity to the operational environment. Depending on the scale of the static obstacles and the density of the LiDAR scans, the computational load can vary significantly, affecting real-time performance. To address these challenges, future research should focus on optimizing the environment mapping function and the calculation of vortex strengths for real-time operation. Implementing the GammaCalculator node in C++ instead of Python could improve system latency.

Future work should also explore adaptive parameter settings based on the operational environment. For instance, dynamically adjusting the panel length based on detected obstacle characteristics could optimize computational resources and improve guidance accuracy. Additionally, incorporating course and velocity information from ADS-B broadcasts could enhance the system's traffic awareness, allowing it to predict and react to the future positions of other aircraft more effectively, especially in high-speed scenarios.

7 REFERENCES

[1]Voisin, M., "Erstes Mal bundesweit: Drohnenflotte startet Lieferdienst," *VDI Verlag GmbH*, 2/21/2024.

[2]Jo, D. and Kwon, Y., "Development of Rescue Material Transport UAV (Unmanned Aerial Vehicle)," *World Journal of Engineering and Technology*, Vol. 05, No. 04, 2017, pp. 720–729.

[3]Atif, M., Ahmad, R., Ahmad, W., Zhao, L., and Rodrigues, J. J. P. C., "UAV-Assisted Wireless Localization for Search and Rescue," *IEEE Systems Journal*, Vol. 15, No. 3, 2021, pp. 3261–3272.

[4]Primicerio, J., Di Gennaro, S. F., Fiorillo, E., Genesio, L., Lugato, E., Matese, A., and Vaccari, F. P., "A flexible unmanned aerial vehicle for precision agriculture," *Precision Agriculture*, Vol. 13, No. 4, 2012, pp. 517–523.

[5]Gheisari, M., Irizarry, J., and Walker, B. N., "UAS4SAFETY: The Potential of Unmanned Aerial Systems for Construction Safety Applications," *Construction Research Congress 2014*, edited by D. Castro-Lacouture, J. Irizarry, and B. Ashuri, American Society of Civil Engineers, Reston, VA, 05132014, pp. 1801–1810.

[6]Larrauri, J. I., Sorrosal, G., and Gonzalez, M., “Automatic system for overhead power line inspection using an Unmanned Aerial Vehicle — RELIFO project,” *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2013, pp. 244–252.

[7]Immerzeel, W. W., Kraaijenbrink, P., Shea, J. M., Shrestha, A. B., Pellicciotti, F., Bierkens, M., and de Jong, S. M., “High-resolution monitoring of Himalayan glacier dynamics using unmanned aerial vehicles,” *Remote Sensing of Environment*, Vol. 150, 2014, pp. 93–103.

[8]Motlagh, N. H., Bagaa, M., and Taleb, T., “UAV-Based IoT Platform: A Crowd Surveillance Use Case,” *IEEE Communications Magazine*, Vol. 55, No. 2, 2017, pp. 128–134.

[9]Nex, F., Armenakis, C., Cramer, M., Cucci, D. A., Gerke, M., Honkavaara, E., Kukko, A., Persello, C., and Skaloud, J., “UAV in the advent of the twenties: Where we stand and what is next,” *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 184, 2022, pp. 215–242.

[10]Budiyono, A. and Higashino, S.-I., “A Review of the Latest Innovations in UAV Technology,” *Journal of Instrumentation, Automation and Systems*, Vol. 10, No. 1, 2023, pp. 7–16.

[11]Bauranov, A. and Rakas, J., “Designing airspace for urban air mobility: A review of concepts and approaches,” *Progress in Aerospace Sciences*, Vol. 125, 2021, pp. 100726.

[12]Fan, B., Li, Y., Zhang, R., and Fu, Q., “Review on the Technological Development and Application of UAV Systems,” *Chinese Journal of Electronics*, Vol. 29, No. 2, 2020, pp. 199–207.

[13]ICAO, “Remotely Piloted Aircraft System (RPAS) Concept of Operations for International IFR Operations,” (2017, March), Accessed: 01.26.2024.

[14]EASA, “Easy Access Rules for Unmanned Aircraft Systems,” (2022, December), Accessed: 01.18.2024.

[15]Chand, B. N., Mahalakshmi, P., and Naidu, V. P. S., “Sense and avoid technology in unmanned aerial vehicles: A review,” *2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, IEEE, 15.12.2017 - 16.12.2017, pp. 512–517.

[16]Chandran, N. K., Sultan, M. T. H., Łukaszewicz, A., Shahar, F. S., Holovatyy, A., and Giernacki, W., “Review on Type of Sensors and Detection Method of Anti-Collision System of Unmanned Aerial Vehicle,” *Sensors (Basel, Switzerland)*, Vol. 23, No. 15, 2023.

[17]Tang, J., Lao, S., and Wan, Y., “Systematic Review of Collision-Avoidance Approaches for Unmanned Aerial Vehicles,” *IEEE Systems Journal*, Vol. 16, No. 3, 2022, pp. 4356–4367.

[18]Yasin, J. N., Mohamed, S. A. S., Haghbayan, M.-H., Heikkonen, J., Tenhunen, H., and Plosila, J., “Unmanned Aerial Vehicles (UAVs): Collision Avoidance Systems and Approaches,” *IEEE Access*, Vol. 8, 2020, pp. 105139–105155.

[19]Bilgin, Z., Yavrucuk, I., and Bronz, M., “Urban Air Mobility Guidance with Panel Method: Experimental Evaluation Under Wind Disturbances,” *Journal of Guidance, Control, and Dynamics*, Vol. 47, No. 6, 2024, pp. 1080–1096.

[20]Bilgin, Z., Bronz, M., and Yavrucuk, I., “Experimental Evaluation of Robustness of Panel-Method-Based Path Planning for Urban Air Mobility,” *AIAA AVIATION 2022 Forum*, American Institute of Aeronautics and Astronautics, Reston, Virginia, 06272022.

[21]Bilgin, Z., Bronz, M., and Yavrucuk, I., “Panel Method Based Guidance for Fixed Wing Micro Aerial Vehicles,” *International Micro Air Vehicle Conference*, Delft, Netherlands, 2022.

[22]Goss, J., Rajvanshi, R., and Subbarao, K., “Aircraft Conflict Detection and Resolution Using Mixed Geometric And Collision Cone Approaches,” *AIAA Guidance, Navigation, and Control Conference and Exhibit*, American Institute of Aeronautics and Astronautics, Reston, Virigina, 08162004.

[23]Bilgin, Z., Bronz, M., and Yavrucuk, I., “Experimental Evaluation of Robustness of Panel-Method-Based Path Planning for Urban Air Mobility,” *AIAA Aviation Forum*, 27 June-1 July 2022.

[24]Bilgin, Z., Bronz, M., and Yavrucuk, I., “Experimental Evaluation of Panel-Method-Based Path Planning for eVTOL in A Scaled Urban Environment,” *Vertical Flight Society Forum 78*, May 10-12 2022.

[25]Velagić, J., Vuković, L., and Ibrahimović, B., “Mobile Robot Motion Framework Based on

Enhanced Robust Panel Method," *International Journal of Control, Automation and Systems*, Vol. 18, No. 5, 2020, pp. 1264–1276.

[26]Riordan, J., Manduhu, M., Black, J., Dow, A., Dooly, G., and Matalonga, S., "LiDAR Simulation for Performance Evaluation of UAS Detect and Avoid," *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2021, pp. 1355–1363.

[27]Cruz, G. C. S. and Encarnação, P. M. M., "Obstacle Avoidance for Unmanned Aerial Vehicles," *Journal of Intelligent & Robotic Systems*, Vol. 65, No. 1-4, 2012, pp. 203–217.

[28]Meier, L., Honegger, D., and Pollefeys, M., "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 26.05.2015 - 30.05.2015, pp. 6235–6240.

[29]Macenski, S., Soragna, A., Carroll, M., and Ge, Z., "Impact of ROS 2 Node Composition in Robotic Systems," *IEEE Robotics and Autonomous Letters (RA-L)*, 2023.

[30]Macenski, S., Foote, T., Gerkey, B., Lalancette, C., and Woodall, W., "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics*, Vol. 7, No. 66, 2022.

[31]Fasano, G., Accado, D., Moccia, A., and Moroney, D., "Sense and avoid for unmanned aircraft systems," *IEEE Aerospace and Electronic Systems Magazine*, Vol. 31, No. 11, 2016, pp. 82–110.

[32]Lohani, B. and Ghosh, S., "Airborne LiDAR Technology: A Review of Data Collection and Processing Systems," *Proceedings of the National Academy of Sciences, India Section A: Physical Sciences*, Vol. 87, No. 4, 2017, pp. 567–579.

[33]Ravankar, A. A., Hoshino, Y., Ravankar, A., Jixin, L., Emaru, T., and Kobayashi, Y., "Algorithms and a Framework for Indoor Robot Mapping in a Noisy Environment Using Clustering in Spatial and Hough Domains," *International Journal of Advanced Robotic Systems*, Vol. 12, No. 3, 2015, pp. 27.

[34]Wang, K., Kooistra, L., Pan, R., Wang, W., and Valente, J., "UAV–based simultaneous localization and mapping in outdoor environments: A systematic scoping review," *Journal of Field Robotics*, 2024.

[35]Lin, Y.-C., Cheng, Y.-T., Zhou, T., Ravi, R., Hasheminasab, S., Flatt, J., Troy, C., and Habib, A., "Evaluation of UAV LiDAR for Mapping Coastal Environments," *Remote Sensing*, Vol. 11, No. 24, 2019, pp. 2893.

[36]Kim, P., Price, L. C., Park, J., and Cho, Y. K., "UAV-UGV Cooperative 3D Environmental Mapping," *Computing in Civil Engineering 2019*, edited by Y. K. Cho, F. Leite, A. Behzadan, and C. Wang, American Society of Civil Engineers, Reston, VA, 2019, pp. 384–392.

[37]Nüchter, A., *3D Robotic Mapping*, Vol. 52, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[38]Gupta, A. and Fernando, X., "Simultaneous Localization and Mapping (SLAM) and Data Fusion in Unmanned Aerial Vehicles: Recent Advances and Challenges," *Drones*, Vol. 6, No. 4, 2022, pp. 85.

[39]Labbé, M. and Michaud, F., "RTAB–Map as an open–source lidar and visual simultaneous localization and mapping library for large–scale and long–term online operation," *Journal of Field Robotics*, Vol. 36, No. 2, 2019, pp. 416–446.

[40]IntRoLab Universite de Sherbrooke, "RTAB-Map library and standalone application," (2024), Accessed: 03.06.2024.

[41]Shish, K. H., Cramer, N. B., Gorospe, G., Lombaerts, T., Stepanyan, V., and Kannan, K., "Survey of Capabilities and Gaps in External Perception Sensors for Autonomous Urban Air Mobility Applications," *AIAA Scitech 2021 Forum*, American Institute of Aeronautics and Astronautics, Reston, Virginia, 2021.

[42]Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al., "A density-based algorithm for discovering clusters in large spatial databases with noise," *kdd*, Vol. 96, 1996, pp. 226–231.

[43]scikit learn, "DBSCAN," (2024), Accessed: 06.04.2024.

[44]SciPy, "ConvexHull — SciPy v1.14.0 Manual," (2024), Accessed: 06.30.2024.

[45]Bilgin, Z., Bronz, M., and Yavrucuk, I., "Panel Method Based Guidance for Fixed Wing Micro Aerial Vehicles," *International Micro Air Vehicle Conference and Competition*, Sep 2022.