2022 International Conference on Unmanned Aircraft Systems (ICUAS)
Dubrovnik, Croatia. June 21-24, 2022

Plannie: A Benchmark Framework for Autonomous Robots Path Planning Algorithms Integrated to Simulated and Real Environments

Lidia Rocha[1] and Kelen Vivaldini[1]

2022 International Conference on Unmanned Aircraft Systems (ICUAS) | 978-1-6654-0593-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICUAS54217.2022.9836102

***Abstract*— Over the years, research has enabled robots to become more autonomous, determining their trajectories. Thus, path planning is an important area in autonomous robots. With the growing number of path planning algorithms, it is essential to have a framework capable of analyzing several algorithms in the same scenario to assess their capabilities. This paper presents Plannie, a framework to develop, simulate, benchmark, and test path planning algorithms in 2D and 3D environments in the real world. It supports many path planning algorithms, such as classic, metaheuristic, and machine learning. Furthermore, this framework offers several maps from an external database. However, it is also possible the build new maps in addition to having control, mapping, and localization techniques available for testing in a real environment. Plannie also offers planning modules that involve dynamic obstacle avoidance, coverage, traveling salesman problems, and multi-robot algorithms. Finally, we demonstrate the capability of the Plannie benchmarking several path planning algorithms in different maps. Plannie is open-source [1].**

I. Introduction

Path planning aims to find the shortest or optimal path between two points. Currently, several algorithms have been developed for validating the best technique for complex missions and can be divided into classic [1], meta heuristics [2], or machine learning [3] techniques.

One way to facilitate this analysis is to benchmark these techniques in the same scenario, evaluating significant metrics to carry out missions [4]. In this way, it is possible to determine which technique obtains the best results in each scenario and the best performance for each mission. Some benchmarking platforms offer this analysis, namely: Open Motion Planning Library (OMPL) [5], Move It [6], Online, Open-source, Programming System (OOPS$_{MP}$) [7], PathBench [8], and Bench-MR [9].

All of these platforms support approximate classical techniques. Only PathBench supports classic exact and machine learning techniques, and none of these platforms support meta heuristic techniques. OOPS, and PathBench has 2D and 3D maps to perform simulations in a shared environment. Only Move It and PathBench support communication in Robot Operating System (ROS) [10], but tests are performed only in a simulated environment.

This work describes a benchmark framework for path planning algorithms that can evaluate and compare different techniques in several environments. The analysis performed by the Plannie can be achieved with classical, meta heuristic, and machine learning algorithms on maps from external datasets or can be built within the framework. Furthermore, it can easily add new features and algorithms to the Plannie as it is modular and open-source. The main differential of this framework is to support natively:

- Open-source framework;
- Mapping, localization, control, and perception algorithms, to assist tests simulating missions in simulated and real environments, in 2D and 3D;
- Comparison between classical, meta heuristic, and machine learning path planning techniques;
- Other path planning categories such as Traveling Salesman Problem (TSP) and Coverage Path Planning (CPP);
- Use of multi-robots;
- Use of Unnamed Aerial Vehicles (UAVs);
- Dynamic obstacle avoidance and smoothed trajectories.

Plannie is a framework for autonomous robots, land, and aerial. However, to demonstrate the capability of Plannie the tests were made in a 3D environment to analyze the most complex issue. As a result, all trajectories returned by Plannie are optimized for aerial and ground robots. In addition, Plannie has available control techniques for land and air robots.

This paper is organized as follows. Section II, we describe the main features of the proposed framework. Section III, we show the path planning techniques implemented and the reason for the choice. In Section IV, we describe the tests environments. In Section V, we show the secondary usage modes offered by the Plannie framework. We conclude our work and propose some directions for future work in Section VII.

II. Proposed Framework

The tests were carried out in Python, simulators, and real environments through ROS Noetic.

An overview of the framework's architecture is shown in Figure 1 demonstrating the Plannie capacity to connect with a robot in Python, simulators, and real environments through ROS.

The planner's framework is described in Figure 2 and comprises path planning, obstacle avoidance, trajectory smoothing, control, mapping, perception, and decision-maker modules. Plannie's architecture is modular to facilitate other/new algorithms incrementally. Each module is described in this section.

The main objective of these modules is to help the user build the perfect algorithm to carry out a specific mission,

[1] The authors are with Federal University of São Carlos, Department of Computer, Brazil lidia@estudante.ufscar.br, vivaldini@ufscar.br

[1]https://github.com/lidiaxp/plannie

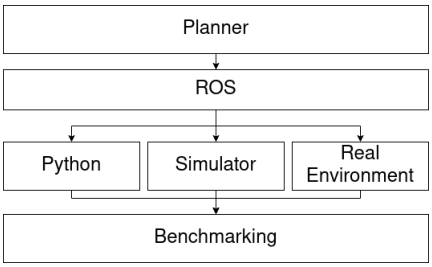


Fig. 1. Architecture High Overview

regardless of the environment. Because it, several modules can assist in different ways and have great results.

A. Static Path Planning

Path planning is essential for a robot to complete the mission to complete it optimally. Path planning is responsible for generating the best trajectory, without collisions, to navigate from the initial node to the objective.

Plannie has implemented classical techniques (exact and approximate), meta-heuristic techniques, and a machine learning technique to better deal with all missions.

Classical exact techniques are based on mathematical models and have high precision but have high computational cost [11]. The classical approximate techniques use sampling techniques to find the best trajectory, requiring less computational cost compared to the exact techniques [11]. Meta-heuristic techniques have a great capacity to deal with uncertainties and manage to generalize the method of arriving at a result, being generic and adaptable methods [12].

Machine learning techniques are divided into supervised, unsupervised, and reinforcement. The main techniques used for path planning are reinforcement learning [13], [14], [15] as they do not require a previous dataset for training, such as supervised techniques, and have greater precision than unsupervised techniques. In addition, it does not require accurate environmental models and can be used in various environments with little information [16].

B. Dynamic Path Planning

During a mission, there may be moving obstacles. Therefore, it is important that the planner has a dynamic path planning module to complete the task without collision and quickly avoid surprise obstacles.

According to the analysis of two sequential frames of the mapping module return Section II-E), these obstacles are labeled as dynamic or not. If the obstacle is inside the mapping sensor range and has changed its position is considered a dynamic obstacle. In this way, it is also possible to identify the vector direction. Thus, the dynamic path planning technique avoids this obstacle by considering fixing pre-known obstacles.

The Riemannian Motion Policies (RMP) [17] and Pedestrian Avoidance Method (PAM) [18] techniques were implemented in the Plannie framework. The RMP is a mathematical object for modular motion generation based on a second-order dynamical system. This algorithm creates a symmetric positive semi definitive matrix defined on the tangent space that measures the inner product between two tangent vectors to create the trajectory to avoid obstacles. PAM starts by tracking the obstacle and identifying which direction has more room to contour. When defining the direction, the trajectory nodes in the space in which the obstacle is moving are smooth.

C. Smooth Trajectory

The ideal trajectory to be followed by a mobile robot must obey its dynamic restrictions. That is, the curves performed must be as smooth as possible [19]. Bezier [20] or B-Spline [21] curves can do path smoothing, and Plannie has implemented both smoothing modes to carry out most missions.

B-Spline curves generalize Bezier curves and consistently smooth the trajectory in the same way. Still, the degree of smoothing can change according to the environment with Bezier curves, aiming to increase the smoothing without collision.

D. Control

For the robot to move between waypoints correctly and stably, it is necessary to use a control method. Plannie supports four different controls: the Model Predictive Control (MPC) [22], SE3 Controller (SE3) [23], Proportional Integral Derivative Controller (PID) [24], and Robust and Recursive Linear Quadratic Regulator (R-LQR) [25].

The MPC allows the current timeslot to be optimized while keeping future timeslots into account and can change future events and take control actions accordingly. SE3 follows the prescribed trajectory of the localization of the center of the mass and the desired direction of the first body-fixed axis. The R-LQR is based on a regulator of penalty parameters and regularized least squares, providing an algorithm independent of tuning parameters in online applications. PID is a control technique that causes the error signal to be minimized by the proportional action, zeroed by the integral action, and obtained with an anticipatory speed by the derivative action. Plannie offers this technique to land and aerial robots.

E. Mapping

Mapping is essential for the robot to be aware of obstacles and to deviate. Therefore, Plannie has implemented several mapping methods. The point cloud conversion to coordinates x, y, and z of the Velodyne sensor were implemented using laser sensors. Plannie also implemented an algorithm to convert the distance returned by Rplidar and Hokuyo sensors into 2D maps. The decision-maker also supports the use of Rplidar in conjunction with the package obstacle_detector [2] which detects and track obstacles from data provided by 2D laser scanners. In addition, the framework supports algorithms that return an occupancy grid, as $gmapping$ [26] and Hector Slam [27], that also produces point cloud.

[2]https://github.com/tysik/obstacle_detector

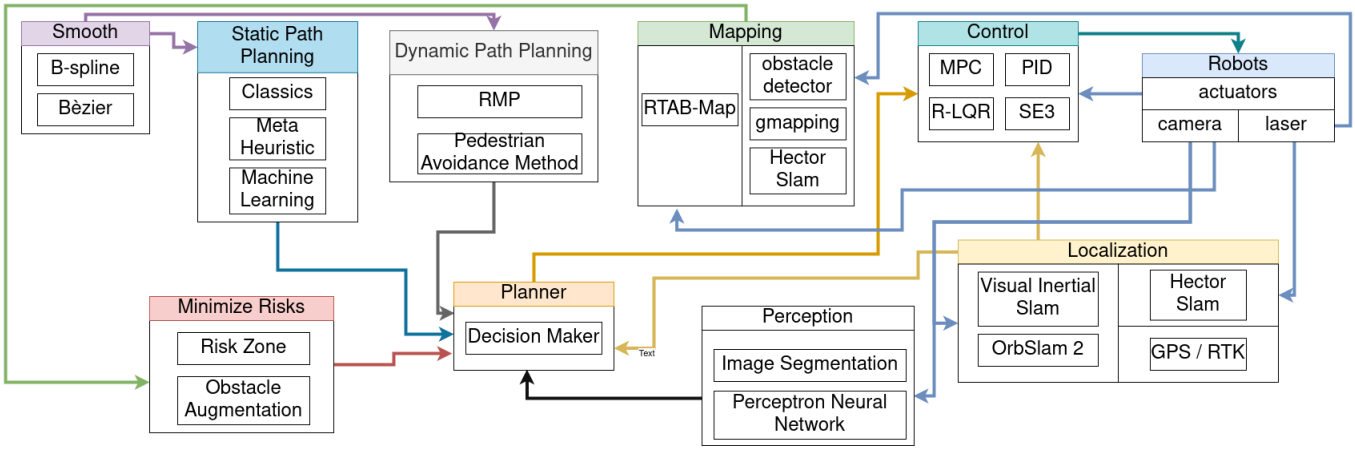


Fig. 2. Planner Architecture

Plannie also offers support to camera-based mapping and supports Real-Time Appearance-Based Mapping (RTAB-Map) [28] algorithm with the Zed camera and OrbSlam 2 [29] with a monocular camera. However, direct point cloud conversion from Zed's camera and OrbSlam 2 did not get good results, even when using the statistical filter of the Point Cloud Library (PCL) [30]. Therefore, according to Silva et al. [31], *gmapping* is the laser-based technique with the best performance, and RTAB-Map is the camera-based mapping technique with the best results.

In addition to these methods, the mapping module supports risk zones around obstacles. These zones are areas around the obstacle that the robot must avoid to minimize the risk of collision. According to [32], the risk of a collision caused by environmental uncertainties or performance constraints are minimized. The implemented risk zone can have different sizes to improve the flight.

F. Localization

In conjunction with mapping, localization is essential for the robot to orient itself and not collide. Therefore, Plannie has implemented algorithms based on laser and cameras to make the localization. The laser-based algorithms are: Hector Slam and the camera-based algorithms are: Visual Inertial Odometry (VIO) [33] and OrbSlam 2 (for a monocular camera). If the robot has these sensors, it is also possible to obtain its location via Global Position System (GPS) or Global Position System Real-Time Kinematic (GPS RTK).

Suppose where the robot will perform the mission has satellite coverage in an outdoor environment. Then, the best method is the RTK GPS against the standard GPS, which has an accuracy of up to 15 mm and 10 cm, respectively. Otherwise, among the methods natively implemented in the Plannie framework, the one with the best performance is Hector Slam, due to its greater accuracy, as shown in [34].

G. Decision-Maker

All modules are in the decision-maker to define the best action the robot should take. Initially, the decision-maker updates the robot's position on the map. Then, the static path planning algorithm is used to define the initial trajectory.

The trajectory built is discrete, so the planner always sends new waypoints to the control. Nevertheless, it is checked for collision between the UAV and a static obstacle before communicating with the control. The trajectory is recalculated based on a static path planning algorithm. Otherwise, it is checked if there is a collision with a dynamic obstacle. If there is, the trajectory is recalculated according to the modules present in dynamic path planning. Otherwise, the robot moves, the current position, and the environment map are updated.

This process continues until reaching the goal. Then, all these actions are carried out according to the modules described above.

H. Perception

In some missions, it is necessary to identify a target in the environment and decide based on this information. For example, in a forest fire, UAV can be used to identify the focus in a determined area using a thermal camera. Then, the robot will be induced to navigate until it covers the whole area.

Computer vision is responsible for identifying the objects in the scene. Plannie supports Image Segmentation [35] and the Perceptron Neural Network [36].

Image Segmentation separates the colors of the image, used in missions where it is necessary to identify an object with a different color from the environment [37]. The Perceptron Neural Network is used when a complex object needs to be found, and it is necessary to teach the neural network to identify it, such as landing bases for UAVs [38]. Plannie can read the serialized objects of these training to integrate with the planner.

III. Supported Path Planning Algorithms

Plannie has classic, meta-heuristic, machine learning algorithms natively. In addition to supporting coverage path

planning (CPP) and traveling salesman problem (TSP) algorithms. Modular architecture allows new algorithms to be added easily.

After calculating the trajectory, three optimizations improve the trajectories regardless of the technique used. This improvement does not necessarily make the trajectory shorter, but it makes it easier to follow according to the aerodynamic constraints of mobile robots. The optimizations are to decrease the number of trajectory curves and smooth them.

A. Classical

The classical algorithms implemented in the framework are divided into exact classical algorithms and approximate classical algorithms. The exact algorithms are A* [39] and the Artificial Potential Field (APF) [40]. The approximate algorithms are Rapid Random Tree (RRT) [41], Rapid Random Tree Connect (RRT-C) [42], and Probabilistic RoadMap (PRM) [43].

These techniques were chosen because they are the best classical techniques found in the literature to deal with unstructured environments [4], [44], [45]. In a 3D environment, only A* and RRT-C were implemented because, among the chosen techniques, they were the ones that presented the best results in the literature for unstructured environments [1].

For classic 2D techniques, there is a module to make the algorithms bidirectional. The path planning algorithm is the module's input, and it is responsible for managing the algorithm starting with the initial and final node until they meet in the middle of the path. Plannie can do the management of the algorithm between the initial and final node at each node, whenever the algorithm falls into a local minimum (the way to define when classical techniques enter into local minimum is shown in [1]), or the each n moves performed by each algorithm.

B. Meta Heuristic

Three meta-heuristic techniques were implemented, namely: Particle Swarm Optimization (PSO) [46], Gray Wolf Optimization (GWO) [47], and Glowworm Swarm Optimization (GSO) [48]. These were the techniques that obtained the best results in the state-of-the-art [12]. GWO has been shown to return the shortest paths in less time [2]. The GSO return trajectories in the second shortest time [49]. Besides, PSO shows are capable of returning a smoothed path within a second [50] and avoiding obstacles in unknown environments showing be reliable [51].

Only the PSO was implemented for 3D environments because, in a survey in the literature, it was the technique that returned trajectories with shorter path lengths and more reliability [12], [52].

C. Machine Learning

The framework has implemented only reinforcement learning techniques among the machine learning techniques. Reinforcement learning techniques have low computational cost [13], return an optimal path rapidly [53], and do not require accurate environmental and can be used with little information from the environment [16].

The technique chosen was Q-Learning [54], as this algorithm has a guaranteed convergence, learns policies in less time, and the state representation facilitates the process to generate a trajectory [55]. Furthermore, the Q-Learning algorithm learns online in unknown environments, adding new obstacles along the trajectory. Moreover, the Q-Learning train and test the trajectory in a known environment before the UAV moves.

IV. Tests Environments

Plannie can perform the tests in various environments through several platform tests. In this work, some platforms were used: Python simulation, Gazebo simulation, and in the real environment. In addition, these platforms support 2D and 3D environments. These environments are shown in Plannie's Github.

A. Python

Python simulations are best for developing new techniques or when it is necessary to validate time, smoothing, and trajectory length information statistically, as it is possible to simulate multiple flights quickly. For example, python can validate the path planning algorithm considering the control, mapping, and perfect result. In addition, it is possible to efficiently simulate the same algorithm several times in the same environment, aiming to generate statistical data to give reliability to the results. 2D simulations also allow the planner to test missions with dynamic obstacles.

B. Gazebo

The Gazebo simulator is a platform to simulate robots and their respective dynamics, making the results similar to a mission with a physical robot. Furthermore, performing tests on simulators allows validating the communication between planning, mapping, localization, and control algorithms, in realistic scenarios.

Plannie is compatible with Multi-Robot System (MRS) [56], Parrot-Sphinx [57], and Hector Quadrator simulator [58]. MRS is a system that simulates UAVs and multi UAVs with a control and estimation system. With the dynamics and aerodynamics, the UAVs supported are F-450, F-550, and T-650. Parrot-Sphinx is a simulation tool to simulate UAVs from Parrot company. The primary UAV is Parrot Bebop 2, which has all dynamics and aerodynamics implemented. The communication between the UAV and the ROS is done through the package *bebop_autonomy* [3]. The Hector Quadrator simulator has implemented the UAV dynamics, being possible to test the UAV performance through the simulator [59].

The MRS and Hector Quadrator simulator is compatible with melodic (Ubuntu 18) and noetic (Ubuntu 20) ROS. Thus, the integration between the UAVs of both simulators was implemented, making possible a swarm between them. On the other hand, Parrot-Sphinx is only compatible with

[3]https://bebop-autonomy.readthedocs.io/en/latest/

kinetic ROS (Ubuntu 16) due to its dependency on the *bebop_autonomy* package.

Besides these UAV tools for Gazebo, Plannie also supports any robot that moves from the */cmd_vel* topic in ROS. In this way, it is possible to use the PID control shown in Section II-D to control autonomous land and aerial robots.

C. Real Environment

The tests in a real environment analyze the robot's performance, considering all implemented algorithms in a real scenario and how it performs in different missions. From these tests, it is possible to send the robot, with the tested algorithms, to solve several tasks in a real environment, such as exploration [60], surveillance [61], search and rescue [62], among others.

In a real environment, Plannie was validated only with the UAV Parrot Bebop 2. ROS makes the communication and the modeling in the simulator consider the dynamics and aerodynamics of the UAVs. Therefore, real robots can use the same algorithms used in the simulator.

D. New Environments

Plannie has some test scenarios natively, such as unstructured indoor environments and forests. In addition, new maps can be imported or generated from the framework itself to diversify the test environments, with the map construction module supporting 2D and 3D environments.

The supported datasets to be imported are HouseExpo [63], which is a large dataset with 252,550 rooms; PathFinding database [64], [65], which have 3D scenarios from video games, like Warcraft III and Waframe, and 2D scenarios from geospatial Open Street Maps. The construction of new scenarios, through the framework, supports forests, structured and unstructured environments, build with box, water environments for monitoring, and 2D environments. In addition, a user interface is available to create 2D environments.

E. Metrics

Several metrics are chosen to evaluate and benchmark the performance of path planning algorithms. These metrics are based on the main challenges to optimize the path planning, ground and aerial [4].

The metrics are success rate - $S_r(\%)$, path length - $P_l(m)$, time to algorithm generate a trajectory - $Time(s)$, time to complete the mission - $T_{cm}(s)$, use of memory - $M(MB)$, use of CPU ($\%$), completeness - $C(\%)$, and battery - $B(\%)$. Time to complete the mission and battery are metrics that are only used in the Gazebo simulator and in a real environment if the UAV support it.

Using Plannie can also calculate the robustness obtained from the tests in simulated and real environments. Thus, it is possible to evaluate how the path planning algorithms can tolerate position-sensitive device, rotation, and acceleration errors [66].

The success rate evaluates how much time the robot moves between the start and goal node without collision. The path length is the size of the final trajectory, in meters. The time to algorithm generates a trajectory is the meantime to each algorithm re-plans the trajectory, in seconds. The time to complete the mission is the entire time it took for the robot to move between the start and goal node, in seconds. The use of memory and CPU are the computational usage of the computer. The completeness metric is the criterion used in path planning for finding the path if it exists.

The Gazebo battery plugin [4] is used to simulate the battery in the Gazebo simulator. This plugin can simulate charge and recharge a real battery of robots. The starter battery of the framework is based on F-450 that has a voltage of 14.8 V, the maximum current of 60000 mAH, the current required by the robot without the motors is 1000 mAH. The current needed with the motor is 75000 mAH.

The battery metric is proportional to the distance traveled by the UAV. However, it is essential to measure this metric separately because, in missions where the UAV needs to be stopped, the battery will decrease, but the path length will not increase. In addition, in missions such as search and rescue, the time the UAV acts is based on the battery. The UAV needs to return to base when the battery is running out. In this way, it is possible to add the battery metric to add this condition to the scheduler.

Plannie can obtain these metrics from simulations performed in Python, Gazebo, and a real environment, in known and unknown environments. These metrics make it possible to analyze each algorithm's complexity in time and space. In addition, when performing tests in different environments, it is possible to define which algorithm obtains better results. Moreover, carrying out flights in a simulated and real environment, it is possible to carry out a robustness analysis in the path planning techniques, validating how reliable they can be when using the real UAV. Finally, it is possible to analyze how each module affects the trajectory executed by the UAV by changing the techniques used. For example, how different will the trajectory be (considering the trajectory returned by Plannie in a Python known environment as ground truth) when using a more robust control technique but with an unstable location technique.

V. Extra Modes of Use

Plannie is configured to handle autonomous UAVs. However, as there are algorithms that work in 2D and the trajectories are generated to obey the movement limitations of most autonomous robots [67] it is possible to control autonomous land robots.

The main objective of the developed Plannie framework is to use path planning techniques for a mobile robot. For these tests, the architecture shown in Figure 2 is used. However, Plannie has three more modes of use: multi robots, coverage path planning, and traveling salesman problem (TSP).

These modules are not the focus of Plannie. The primary purpose of the extra modules is to help the user if he needs some of these functions in a specific mission. The extra

[4]https://github.com/robotizandoufsc/battery_simulator

modules have easy integration with Plannie, facilitating the development of an algorithm for a specific mission. For example, the main functions of Plannie do not perform multi-objective missions, but it is possible to easily add the TSP mode to the decision-maker to complete these missions.

A. Multi-Robots

For some missions, there is a significant improvement when using multiple robots. For example, to monitor risk areas, such as forests with fire [68] and polluted seas [69], or surveillance of a site [70]. By using multi-robots, tasks can be completed in less time and with more precision, reducing the damage to the environment.

In this framework, there is a module to control a robot swarm, as shown in Section IV-B. The architecture implemented in this algorithm is Master/Slave, which is a distributed architecture for multi-robot problems that have easy management of tasks allocated to each robot and the position of each robot during the mission [71].

B. Coverage Path Planning

The goal of coverage algorithms is to cover the largest area of interest in less time. In some missions, it is only necessary to investigate the areas that offer some risk to the environment [72]. There are also missions where the objective is to cover the entire area to map the area [73].

The Sweep Coverage [74] algorithm is implemented in the framework, which covers the entire informed area. The distance between the defined nodes is equivalent to the camera's ability to see. For example, if the camera can see an area of 10 meters, the coverage nodes will be separated by 10 meters without seeing repeated frames.

C. Travelling Salesman Problem

Travelling Salesman Problem algorithms aim to optimize routes concerning the path length, organizing them so that the next point to be followed is the closest to the current one. Therefore, it is an important technique to use in multi-objective missions. Examples of tasks are search and rescue [75], in the oil-gas industry [76], among others.

Plannie has the possibility of using TSP based on Genetic Algorithm (GA) [77], based on K-Nearest Neighbor (KNN) [78], and on a deterministic algorithm based on combination [79]. Plannie offers a classic technique, a meta heuristic, and a machine learning one as each performs better according to the mission. For example, the classical technique is faster with few nodes to optimize. Still, when the number of nodes increases too much, the classical technique increases the response time, and machine learning keeps constant. The same goes for the computational cost [80].

VI. Results

This Section presents some experiments in different scenarios using Plannie Framework. These tests were carried out with a notebook with an Intel Core i7 7700HQ computer with 16 GB of RAM and an Intel® HD Graphics 620.

All experiments use Plannie's main modules: path planning, smooth, minimize risks, and planner. The experiments carried out in the Gazebo and in a real environment also use the mapping, control, and location modules to assist the safe and reliable movement of the UAV in the environment and can be easily replaced by others for better results. The real environment tests also used the perception module to identify targets.

A. One Robot

Plannie demonstrates the benchmarking ability and the capacity to work in any Plannie environment. Several simulations were performed on the framework maps and external datasets, performing simulations in Python and Gazebo.

For the simulations, the algorithms with the best performance of each category of the path planning techniques in Plannie (Section III) were used.

Table I shows the average of the metrics of 10 Python simulations in indoor and outdoor environments belonging to Plannie. The indoor environment is known and is similar to a house with $21mx24mx6m$. The simulation in this environment is performed considering a 3D scene. In the outdoor environment, the simulation is done in 2D in an unknown forest setting with dimensions of $50mx50m$.

TABLE I
SIMULATIONS WITH INTERNAL MAPS IN PYTHON

	Alg	S_r	P_l	$Time$	M	CPU	C
Indoor	A*	100	34.79	1.28	8.83	80.39	100
	RRT-C	100	52.18	9.88	15.18	28.17	100
	PSO	20	33.03	36.84	9.81	80.89	100
	Q-Learning	100	45.48	31.06	20.73	16.67	100
Outdoor	A*	100	64.42	0.28	14.80	13.89	100
	RRT-C	100	67.79	0.27	13.83	12.50	100
	PSO	80	63.14	11.82	13.86	13.28	100
	Q-Learning	90	66.93	8.30	13.94	13.39	100

Table II shows the metrics of a flight performed in the Gazebo simulator in an unknown environment for Plannie's indoor and outdoor environments, presented in Table I.

Performing the normalization of the data in the previous tables (Table I and II), it is possible to compare them in radar charts aiming for better visualization of which technique has better results.

Figure 3 shows the data from the Python simulations. Figure 3 (a) shows that the CPU usage by Q-Learning and RRT-C is much lower than the other algorithms and can be used for missions that require onboard computing. The success rate of all techniques is good, except for the PSO, showing the algorithm's unreliability in indoor environments. Besides, the A* and RRT-C response time is one of the smallest, showing that the techniques can be well used to replay the trajectory. In Figure 3 (b) it is notable that all techniques have similar performance, except for time. The

TABLE II

SIMULATIONS WITH INTERNAL MAPS IN GAZEBO

	Alg	T_{cm}	P_l	M	CPU	B
Indoor	A*	63.94	31.25	3.57	89.6	2
	RRT-C	114.29	58.27	9.42	162	4
	PSO	160.67	29.11	12.81	291	5
	Q-Learning	129.36	43.63	10.26	305	2
Outdoor	A*	117.88	63.21	7.91	25	2
	RRT-C	131.21	63.97	6.34	28	2
	PSO	214.19	62.83	6.72	198	3
	Q-Learning	148.28	66.15	7.02	87	3

time metrics of RRT-C and A* are much smaller than the others.

Figure 4 shows the data from the simulations in the Gazebo. In Figure 4 (a), we can quickly assess that A* was the technique that obtained the best performance, obtaining the lowest values in all metrics. It is also noticeable that PSO is a costly algorithm, but it returned one of the shortest path lengths.

In Figure 4 (b), we can see that most metrics keep the same pattern, but, notably, the PSO is the most expensive of all, having the highest values in all metrics. The CPU metric has the same behavior shown in Figure 3 (a), in which RRT-C and Q-Learning had the lowest cost.

Table III presents the average metrics of 10 Python simulations of external maps, shown in Section IV-D. The environment of the House Expo dataset was unknown, 2D, and sized $40mx70m$. The video game environment is a 3D map of the Warframe game with dimensions of $120mx20mx25m$.

TABLE III

SIMULATIONS WITH EXTERNAL MAPS IN PYTHON

	Alg	S_r	P_l	$Time$	M	CPU	C
House Expo Map	A*	100	74.10	3.06	15.61	45.39	100
	RRT-C	100	78.19	3.72	15.13	30.07	100
	PSO	10	72.92	70.29	13.07	57.78	100
	Q-Learning	100	77.24	10.13	14.76	56.07	100
Video Game Map	A*	100	141.75	152.81	10.38	31.39	100
	RRT-C	100	136.03	11.13	15.57	41.00	100
	PSO	100	134.30	10.56	9.33	46.28	100
	Q-Learning	100	135.42	66.22	15.40	69.00	100

As can be seem Table III, the time in each environment has an average, but some environments have algorithms that take much longer than others, as is the example of the PSO in the House Expo maps, where we noticed that the PSO's time is seven times longer than the others. There was a behavior similar to this in Plannie's indoor maps, proving the PSO's difficulty finding a trajectory when the environment is very closed. However, this is not the case with Warframe maps, where PSO took the shortest time, but A* and Q-Learning took a very long. In this case, we must remember that the Warframe environment is three times larger than that of the House Expo, and both A* and Q-Learning have complexity directly related to the size of the environment.

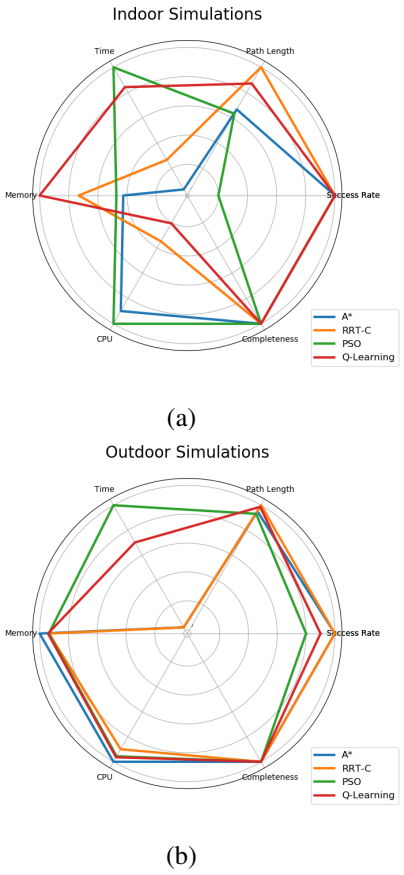
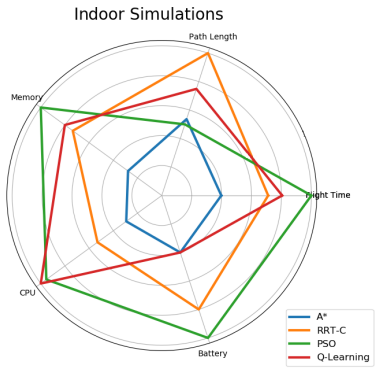


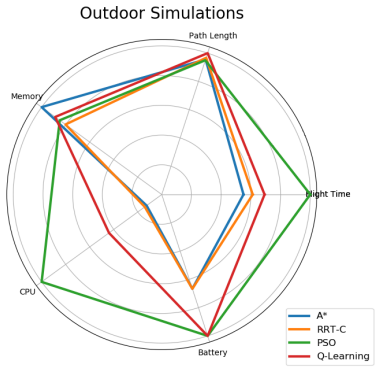
Fig. 3. Python Simulation Metrics. (a) Indoor environment. (b) Outdoor environment.

Another important point that we can see with this analysis is that the computational cost (memory and CPU) had little difference between the environments. The most constant was RRT-C and Q-Learning, which have their computational cost independent of the environment in which they are operating [13]. Besides that, PSO is not a reliable technique to be used in very closed environments. The technique obtained only a 10% success rate on the house expo maps.

Considering all these analyses, we can see that the PSO has a low success rate but returns the shortest trajectories, as Q-Learning. On the other hand, a* planning time increases exponentially as the size of the environment grows. RRT-C returns one of the most extended trajectories but fastest times. This conclusion, aiming to understand how each technique



(a)



(b)

Fig. 4. Gazebo Simulation Metrics. (a) Indoor environment. (b) Outdoor environment.

works, is possible due to the high amount of simulations in different environments.

B. Extra Mode of Use

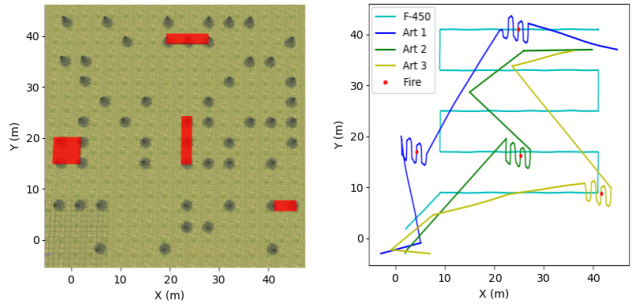
This Sub Section presents an application for the extra modes of use of Plannie. For example, figure 5 (a) shows a $50mx50m$ forest where the red squares represent fire in the trees.

For this mission, the monitoring of fire spots is carried out using multi UAVs. One of the UAVs will cover the entire area, search for fire focus, and send out other UAVs to explore the area closest to the fire. To manage trajectories when there are many fires, TSP is used to optimize the route order to UAVs visit them. Figure 5 shows the path taken by each of the UAVs to monitor the area.

The F-450 carried out the trajectory in 221.22 meters, and the Arts2 in 137.33, 109.87, and 124.98 meters, respectively. The time to cover the area and Arts2 investigates the fire was 327.36 seconds with eight percent battery consumption. The route of each Art2 was close to each other, showing the efficiency of the Plannie route manager, allowing to carry out the mission more efficiently.

C. Real World Tests

Plannie can perform flights in a simulated and real environment through ROS. Real robots can use the same algorithms with the validation done in the simulated environment due to the philosophy of software-in-loop. In the supplementary video and on GitHub, the UAV is demonstrated following a pre-planned route by this framework. In the video, there are no obstacles because the UAV used only has a monocular camera, so the mapping with this camera is not very accurate. The algorithm used for localization is with OrbSlam 2, with the monocular camera. The control used is the R-LQR. No perception technique was used because the UAV was not looking for anything, just validating the path planning, which was previously done by the exact classic technique, the artificial potential field.



(a) (b)

Fig. 5. Simulation with Algorithms Described in Extra Mode of Use. (a) A forest with fires (b) The trajectory carried out by UAV to coverage all the forest prioritizing the fires. The cyan trajectory is from the UAV (F-450) that performs coverage in the area. The blue, green, and yellow trajectories are from the auxiliary UAVs. The red circles are where there is a fire in the area.

VII. Conclusion

Plannie facilitates the development of specific algorithms for autonomous robot missions, improving mission performance by evaluating and using the best techniques for each mission. This framework presents a benchmarking between classical, meta heuristics, and machine learning, path planning techniques, enabling flights in a simulated and real environment through ROS. Furthermore, the framework has several integrated modules to facilitate the development of new algorithms and tests, such as control module, localization, mapping, perception, scenario construction, coverage, multi-robot support, among others.

Besides that, this work demonstrated the framework's performance through several simulations in shared environments created with external dataset environments and inside the framework analyzing various metrics, which can be easily visualized and quickly draw conclusions about the performance of the algorithms.

In future works, we intend to extend the Plannie framework to train data for machine learning algorithms and offer natively more reinforcement learning techniques. Besides, it implements Bayesian Optimization to give more competent coverage and allow simulations to be performed with dynamic obstacles in 3D in python. In addition, the number of tests will be increased to carry out tests in real and outdoor environments with aerial and ground robots,

to evaluate unknown wind, limited perception, and unstable communication.

ACKNOWLEDGMENT

The authors would like to acknowledge CAPES for financial support. We would also like to thank the partnership with the Flying U2 team, the LASI and LabRoM (EESC - USP) research labs, and LARIS - UFSCar.

REFERENCES

[1] L. Rocha, M. Aniceto, I. Araújo, and K. Vivaldini, "A uav global planner to improve path planning in unstructured environments," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2021, pp. 688–697.

[2] R. K. Dewangan, A. Shukla, and W. W. Godfrey, "Three dimensional path planning using grey wolf optimizer for uavs," *Applied Intelligence*, vol. 49, no. 6, pp. 2201–2217, 2019.

[3] F. Zeng, C. Wang, and S. S. Ge, "A survey on visual navigation for artificial agents with deep reinforcement learning," *IEEE Access*, vol. 8, pp. 135 426–135 442, 2020.

[4] S. Aggarwal and N. Kumar, "Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges," *Computer Communications*, 2019.

[5] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[6] M. Moll, I. A. Sucan, and L. E. Kavraki, "Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization," *IEEE Robotics Automation Magazine*, vol. 22, no. 3, pp. 96–102, 2015.

[7] E. Plaku, K. E. Bekris, and L. E. Kavraki, "Oops for motion planning: An online, open-source, programming system," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 3711–3716.

[8] A. Toma, H. Hsueh, H. A. Jaafar, R. Murai, P. H. J. Kelly, and S. Saeedi, "Pathbench: A benchmarking platform for classical and learned path planning algorithms," *CoRR*, vol. abs/2105.01777, 2021. [Online]. Available: https://arxiv.org/abs/2105.01777

[9] E. Heiden, L. Palmieri, L. Bruns, K. O. Arras, G. S. Sukhatme, and S. Koenig, "Bench-mr: A motion planning benchmark for wheeled mobile robots," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4536–4543, 2021.

[10] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[11] L. Rocha, M. Aniceto, I. Araújo, and K. Vivaldini, "A uav global planner to improve path planning in unstructured environments," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2021, pp. 688–697.

[12] L. Rocha and K. Vivaldini, "Comparison between meta-heuristic algorithms for path planning," in *Anais do VIII Workshop de Teses e Dissertações em Robótica/Concurso de Teses e Dissertações em Robótica*. SBC, 2020, pp. 1–10.

[13] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," *arXiv preprint arXiv: 2103.08624*, 2021.

[14] S. Kulkarni, V. Chaphekar, M. M. U. Chowdhury, F. Erden, and I. Guvenc, "Uav aided search and rescue operation using reinforcement learning," *arXiv preprint arXiv:2002.08415*, 2020.

[15] S. Sudhakar, V. Vijayakumar, C. S. Kumar, V. Priya, L. Ravi, and V. Subramaniyaswamy, "Unmanned aerial vehicle (uav) based forest fire detection and monitoring for reducing false alarms in forest-fires," *Computer Communications*, vol. 149, pp. 1–16, 2020.

[16] H. Chen, Y. Ji, and L. Niu, "Reinforcement learning path planning algorithm based on obstacle area expansion strategy," *Intelligent Service Robotics*, pp. 1–9, 2020.

[17] N. D. Ratliff, J. Issac, and D. Kappler, "Riemannian motion policies," *CoRR*, vol. abs/1801.02854, 2018. [Online]. Available: http://arxiv.org/abs/1801.02854

[18] Y. Hiroi and A. Ito, "A pedestrian avoidance method considering personal space for a guide robot," *Robotics*, vol. 8, no. 4, 2019. [Online]. Available: https://www.mdpi.com/2218-6581/8/4/97

[19] P. Pandey, A. Shukla, and R. Tiwari, "Three-dimensional path planning for unmanned aerial vehicles using glowworm swarm optimization algorithm," *International Journal of System Assurance Engineering and Management*, vol. 9, no. 4, pp. 836–852, 2018.

[20] H. A. Satai, M. M. A. Zahra, Z. I. Rasool, R. S. Abd-Ali, and C. I. Pruncu, "Bézier curves-based optimal trajectory design for multirotor uavs with any-angle pathfinding algorithms," *Sensors*, vol. 21, no. 7, p. 2460, 2021.

[21] F. Stoican, I. Prodan, D. Popescu, and L. Ichim, "Constrained trajectory generation for uav systems using a b-spline parametrization," in *2017 25th Mediterranean Conference on Control and Automation (MED)*. IEEE, 2017, pp. 613–618.

[22] A. Altan and R. Hacıoğlu, "Model predictive control of three-axis gimbal system mounted on uav for real-time target tracking under external disturbances," *Mechanical Systems and Signal Processing*, vol. 138, p. 106548, 2020.

[23] Y. Tadokoro, T. Ibuki, and M. Sampei, "Nonlinear model predictive control of a fully-actuated uav on se (3) using acceleration characteristics of the structure," in *2019 12th Asian Control Conference (ASCC)*. IEEE, 2019, pp. 283–288.

[24] A. Noordin, M. A. M. Basri, Z. Mohamed, and I. M. Lazim, "Adaptive pid controller using sliding mode control approaches for quadrotor uav attitude and position stabilization," *Arabian Journal for Science and Engineering*, vol. 46, no. 2, pp. 963–981, 2021.

[25] J. R. Benevides, R. S. Inoue, M. A. Paiva, and M. H. Terra, "Ros-based robust and recursive optimal control of commercial quadrotors," in *15th Int. Conf. on Automation Science and Engineering (CASE)*. IEEE, 2019, pp. 998–1003.

[26] M. Gilson, J. Gauthier, K. Garcia, C. Kienzle, J. A. Muse, and X. Zhang, "Fault-tolerant mapping and localization for quadrotor uavs," in *AIAA Scitech 2021 Forum*, 2021, p. 1812.

[27] M. Petrlík, T. Báča, D. Heřt, M. Vrba, T. Krajník, and M. Saska, "A robust uav system for operations in a constrained environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2169–2176, 2020.

[28] Á. Wolf, P. Troll, S. Romeder-Finger, A. Archenti, K. Széll, and P. Galambos, "A benchmark of popular indoor 3d reconstruction technologies: Comparison of arcore and rtab-map," *Electronics*, vol. 9, no. 12, p. 2091, 2020.

[29] A. Yusefi, A. Durdu, and C. Sungur, "Orb-slam-based 2d reconstruction of environment for indoor autonomous navigation of uavs," *Avrupa Bilim ve Teknoloji Dergisi*, pp. 466–472, 2020.

[30] S. Wei, D. Niu, Q. Li, X. Chen, and J. Liu, "A 3d vehicle recognition system based on point cloud library," in *2019 Chinese Control Conference (CCC)*. IEEE, 2019, pp. 7023–7027.

[31] B. M. F. da Silva, R. S. Xavier, T. P. do Nascimento, and L. M. Goncalves, "Experimental evaluation of ros compatible slam algorithms for rgb-d sensors," in *2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR)*, 2017, pp. 1–6.

[32] S. Sankararaman and K. Goebel, "Computational architecture for autonomous decision-making in unmanned aerial vehicles," in *Micro- and Nanotechnology Sensors, Systems, and Applications X*, vol. 10639. International Society for Optics and Photonics, 2018, p. 106391Y.

[33] E. M. Lee, I. Wee, T. Kim, and D. H. Shim, "Comparison of visual inertial odometry using flightgoggles simulator for uav," in *2019 19th International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2019, pp. 1166–1169.

[34] M. Filipenko and I. Afanasyev, "Comparison of various slam systems for mobile robot in an indoor environment," 09 2018.

[35] M. K. Vasić, A. Drak, N. Bugarin, S. Kružić, J. Musić, C. Pomrehn, M. Schöbel, M. Johenneken, I. Stančić, V. Papić, *et al.*, "Deep semantic image segmentation for uav-ugv cooperative path planning: A car park use case," in *2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. IEEE, 2020, pp. 1–6.

[36] V. Annepu and A. Rajesh, "An unmanned aerial vehicle-aided node localization using an efficient multilayer perceptron neural network in wireless sensor networks," *Neural Computing and Applications*, vol. 32, no. 15, pp. 11 651–11 663, 2020.

[37] A. A. Gebrehiwot and L. Hashemi-Beni, "Three-dimensional inundation mapping using uav image segmentation and digital surface model," *ISPRS International Journal of Geo-Information*, vol. 10, no. 3, p. 144, 2021.

[38] J. P. C. de Souza, A. L. M. Marcato, E. P. de Aguiar, M. A. Jucá, and A. M. Teixeira, "Autonomous landing of uav based on artificial neural

network supervised by fuzzy logic," *Journal of Control, Automation and Electrical Systems*, vol. 30, no. 4, pp. 522–531, 2019.
[39] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
[40] C. W. Warren, "Global path planning using artificial potential fields," in *1989 IEEE International Conference on Robotics and Automation*. IEEE Computer Society, 1989, pp. 316–317.
[41] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.
[42] M. Jordan and A. Perez, "Optimal bidirectional rapidly-exploring random trees," 2013.
[43] P. O. Pettersson and P. Doherty, "Probabilistic roadmap based path planning for an autonomous unmanned aerial vehicle," *Sensors*, vol. 200, p. 66Hz, 2004.
[44] B. Patle, G. Babu L, A. Pandey, D. Parhi, and A. Jagadeesh, "A review: On path planning strategies for navigation of mobile robot," *Defence Technology*, vol. 15, no. 4, pp. 582–606, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214914718305130
[45] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Path planning and trajectory planning algorithms: A general overview," *Mechanisms and Machine Science*, vol. 29, pp. 3–27, 03 2015.
[46] B. Sun, W.-d. Chen, and Y.-g. Xi, "Particle swarm optimization based global path planning for mobile robots," *Control and Decision*, vol. 20, no. 9, p. 1052, 2005.
[47] S. Zhang, Y. Zhou, Z. Li, and W. Pan, "Grey wolf optimizer for unmanned combat aerial vehicle path planning," *Advances in Engineering Software*, vol. 99, pp. 121–136, 2016.
[48] T. DU Peng-zhen, J.-f. LU, and Y. SUN, "Global path planning for alvbased on improved glowworm swarm optimization under uncertain environment," *ACTA ELECTONICA SINICA*, vol. 42, no. 3, p. 616, 2014.
[49] K. N. Kaipa and D. Ghose, *Glowworm swarm optimization: theory, algorithms, and applications*. Springer, 2017, vol. 698.
[50] W. Li, M. Tan, L. Wang, and Q. Wang, "A cubic spline method combing improved particle swarm optimization for robot path planning in dynamic uncertain environment," *International Journal of Advanced Robotic Systems*, vol. 17, no. 1, p. 1729881419891661, 2020.
[51] E. Krell, A. Sheta, A. P. R. Balasubramanian, and S. A. King, "Collision-free autonomous robot navigation in unknown environments utilizing pso for path planning," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 9, no. 4, pp. 267–282, oct 2019. [Online]. Available: https://www.sciendo.com/article/10.2478/jaiscr-2019-0008
[52] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889015300671
[53] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 285–292.
[54] C. Li, J. Zhang, and Y. Li, "Application of artificial neural network based on q-learning for mobile robot path planning," in *2006 IEEE International Conference on Information Acquisition*. IEEE, 2006, pp. 978–982.
[55] S. J. Waskow, "Reinforcement learning using tile coding in multi agent scenarios," 2010.
[56] T. Baca, M. Petrlik, M. Vrba, V. Spurny, R. Penicka, D. Hert, and M. Saska, "The mrs uav system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles," 2021.
[57] W. Giernacki, P. Kozierski, J. Michalski, M. Retinger, R. Madonski, and P. Campoy, "Bebop 2 quadrotor as a platform for research and education in robotics and control engineering," in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2020, pp. 1733–1741.
[58] M. Nakamura, K. Takaya, H. Ohta, K. Shibayama, and V. Kroumov, "Quadrotor modeling and simulation for industrial application," in *2019 23rd International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, 2019, pp. 37–42.
[59] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. von Stryk, "Comprehensive simulation of quadrotor uavs using ros and gazebo," in *3rd Int. Conf. on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR)*, 2012, p. to appear.
[60] B. Zhou, Y. Zhang, X. Chen, and S. Shen, "Fuel: Fast uav exploration using incremental frontier structure and hierarchical planning," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 779–786, 2021.
[61] D. H. Stolfi, M. R. Brust, G. Danoy, and P. Bouvry, "A competitive predator-prey approach to enhance surveillance by uav swarms," *Applied Soft Computing*, p. 107701, 2021.
[62] M. Atif, R. Ahmad, W. Ahmad, L. Zhao, and J. J. Rodrigues, "Uav-assisted wireless localization for search and rescue," *IEEE Systems Journal*, 2021.
[63] L. Tingguang, H. Danny, L. Chenming, Z. Delong, W. Chaoqun, and M. Q.-H. Meng, "Houseexpo: A large-scale 2d indoor layout dataset for learning-based algorithms on mobile robots," *arXiv preprint arXiv:1903.09845*, 2019.
[64] N. R. Sturtevant, "Benchmarks for grid-based pathfinding," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 2, pp. 144–148, 2012.
[65] D. Brewer and N. R. Sturtevant, "Benchmarks for pathfinding in 3d voxel space," in *Eleventh Annual Symposium on Combinatorial Search*, 2018.
[66] S. Aggarwal and N. Kumar, "Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges," *Computer Communications*, vol. 149, pp. 270–299, 2020.
[67] I. Negrean, C. Schonstein, K. Kacso, C. Negrean, and A. Duca, "Formulations about dynamics of mobile robots," in *Solid State Phenomena*, vol. 166. Trans Tech Publ, 2010, pp. 309–314.
[68] V. Sherstjuk, M. Zharikova, and I. Sokol, "Forest fire-fighting monitoring system based on uav team and remote sensing," in *2018 IEEE 38th International Conference on Electronics and Nanotechnology (ELNANO)*. IEEE, 2018, pp. 663–668.
[69] C. Liu, X. Zhou, Y. Zhou, and A. Akbar, "Multi-temporal monitoring of urban river water quality using uav-borne multi-spectral remote sensing." *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 43, 2020.
[70] J. Gu, T. Su, Q. Wang, X. Du, and M. Guizani, "Multiple moving targets surveillance based on a cooperative network for multi-uav," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 82–89, 2018.
[71] T. Lemaire, R. Alami, and S. Lacroix, "A distributed tasks allocation scheme in multi-uav context," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 4, 2004, pp. 3622–3627 Vol.4.
[72] J. R. Souza, C. C. Mendes, V. Guizilini, K. C. Vivaldini, A. Colturato, F. Ramos, and D. F. Wolf, "Automatic detection of ceratocystis wilt in eucalyptus crops from aerial images," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 3443–3448.
[73] A. Bouras, Y. Bouzid, and M. Guiatni, "Multi-uavs coverage path planning," in *Proceedings of the 4th International Conference on Electrical Engineering and Control Applications*, S. Bououden, M. Chadli, S. Ziani, and I. Zelinka, Eds. Singapore: Springer Singapore, 2021, pp. 23–36.
[74] W. Cheng, K. Liu, Y. Liu, X. Li, and X.-K. Liao, "Sweep coverage with mobile sensors," vol. 10, 05 2008, pp. 1 – 9.
[75] T. Liu, Q. Yuan, L. Wang, Y. Wang, and N. Zhang, "Multi-objective optimization for oil-gas production process based on compensation model of comprehensive energy consumption using improved evolutionary algorithm," *Energy Exploration & Exploitation*, vol. 39, no. 1, pp. 273–298, 2021.
[76] S. Hayat, E. Yanmaz, T. X. Brown, and C. Bettstetter, "Multi-objective uav path planning for search and rescue," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 5569–5574.
[77] G. Zames, N. Ajlouni, N. Ajlouni, N. Ajlouni, J. Holland, W. Hills, and D. Goldberg, "Genetic algorithms in search, optimization and machine learning," *Information Technology Journal*, vol. 3, no. 1, pp. 301–302, 1981.
[78] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-nearest neighbor algorithm," *IEEE transactions on systems, man, and cybernetics*, no. 4, pp. 580–585, 1985.
[79] O. Abdoun and J. Abouchabaka, "A comparative study of adaptive crossover operators for genetic algorithms to resolve the traveling salesman problem," *arXiv preprint arXiv:1203.3097*, 2012.
[80] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," *CoRR*, vol. abs/2103.08624, 2021. [Online]. Available: https://arxiv.org/abs/2103.08624