Smart Agricultural Technology 4 (2023) 100191

Contents lists available at ScienceDirect

Smart Agricultural Technology

journal homepage: www.journals.elsevier.com/smart-agricultural-technology

Harpia: A hybrid system for agricultural UAV missions

Veronica Vannini *, Gustavo de Moura Souza, Claudio Fabiano Motta Toledo

Institute of Mathematics and Computer Sciences, University of Sao Paulo, Brazil

ARTICLE INFO

Keywords:
Autonomous system
Mission planning
Path planning
Precision agriculture
Risk mitigation

ABSTRACT

The present paper introduces Harpia, a hybrid artificial intelligent planning system for UAVs. Harpia aims to execute tasks for agricultural applications with minimum human intervention. The mission's execution is on-board, running under the Robotic Operating System and executing re-planning for tasks and path planning with obstacle avoidance. The re-planning can happen after mission changes in real-time or unpredictable UAV behavior. It combines Planning Domain Definition Language for task planning, Bayesian Network to evaluate mission execution, and K-Nearest Neighbors algorithm to select a path planner. Thus, Harpia's novelty focuses on robustness for autonomously planning and re-planning the sequence of tasks and trajectories to regions of interest. The main contributions include an autonomous system architecture for planning missions with minimum human intervention, no boundary by specific tasks, and computationally simple for operating within non-convex scenarios. The computational tests report results for 21 simulated scenarios, where Harpia handled all situations properly, e.g., making decisions about task re-planning with 97.57% accuracy based on battery health and choosing the better path-planning for each case with at least 95% of accuracy.

1. Introduction

There are increasing demands for UAVs application in precision agriculture, but requirements about full autonomy are still challenging, see [1] and [2]. The level of independence is enhanced for UAV systems through decision-making capability embedded onboard to handle external (e.g., environment or mission changes) and internal (e.g., system failure or low battery) situations. The embedded systems reduce the dependence on ground control systems, with a human pilot in charge, to fully autonomous flight [3].

The present research introduces Harpia, an autonomous planning system for Unmanned Aerial Vehicles (UAVs). In the current development, Harpia's resources mainly operate in farms where there is a more static environment; however, even on a farm, the UAV executes many tasks and must avoid obstacles. Thus, we report Harpia's robustness for autonomously planning and re-planning the sequence of tasks and trajectories to regions of interest.

The main contributions brought by Harpia are (i) an autonomous system for planning and executing missions with minimum human intervention; (ii) a general system architecture that is no boundary to accomplish only a specific task within a farm, as usual in agricultural applications as reported in [1] and [2]; (iii) the system architecture presents a computational simplicity for practical proposes, but adds the convergence of methods such as Planning Domain Definition Language for task planning, Bayesian Network to evaluate mission execution, and K-Nearest Neighbors algorithm to select a path planner; (iv) the system autonomously operates within non-convex scenarios, which happens in farms and other real-world applications, making decisions about mission planning and re-planning in real-time; (v) we propose some adaptations over RRT and PFP path planning algorithms to deal with obstacle avoidance using ray casting and chance-constraints for risk allocation, respectively.

The paper is organized as follows: Section 2 describes the main related work and Section 3 states the problem approached. Harpia's architecture is explained in Section 4 and we report the experimental results in Section 5. The conclusion follows in Section 6.

2. Related works

The use of UAVs in precision agriculture has been advancing in recent years. For example, the recent work in [4] reports a wheat ear counting method employing UAV, where transfer learning from the ground-based counting model to the UAV improves the overall wheat ears counting. The works in [1] and [2] review researches about the use of UAV in precision agriculture. However, the survey in [1] points out the limitations of UAV systems for agriculture applications, where

* Corresponding author.
E-mail addresses: veronica.vannini@usp.br (V. Vannini), gustavo.moura.souza@usp.br (G. de Moura Souza), claudio@icmc.usp.br (C.F.M. Toledo).

https://doi.org/10.1016/j.atech.2023.100191
Received 12 December 2022; Received in revised form 22 January 2023; Accepted 22 January 2023
Available online 13 February 2023

the mission and path planning must optimize battery and flight time to accomplish more tasks in crop fields.

A total of 20 UAV applications is reported in [2] for aerial crop monitoring processes or spraying tasks, where the authors evaluate UAV system architectures among other features. The architectures are designed for specific purposes in monitoring and spraying tasks, leading the authors to suggest decision support systems as future work. There is a lack of applications in [1] and [2] that describes systems handling autonomous operation, decision-making, and risk mitigation for UAVs in precision agriculture.

The authors in [5] reviewed Unmanned Aerial Systems (UAS) literature based on the autonomy, cognition, and control features. The work highlights the need for UAS with safety measures and cognitive controls dealing correctly with the environment in real-world applications. [6] review literature works with algorithms for UAV safe landing, which are helped by systems using on-board camera and LiDAR for safe area detection employing vision and non-vision techniques. The authors indicate as possible future direction to approach the safe landing taking into account obstacle avoidance.

[7] present a hardware and software architecture for a legged-aerial autonomous system applied to missions in underground areas. In the autonomous aerial system, an artificial potential field formulation and the so-called deepest-point heading regulation technique, employing a 3D LIDAR for clustering points, allows an efficient reactive behavior for obstacle avoidance. UAV is transported by the legged ground robot, which uses a SLAM system to be aware of its localization. It is a complex system framework whose reported results show its capability to execute missions autonomously within underground areas.

A low-cost UAV system is introduced by [8] for crop data acquisition, handling issues such as wind and battery consumption. A path planning system optimizes the coverage paths, comparing the algorithms wavefront, Dijkstra, and spiral. A task planning system for UAV in [9] executes plant protection tasks, e.g., sowing or pesticide spraying. Dragonfly algorithm is applied to achieve near-optimal schedules for a set of 20 instances generated from real-world data. The work in [10] approaches the task assignment for UAVs by introducing a multi-objective mathematical model, where system design aspects such as the location of base stations, number of UAVs in each station, and UAV schedules for mission execution are also defined. The proposed solution must work under a disaster management situation. Therefore, a two-phase method is applied to deal with the computational complexity and improve accuracy when solving the proposed multi-objective model. Harpia will generate mission planning aiming autonomous execution such as [8,9]. However, we advance from these previous works by including Bayesian Networks (BN) and K-nearest neighbors (KNN) classifiers to improve mission and path planning decisions. Different from [10], we are not using mathematical formulation aiming to reduce the computational complexity of the overall embedded system.

Task planning and scheduling with risk mitigation deal with failure using Bayesian Network in [11].

The authors evaluated the system through Matlab simulations with satisfactory results when mitigating collisions within short periods. The so-called In-Flight Awareness Augmentation System (IFA2S) in [12] improves flight safety by applying Systems-Theoretic Process Analysis (STPA) and proposing a state machine for onboard operation. STPA is a hazard analysis technique that defines the requirements for risk mitigation, while the state machines define how the UAV should react based on those requirements.

Harpia approaches risk mitigation as [11] by applying BN to decide autonomously about re-planning based on battery health. The path planning system in Harpia will also deal with obstacle avoidance, executing online path re-planning. However, we are not assuming a dynamic scenario like the one described in [13]. We are dealing with non-fly zones similar to [12], but without defining all those risk requirements or using a state machine for autonomous behavior.

Table 1
Contributions with related work based on some specific features.

Reference	Mission	Risk	Path	Non-Convex	Onboard
[4]	No	No	Yes	No	Yes
[10]	Yes	No	No	No	No
[7]	Yes	No	Yes	Yes	Yes
[17]	No	Yes	Yes	Yes	No
[12]	No	Yes	Yes	Yes	Yes
[16]	No	Yes	Yes	Yes	Yes
[8]	No	No	Yes	No	No
[9]	Yes	No	Yes	No	Yes
[15]	No	Yes	Yes	Yes	No
[11]	Yes	Yes	No	No	Yes
[21]	Yes	No	Yes	No	Yes
[14]	No	Yes	Yes	Yes	Yes
[19]	No	Yes	Yes	Yes	No
[20]	Yes	No	No	No	No
[18]	Yes	No	Yes	No	No

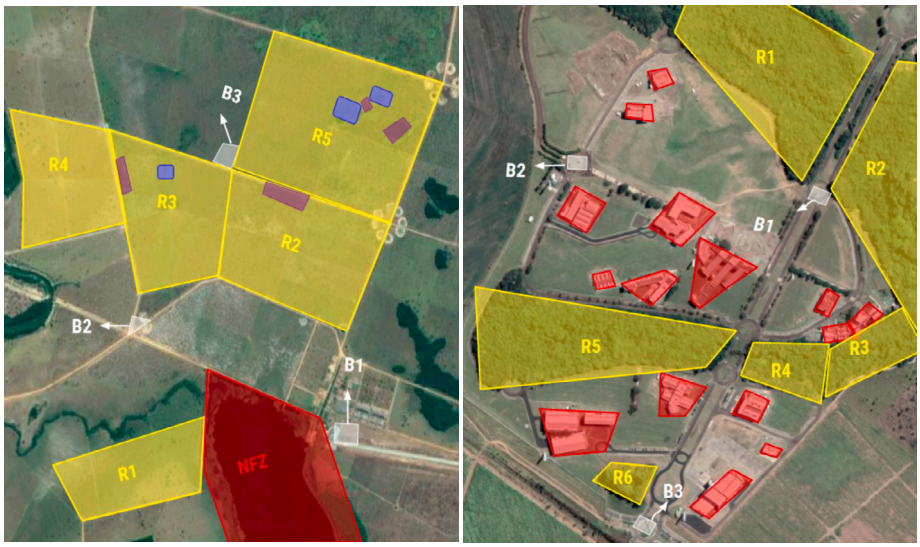
We are avoiding obstacles or non-fly areas by solving a non-convex path planning problem and using chance constraints for risk allocation, similar to [14] and [15]. The authors in [15] introduce a Mixed-Integer Linear Programming (MILP) formulation that finds optimal solutions for complex maps within a reduced computational time. Harpia will apply metaheuristics approaches once the systems must find solutions within a short time. The authors in [14] apply a hybrid method combining genetic algorithms with Voronoi diagrams to solve the non-convex path planning with risk allocation. The technique finds satisfactory solutions within a few seconds. Another hybrid evolutionary algorithm with satisfactory performance is described in [16], however, without handling the chance-constraint with risk allocation. The method combines the evolutionary approach with ray casting technique for obstacle avoidance. The authors in [17] approach the uncertain within multi-obstacle and dynamic environments by applying a deep reinforcement learning method for autonomous path planning. The twin delayed deep deterministic policy gradients algorithm is improved to deal with UAV path planning with satisfactory results from environment simulations using Unity 3D.

Harpia is a system architecture that can be embedded in UAVs. The authors in [18] present a related architecture with two main modules. The first module runs embedded on the UAV and controls its dynamics. The second module executes from the ground station, and it plans and updates the trajectory. Another similar system architecture, described in [19], focused on control the UAV trajectory. There are three algorithms for trajectory prediction, collision estimation, and obstacle avoidance based on laser sensing.

The authors in [20] introduce a hardware and software architecture for navigation and automatic spraying on crop fields for an unnamed helicopter. The ground station executes the path planning, but the flight and spraying control systems are on-board. The work of [21] presents a more robust system for spraying on crop fields, pest detection, among other tasks. The mission's target point can change online, leading to a path re-planing autonomously built during the flight. If the system detects weed locations, it sprays those areas locally. Robotic Operating System (ROS) packages are used to develop some functionalities.

The systems in [18–21,4,17,7] accomplish a specific type of task, where considerations about planning several tasks are not addressed. Harpia proposes a robust planning system where different tasks can be autonomously executed within a farm.

Table 1 summarizes the main features covered by Harpia against those reviewed in this section. **Mission** means those papers whose UAV systems deal with Mission Planning autonomously, and the same idea applies to **Path** for works reporting systems with Path Planning. It is also listed papers in **Risk** for those impending system failure or environment hazards, **Onbording** for systems wholly embedded in the UAV, **Non-Convex** for result in environments with obstacles or non-fly zones.



(a) Map 1 (b) Map 2

Fig. 1. Farm scenarios for planning tasks: regions of interest, no-fly zones and support bases.

3. Problem definition and proposed solution

Let's suppose that the actions performed on a farm by a drone include plantation imaging and spraying biological agents as shown in Fig. 1a. The regions R[1...5] represent spraying areas and the colors indicate the type of biological agent to be applied. The monitoring and spraying actions may require multiple flights, which also demand some stop by support base areas B[1...3] to recharge battery, fill tank with biological agents or process images.

The re-planning of actions can happen online for an unforeseen situation such as the addition or exclusion of some actions by the user or fault detection. The path planning occurs within a static and non-convex scenario. It means we know in advance the map, its areas of interest, and no-fly zones. This is a pretty realistic scenario when handling drones on farms. Thus, besides the mission planning problem, we are also solving a non-convex path planning problem with stay-in and stay-out areas similar to [15].

In the presence of a reliable communication link between the ground station and the embedded system of the UAV, it is possible to update the information about the map and mission in real-time. Our system covers these aspects when planning tasks and paths, reaching a reasonable level of self-awareness for the UAV regarding its surroundings and internal functioning. We clarify that the present research focuses on a system designed for autonomy in dealing with external and internal events. We illustrate that aspect in our current architecture's implementation for a scenario where the battery needs a recharge (internal event) within an environment where obstacle avoidance must be handled (external event) when replanning the trajectories. On the other hand, the details about how a specific sensor should be implemented or integrated into our architecture are not the main point of this work. Next, we define the problems approached more formally.

Definition 1. The planning problem in a farm scenario, as previously described, aims to find a solution $\mathcal{S}^* = \langle \mathbf{x}^*, \mathbf{u}^* \rangle$ from $\langle \mathbf{x}, \mathbf{u}, \mathcal{I}, \mathcal{M}, \mathcal{P}, \mathcal{J} \rangle$ such as:

- $\mathbf{x} = [x_0, x_1, ..., x_{T-1}]$: variables of the UAV states through the time steps $t = 0, ..., T-1$.
- $\mathbf{u} = [u_0, u_1, ..., u_{T-1}]$: variables of controls ($\mathbf{u}_t$) applied in UAV through the time steps.
- $\mathcal{I} = \left\langle \bar{\mathbf{x}}_0, \Sigma_{x_0} \right\rangle$ initial conditions with $\mathbf{x}_0 \sim \mathcal{N}(\bar{\mathbf{x}}_0, \Sigma_{x_0})$.
- $\mathcal{M} = \left\langle A, B, \Sigma_{w_t} \right\rangle$ stochastic plant model: $\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t + \omega_t$, where ω_t is an additive noise with $\omega_t \sim \mathcal{N}(0, \Sigma_{w_t})$.
- $\mathcal{P}$: *State Plan* (see *Def. 2*).
- $\mathcal{J}$: objective function to be optimized.

Definition 2. The state plan is given by the tuple $\mathcal{P} = \langle \mathcal{E}, \mathcal{C}, \mathcal{A} \rangle$, where:

- $\mathcal{E} = \{e_0, e_1, ...\}$: set of discrete events $e_i \in \mathcal{E}$.
- $\mathcal{C} = \{r_1, r_2, ...\}$: constraints related to the plant state through the time steps. $[\mathcal{C}_r^{lb}, \mathcal{C}_r^{ub}]$, for events.
- $\mathcal{A} = \{a_1, a_2, ...\}$: set of actions with each action $a \in \mathcal{A}$ happening between two events.

Fig. 2 illustrates events, constraints and episodes for our planning problem, where we have a directed acyclic graph with events drawn as vertices and episodes as rectangles. The episode is a constraint with $a = \langle e_a^S, e_a^E, \Pi_a, R_a \rangle$, where e_a^S and e_a^E are the initial and final event for the episode a. The set Π_a has the time steps when episode a is activated, and R_a summarizes the set of constraints to be satisfied. The authors in [22] report three episodes of interest:

1. *Start-in* ($a \in \mathcal{A}^S$): $x_t \in R_a$ holds in the episode beginning.
2. *End-in* ($a \in \mathcal{A}^E$): $x_t \in R_a$ holds at the episode ending
3. *Remain-in* ($a \in \mathcal{A}^R$): $x_t \in R_a$ while the episode is activated.

Definition 3. The non-convex path planning problem with chance-constraints is stated as:

$$\textit{Minimize } \Theta() = \sum_t \|u_t\| \tag{1}$$

where:

$$x_T = x_{goal} \tag{2}$$

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t + \omega_t \qquad \forall(t) \tag{3}$$

$$\omega_t \sim \mathcal{N}(0, \Sigma_{w_t}) \qquad \forall(t) \tag{4}$$

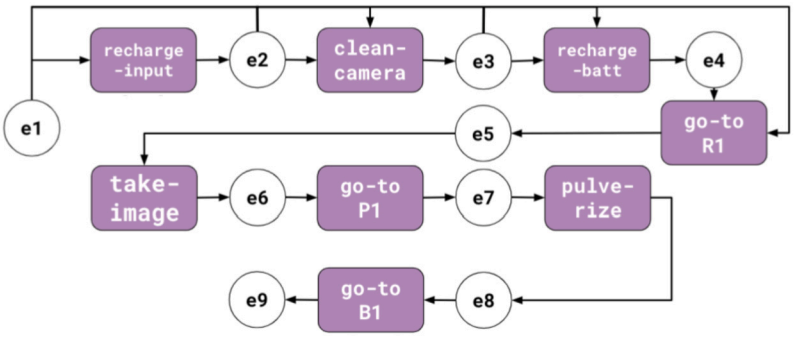


Fig. 2. Example of state plan for the planning problem in a farm scenario.

$$x_t \in \mathbb{I}_j \Leftrightarrow \bigwedge_{i \in H_j^{\mathbb{I}}} h_i^T x_t \leq g_i \tag{5}$$

$$x_t \in \mathbb{O}_j \Leftrightarrow \bigvee_{i \in H_j^{\mathbb{O}}} h_i^T x_t \geq g_i \tag{6}$$

The objective function (1) can minimize some metrics related to the control, while constraints (2) defines the goal as the last state of the UAV, when planning a trajectory. Constraints (5) and (6) describe hyperplanes defining convex regions for stay-out (obstacles or non-fly zones) and stay-in (landing spots) regions. Stay-out is a disjunction of linear constraints, while stay-in is established as the conjunction of linear constraints. The plan obtained from the State Plan in *Def.* 2 will demand the resolution of several path planning problems, as stated in *Def.* 3, during its execution.

4. Material and methods

In this section, we describe the proposed autonomous system, Harpia, which can run from a ground station or be embedded in the UAV platform. The UAV with Harpia embedded in it can operate without a link of communication with the ground station. In this case, our system can replace the pilot for designed decision-making and assume some behaviors for critical situations, e.g., performing an emergency landing or returning to base. The current decision-making features of Harpia improve flight safety, facilitate handling UAV operations, and increase the efficiency of carrying out missions. Flight safety improves since Harpia defines the mission plan and the related trajectories taking into account obstacle avoidance as an external factor and battery health as an internal one. The operation of the UAV under Harpia becomes easy once the pilot only needs to set the necessary inputs and start the UAV, with Harpia autonomously carrying out the mission plan and execution. Finally, the efficiency of carrying out missions arises from the system's ability to generate optimized plans using PDDL to describe the scenario of a current mission.

4.1. Harpia overview

In the current commercial solutions, the pilot usually is in charge of the UAV flight. For example, the scenarios described in Fig. 1 would generate one mission for each region, with the user defining the missions' execution sequence and the pilot controlling each flight. There is a need for commercial systems that execute repetitive missions without the pilot guidance as reported in [23].

In the case of in-flight diagnostic of the UAV systems, there are tailor-made commercial solutions, but the authors in [24] report the demand for systems with autonomous and adaptive diagnostics. Such systems can improve robustness, reduce costs and increase autonomy. Our system advances by adding more autonomy to plan and re-plan missions and trajectories.

Harpia system is developed on ROS following Fig. 3 architecture. The requirement to embed Harpia onboard is a companion computer, such as a Raspberry Pi model B or a similar one, with at least 4 GB SDRAM. In the presence of a reliable link between the ground station and the UAV, we do not need to embed the proposed system. In this case, Harpia can run using, e.g., a regular laptop at the ground station. On the other hand, if the communication link is unreliable, Harpian can be embedded in single-board computers as mentioned.

The systems in ROS nodes are:

- **ROSPlan:** We add here the Kings College's system ROSPlan, describe in [25], to use the available PDDL structure and solvers.
- **Decision Support & Making:** Manage the mission plan and execution.
  - **Mission Planning:** Calls ROSPlan features for mission planning.
  - **Fault Detection:** A Bayesian Network (BN) was developed to decide about keeping the plan execution, re-planning actions, or abort the current plan.
  - **Mission Goal Manager:** Updates goals since the user may add or remove them.
- **Path Planners:** This module interfaces with different path planners. The K-nearest neighbor (KNN) algorithm was employed as a machine learning approach to select a planner for each scenario.

Algorithms 1-3 summarize the iterations among ROS nodes in Harpia. The **Mission Goal Manager** (Algorithm 1) feeds the knowledge base to create the PDDL domain by sending the input data: UAV Info, Map, and Goals. UAV Info has the hardware attributes of the aircraft, e.g., avionics system and sensors features, which depend on the UAV platform employed and the mission executed. Map describes the current environment (map area, no-fly zones, support bases), and Goals have the regions of interest to be reached and the actions to be executed. We integrate all these Harpia Interface inputs using ROS tools. **Mission Goal Manager** will update inputs and ROSPlan knowledge base, while **Mission Planning** returns true. In Algorithm 2, the **Mission Planning** node calls **ROSPlan** to generate the problem and a plan to solve it.

If the problem or plan generation do not succeed, we have a problem and the **Mission Goal Manager** will receive False reporting it to the system. For instance, the users can receive a message telling that it was not possible to execute the mission with the current data. If everything works, **Mission Planning** will execute **ROSPlan** to dispatch an action to **Mission Manager**. In Algorithm 3, the **Fault Detection** system is called to verify if the action can be executed.

At this point, BN will evaluate the feasibility of the current sequence of actions. If there is a problem, the system will try re-plan the mission by calling **ROSPlan** again with the current UAV state. If it is not possible to generate a new plan, **Mission Planning** will receive false and report that plan generation does not succeed to **Mission Goal Manager**. Otherwise, we have re-planned the mission and **Mission Manager** will execute the plan actions. Next, we describe how PDDL, BN and path planning algorithms work in Harpia.

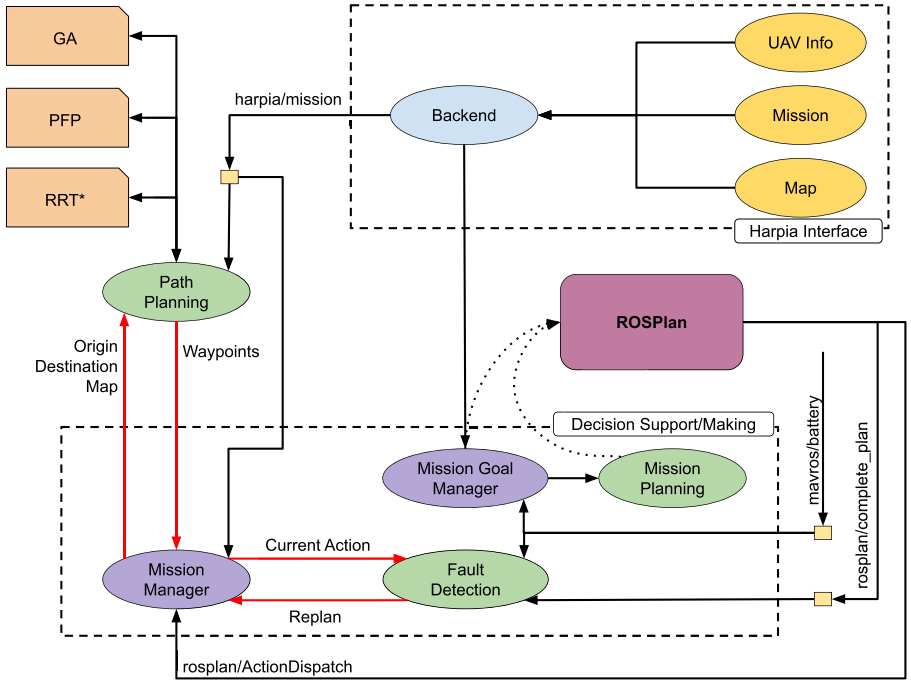


Fig. 3. Description of the ROS nodes and communication.

Algorithm 1: Mission Goal Manager.

```
Input : UAV, Map, Goals
ROSPLAN.knowledge base(UAV, Map, Goals)
while Mission_Planning() do
    UAV,MAP,Goals ← updateInput()
    ROSPLAN.knowledge_base(UAV, Map, Goals)
end
```

Algorithm 2: Mission Planning.

```
stop ← False;
if problem ← ROSPLAN.generateProblem() then
    if plan ← ROSPLAN.generatePlan(problem) then
        repeat
            action ← ROSPLAN.actionDispatch(plan);
            if action = = NULL then
                stop ← True;
            else
                stop ← Mission_Manager(action,plan);
        until stop;
return stop;
```

Algorithm 3: Mission Manager.

```
Input : action,plan
fault ← Fault_Detection(action,plan) ;
if fault then
    re-plan ← ROSPLAN.generatePlan(problem,action,plan);
    if not re-plan then
        return False;
Do_Action(action,plan);
return False
```

4.2. PDDL

The task planning problem is solved by using the PDDL2.1 language, which is expressive enough for domain and problem definition. PDDL2.1 will also facilitate to describe temporal constraints [26]. Fig. 4 gives an overview of the domain, where we assume high-level planning to schedule actions. For instance, the UAV must load the right input for spraying. The battery must execute the trajectory, and the on-board

```
(define (domain harpia)
        (:requirements ...
        (:types region - object
                base  - region)
        (:functions
                (battery-amount)
                (distance ?from-region - region ?to-region -region) ; ; m
                (discharge-rate-battery) ; ; % / s
        ...)
        (:predicates
                (at ?region - region)
                (taken-image ?region - region)
                (picture-goal ?region - region)
        ...)
        (:durative-action go_to_picture ...
        (:durative-action go_to_pulverize ...
        (:durative-action go_to_base ...
        (:durative-action recharge_input ...
        (:durative-action pulverize_region ...
        (:durative-action take_imate ...
        (:durative-action recharge_battery ...
)
```

Fig. 4. PDDL Domain overview.

camera must usually be clean after a spraying action. PDDL2.1 allows us to properly define domain and actions for a temporal constrained plan, where the UAV can verify preconditions before performing actions. This will prevent the aircraft going from a region to another without enough battery, leading it to recharge the battery in a support base (`go_to_base`). UAV autonomously goes to the base to load the battery as often as necessary to carry out a mission.

Fig. 5 illustrate the model for a problem. Our system establishes the communication between the user and ROSPlan. As mentioned, the user only needs to provide information about the map, mission goals, and UAV hardware. The system updates the knowledge base for each mission request, calls for a plan, and the planned dispatch. In this example, the mission goals are to capture the images of regions three to six. We only add the distances between the goals regions and the bases to avoid unnecessary complexity during the plan schedule search.

4.3. Bayesian Network

The Bayesian Network (BN) is a semantic way to represent probabilistic models, structured as a directed graph [27]. The nodes and

```
(define (problem task)
(:domain icra)
(:objects
    region_1 region_2 region_3 region_4 region_5 region_6 - region
    base_1 base_2 base_3 - base
)
(:init
    (at base_1)
    (can-take-pic)
    (has-picture-goal)
    (= (battery-amount) 100)
    (= (input-amount) 0)
    (= (recharge-rate-battery) 2.142)
    (= (discharge-rate-battery) 0.042)
    (= (battery-capacity) 100)
    (= (input-capacity) 3)
    (= (velocity) 3.5)

    (its-not-base region_1) ...
    (picture-goal region_1) ...

    (= (distance region_1 region_2) 338.825)
    (= (distance region_1 region_3) 655.288) ...
    (= (distance region_2 region_1) 338.825 ...

    (= (picture-path-len region_1) 1000) ...

)
(:goal (and
    (taken-image region_3)
    (taken-image region_4)
    (taken-image region_5)
    (taken-image region_6)
    (at base_3)
))
(:metric minimize (total-time)))
```

Fig. 5. PDDL Problem overview.

Table 2
Probability distribution.

Battery	Probability	Battery	Probability
0<Y<15	0.20	15<=Y<30	0.50
30<=Y<60	0.75	60<=Y<80	0.85
-	-	80<=Y<=100	0.95

edges represent variables and conditional dependency between nodes, respectively. Harpia calls a BN within the Fault Detection system every time ROSPlan activates an action. The idea is to verify the feasibility of the plan based on the level of the battery. Fig. 6 shows the incremental build process of the BN, which is related to the next action to be performed, e.g., BN is initiated before the drone takes a picture in region 2.

Table 2 has the probability distribution employed, based on the Li-ion discharge voltage curve, where Y is the percentage of available battery to execute the plan's selected action, and P(Y) is the chance of executing such action without a re-planning. For example, we can have $Y = 85\%$, meaning a battery with 85% of the full capacity, where P(Y) = 0.95 is the chance of executing the selected action following the current plan. Thus, if the probability of executing an action is greater or equal to keeping the plan, BN evaluates the plan. BN adds a new child node as the next action and adjusts the previous action as evidence. The process will continue for the whole plan, or if a node evaluation demands a re-plan. Fig. 6 shows when the probability of success for an action is smaller than the drone's probability of not accomplishing an action in the future. At this point, the system stops the BN process and triggers a re-plan.

In Fig. 7, we have a scenario where the system decides to re-plan in two different points after it calls the BN. In Fig. 7a, the black arrows represent the original plan, but the BN identifies the need to recharge before the end. This happens since the percentage of battery consumption becomes greater than expected until this point, which means a low available battery as the UAV reaches region R2, leading to the re-plan of the current sequence of actions. The new plan is shown in blue, where the UAV will go to the base after the action in R4 to recharge the battery, mitigating a future hazard. We assume that the battery is not working as expected, which means the discharge rate is not following the desired behavior, e.g., due to wind resistance, an increase in the previous path to avoid obstacles or even a battery failure. Thus, when the aircraft arrives at R3 in Fig. 7b, another action is dispatched by ROSPlan and the BN identifies that it is impossible even to reach R4. Next, a second re-plan is called and it decides to advance in the battery recharge.

4.4. Path Planning

The Path Planning system will receive origin, destination, and obstacle positions to calculate a feasible route. It is called by *Do_Action*() in Algorithm 3 with trajectories being requested through *go_to* actions. The trajectory can be generated by one of the three available algorithms:

Hybrid Genetic Algorithm (HGA) - [16], Rapidly-exploring Random Trees (RRT) and Potential Field Planning (PFP) - [28].

We made changes in RRT and PFP algorithms to address obstacle avoidance. RRT applies Ray Casting (RC) algorithm to check collisions following the approach introduced in [16]. The Ray Casting (RC) algorithm is used in computer graphic applications since it traces rays from a source and finds the nearest point blocking the beam. Polygons represent non-fly zones or obstacles in our scenarios, and RRT will avoid such polygons by using RC to validate the waypoints sampled when building branches. In this context, RC traces horizontal rays from each waypoint and calculates the number of intersections with the polygon.

In RRT, we first define the Ray Cast Point (RC_P) to verify a waypoint inside the obstacle. If the ray intercepted the polygon an odd number, the waypoint is inside the area; if it blocked an even number, the waypoint is outside the area, as summarized by expression (7). Next, Ray Cast Segment (RC_S) checks if there are intersections between the segment of two consecutive waypoints and the polygons, as stated by expression (8):

$$RC_P(x_s, \mathbb{O}_j) = \begin{cases} 1 & \text{, if } x_s \in \mathbb{O}_j \\ 0 & \text{, otherwise} \end{cases} \tag{7}$$

$$RC_S(x_s, x_{s+1}, \mathbb{O}) = \sum_{j=0}^{|\mathbb{O}|} |\overline{x_s, x_{s+1}} \bigcap \mathbb{O}_j| \tag{8}$$

where x_s and x_{s+1} are two consecutive waypoints defining the segment $\overline{x_s, x_{s+1}}$, and $\mathbb{O}$ is the set of obstacle with $\mathbb{O}_j \in \mathbb{O}$.

Equations (10) and (12) calculate the repulsive field in PFP algorithm, whose novelty is the inclusion of chance-constrains.

$$PF(x_t) = RP(x_t) + AP(x_t) \tag{9}$$

$$AP(x_t) = \frac{d(x_t, D)}{d(O, D)} \tag{10}$$

$$RP(x_t) = \sum_{j=1}^{|\mathbb{O}|} (Pr(x_t \in Z_{\mathbb{O}_j})) \tag{11}$$

$$Pr(x_t \in Z_{\mathbb{O}_j}) = 1 - F(x_t) \tag{12}$$

Equation (9) adds up attractive and repulsive potentials in Equations (10) and (12), respectively. Euclidean distance $d(A, B)$ measures the UAV's position (x_t) from the destination D, where O is the origin. The repulsive field applies chance constraints as reported in [15], thus, the probability of collision with an obstacle $Z^i_{\Phi^n}$ is given by Pr, and Φ is the set of obstacles. Equation (9) indicates how to find Pr from the cumulative distribution function $F(x_t)$ as illustrated by Fig. 8 for a normal distribution.

Each algorithm produces feasible routes at different run times and generates a path with distinct advantages. Moreover, the path planning system chooses the path planner based on the situation, where the variables analyzed to make a choice are: battery health, obstacle quantity between the regions, and the distance (in a straight line) between the origin and destination points.

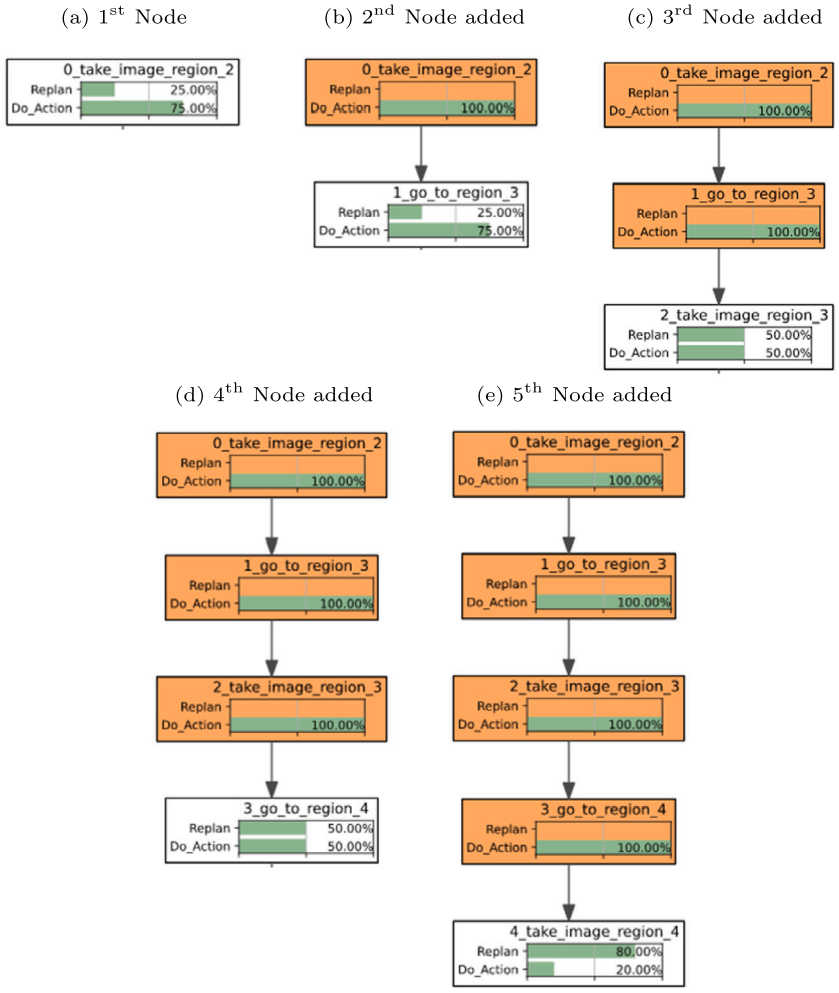


Fig. 6. BN result example where replan is needed. The images show the addition of new nodes on the BN, setting the actions that happened and calculating the probability of re-plan for added nodes.

The planner choice is made using K-nearest neighbors (KNN) classifier with $K = 3$ that evaluated: battery health (b), time to run the algorithm (t), ratio (r, see Equation (15)), route length (l), quantity of waypoints (q). These variables were normalized following Equation (16). The variables are calculated for each possible origin and destination on Map 1 and Map 2, by executing each path planner. Next, the values of the trajectories are given by Equation (13) and used as input for training the KNN classifier.

$$fitness = a + t + r + l + q \tag{13}$$

$$a = b * r \tag{14}$$

$$r = \frac{\sum_{i=1}^{n} d(x_i, x_{i+1})}{d(O, D)} \tag{15}$$

$$\hat{\varsigma} = \frac{\sigma - min(\sigma)}{max(\sigma) - min(\sigma)}, \forall \sigma \in \{a, t, r, l, q\} \tag{16}$$

In Fig. 9, we can observe some tendencies in the choice of each algorithm. For example, HGA is usually chosen when there are fewer obstacles, while RRT is usually chosen for longer distances.

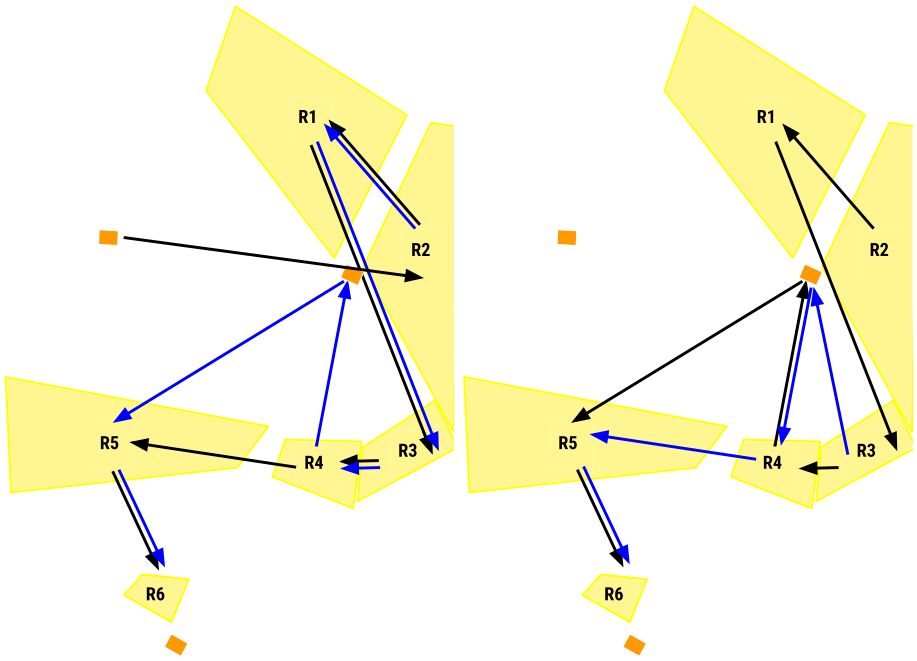
5. Experimental results

We report simulation results for a real-world quad-copter with the following configuration: 9 kg of maximum take-off weight (MTOW), 72 km/h of top cruising speed, 22 min of MTOW autonomy, and 7.0 g/w of MTOW Efficiency. Simulations will not replace the potential findings in real-world scenarios, and we tried to reduce such impact by using the real-world model provided by the Gazebo simulator.

Also, since our system operates autonomously, there is no need for some usual concerns on farms, like loss of communication or minimum bandwidth between the base station and the UAV. The reason is that Harpia operates without the need to exchange data with a ground station. Thus, under a possible lack of communication or minimum bandwidth scenario, Harpia will keep re-planning missions and trajectories only based on the UAV's sensors data.

The simulations evaluate Harpia on twenty-one scenarios. Gazebo PX4 Firmware is employed for drone simulation, and the POPF solver is executed to plan actions in the PDDL 4.1. We simulate a quad-copter UAV with $3.5 m/s$ efficient velocity, input capacity for three pulverizations, on-board camera, and forty minutes of battery autonomy. The scenarios are defined from the two maps shown in Fig. 1, where Map1 has one non-fly zone (NFZ) (obstacle), and Map2 has 14 NFZs. We assume three support bases for both maps and missions with two, four, and six regions of interest (goals).

A total of four levels of battery health is considered for testing the ROSPlan (Mission Planning), BN (Fault and Diagnosis) and KNN (Path Planning) response to detect a problem and re-plan actions and trajectories. For instance, if the drone should fly forty minutes but the battery health is 50% and the discharge rate is two times greater, it will fly only 20 minutes. Table 3 shows the results for each scenario, reporting the number of re-plannings, the KNN decision made for each path planer, and average CPU time spent by the POPF solver.



(a) First Re-plan (b) Second Re-plan

Fig. 7. Re-planning sequence of actions. The black arrows show the previous sequence of actions, and blue arrows indicate the re-planning one.

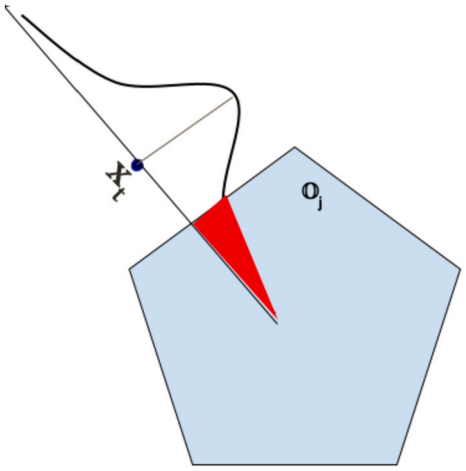


Fig. 8. Risk incurred by the uncertainty related to the state x_t.

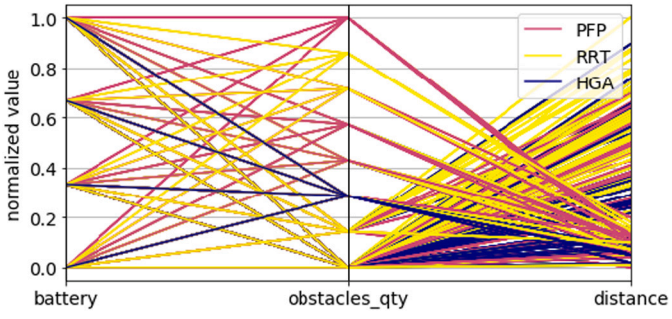


Fig. 9. KNN and path planner selection based on some features.

The minimum distance in Map 1 for the launch site, region and base is approximately 3 km, which becomes infeasible for the drone to fly with 30% of battery health to reach a base. Moreover, we test the 30% battery health only for Map 2. As it should be expected, the results in Table 3 indicate the increase in the number of re-planning calls when the battery health decreases (see Fig. 10). However, the calls from Mission Planning to POFP solver (ROSPlan) do not increase the average time spent re-planning actions.

In the decisions about planning and re-planning of missions, supported by the BN, only 2.3% of them lead the UAV to land with less than 5% of battery at the end. UAV is supposed to conclude the mission with at least 5% of battery in our simulations. KNN is also making decisions as expected since it chooses different planners for different scenarios. HGA, RRT and PFP were chosen with 100%, 95% and 96% of accuracy, respectively, based on KNN learning phase. It means that HGA was properly chosen, based on the learning phase of KNN, while RRT and PFP were exchanged sometimes by KNN for few scenarios over the 21 evaluated in Table 3.

Table 3
Results for each scenario.

Battery Health	Obstacles Qty	Total Goals	Re-plan Needed	Chosen Path Planners			Average CPU Plan Time (s)
				AG	RRT	PFP	
100%	1	2	1	1	2	1	0.02
		4	0	1	4	2	0.12
		6	0	4	3	3	0.76
	14	2	0	1	1	1	0.01
		4	0	2	1	2	0.06
		6	0	5	1	1	0.2
70%	1	2	2	1	0	2	0.01
		4	0	0	6	1	0.12
		6	0	3	5	3	0.84
	14	2	0	0	2	1	0.02
		4	0	2	1	2	0.06
		6	0	4	2	2	0.2
50%	1	2	3	0	2	2	0.2
		4	3	0	6	2	0.08
		6	2	4	5	2	0.23
	14	2	0	0	2	1	0.01
		4	2	1	1	3	0.03
		6	2	6	1	1	0.24
30%	14	2	1	0	2	1	0.01
		4	2	1	3	2	0.1
		6	8	6	0	3	0.13

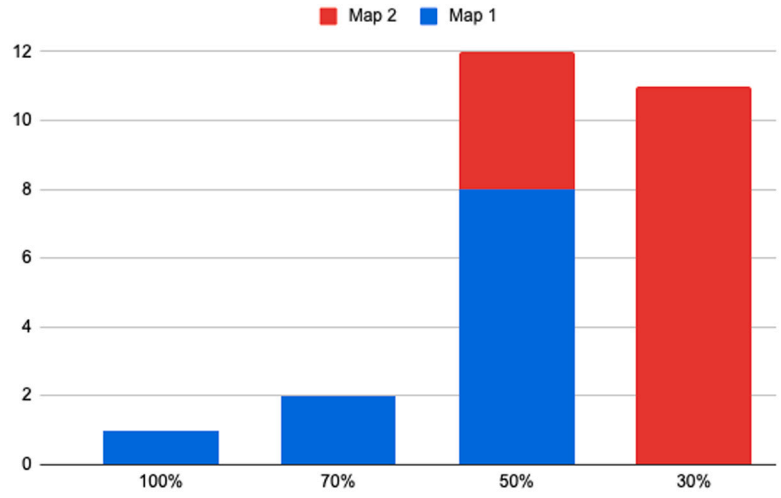


Fig. 10. Re-plan Calls per Battery Health.

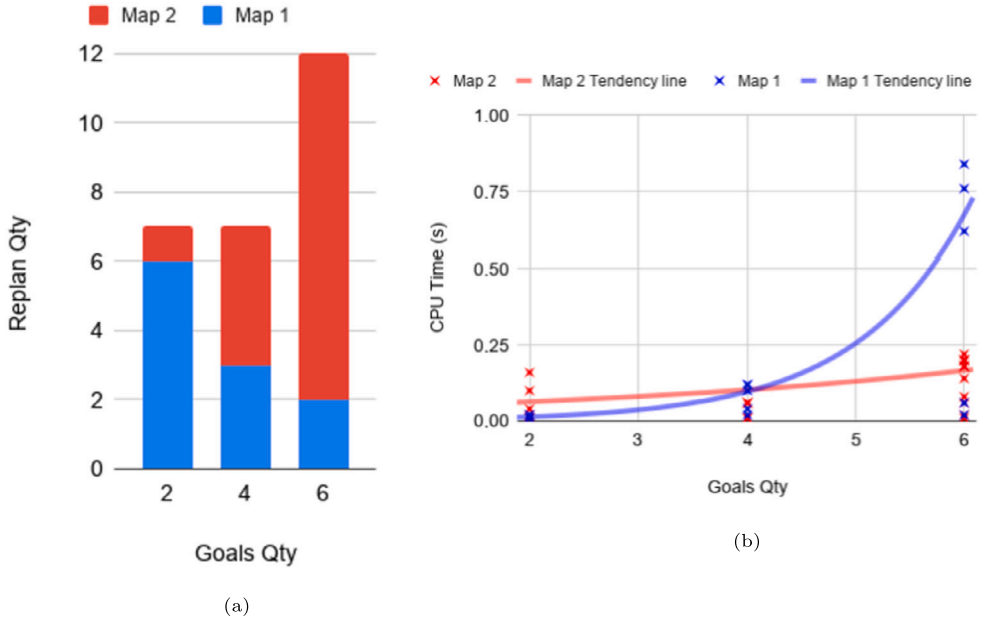


Fig. 11. Execution performance: (a) re-plan calls per goals; (b) CPU time per goals.

Fig. 11a shows the number of re-planning actions based on the number of goals, while Fig. 11b has the time spent re-planning on Map 1 and Map 2. Map 2 has more obstacles, which demand re-plannings when the number of goals increases. However, the overall time is less than 0.84 seconds, which is fast enough for the problem at hand.

There is no re-plan in Map 2 for 70% of battery health once the max distance between regions of interest is 1 km, and the UAV can fly for 28 minutes. Otherwise, in the case of 50% and 30% battery health, we can observe the re-plan increase based on less battery healthy and more goals to be accomplished in Fig. 11a. The re-plan is a safety measure for the mission; thus, it is mandatory to find a new plan quickly. This requirement is satisfied in our model, as shown in Fig. 11b with a maximum CPU time of around 0.76 seconds.

6. Conclusion

This work introduced Harpia as an autonomous system that combines PDDL for task planning, BN to evaluate mission execution, and KNN algorithm to select a path planner. The task planning is a Temporally Flexible State Plan, while the path planning must solve a non-convex problem. Harpia was able to manage battery failure with the BN, executing the necessary amount of re-planning to accomplish the mission. BN supported the mission system to make decisions, leading the UVA to land with the expected battery health in 97.5% times for all re-plannings executed. KNN selected the path planner to avoid obstacles and reach the region of interest in all scenarios. In this case, KNN decisions have 100% of accuracy when deciding about HGA execution as well as 96% and 95% about PFP and RRT, respectively. As future work, we will validate the current version on a farm. Data from other sensors can be integrated into the BN as the GPS, improving the system security and fault detection capabilities. The path planning module will add other algorithms, such as decision trees or even neural networks to select the path planners. Finally, to add more security features when a failure occurs, an emergency system is under development to execute emergency landing in safe regions.

Declaration of competing interest

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

Data availability

Data will be made available on request.

Acknowledgements

Research developed using computational resources and financial support of Centro de Ciências Matemáticas Aplicadas à Indústria (CeMEAI) supported by Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) - CEPID-CeMEAI (FAPESP 2013/07375-0). Also supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq 315158/2020-4.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.atech.2023.100191.

References

[1] J. Kim, S. Kim, C. Ju, H.I. Son, Unmanned aerial vehicles in agriculture: a review of perspective of platform, control, and applications, IEEE Access 7 (2019) 105100–105115.
[2] P. Radoglou-Grammatikis, P. Sarigiannidis, T. Lagkas, I. Moscholios, A compilation of UAV applications for precision agriculture, Comput. Netw. 172 (2020) 107148.
[3] E. Blasch, Autonomy in use for information fusion systems, in: NAECON 2018-IEEE National Aerospace and Electronics Conference, IEEE, 2018, pp. 1–8.
[4] J. Ma, Y. Li, H. Liu, Y. Wu, L. Zhang, Towards improved accuracy of UAV-based wheat ears counting: a transfer learning method of the ground-based fully convolutional network, Expert Syst. Appl. 191 (2022) 116226.
[5] A. Montazeri, A. Can, I.H. Imran, Unmanned aerial systems: autonomy, cognition, and control, in: Unmanned Aerial Systems, Elsevier, 2021, pp. 47–80.
[6] M.S. Alam, J. Oluoch, A survey of safe landing zone detection techniques for autonomous unmanned aerial vehicles (UAVs), Expert Syst. Appl. (2021) 115091.
[7] B. Lindqvist, S. Karlsson, A. Koval, I. Tevetzidis, J. Haluška, C. Kanellakis, A.-a. Agha-mohammadi, G. Nikolakopoulos, Multimodality robotic systems: integrated combined legged-aerial mobility for subterranean search-and-rescue, Robot. Auton. Syst. 154 (2022) 104134.
[8] L.V. Campo, A. Ledezma, J.C. Corrales, Optimization of coverage mission for lightweight unmanned aerial vehicles applied in crop data acquisition, Expert Syst. Appl. 149 (2020) 113227.
[9] F. Sun, X. Wang, R. Zhang, Task scheduling system for UAV operations in agricultural plant protection environment, J. Ambient Intell. Humaniz. Comput. (2020).
[10] B.D. Song, H. Park, K. Park, Toward flexible and persistent UAV service: multi-period and multi-objective system design with task assignment for disaster management, Expert Syst. Appl. 206 (2022) 117855.
[11] C. Hireche, C. Dezan, J.-P. Diguet, L. Mejias, Bfm: a scalable and resource-aware method for adaptive mission planning of UAVs, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2018, pp. 6702–6707.
[12] A.L. Mattei, E.S. Orbital, C.F. Toledo, J. da Silva Arantes, O. Trindade Jr., Unmanned aerial vehicles flight safety improvement using in-flight awareness, Intell. Inf. Manag. 13 (2) (2021) 97–123.
[13] C.E. Luis, M. Vukosavljev, A.P. Schoellig, Online trajectory generation with distributed model predictive control for multi-robot motion planning, IEEE Robot. Autom. Lett. 5 (2) (2020) 604–611.
[14] M.S. Arantes, J.S. Arantes, C.F.M. Toledo, B.C. Williams, A hybrid multi-population genetic algorithm for UAV path planning, in: Genetic and Evolutionary Computation Conference, 2016.
[15] M. Arantes, C. Toledo, B. Wiliams, M. Ono, Collision-free encoding for chance-constrained nonconvex path planning, IEEE Trans. Robot. 35 (2) (2019) 433–448.
[16] G. Moura Souza, C.F.M. Toledo, Genetic algorithm applied in UAV's path planning, in: 2020 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2020, pp. 1–8.
[17] S. Zhang, Y. Li, Q. Dong, Autonomous navigation of UAV in multi-obstacle environments based on a deep reinforcement learning approach, Appl. Soft Comput. 115 (2022) 108194.
[18] I. Prodan, S. Olaru, R. Bencatel, J.B. de Sousa, C. Stoica, S.-I. Niculescu, Receding horizon flight control for trajectory tracking of autonomous aerial vehicles, Control Eng. Pract. 21 (10) (2013) 1334–1349, https://doi.org/10.1016/j.conengprac.2013.05.010.
[19] S. Ramasamy, R. Sabatini, A. Gardi, J. Liu, LIDAR obstacle warning and avoidance system for unmanned aerial vehicle sense-and-avoid, Aerosp. Sci. Technol. 55 (2016) 344–358, https://doi.org/10.1016/j.ast.2016.05.020.
[20] X. Xue, Y. Lan, Z. Sun, C. Chang, W.C. Hoffmann, Develop an unmanned aerial vehicle based automatic aerial spraying system, Comput. Electron. Agric. 128 (2016) 58–66, https://doi.org/10.1016/j.compag.2016.07.022.
[21] B.H.Y. Alsalam, K. Morton, D. Campbell, F. Gonzalez, Autonomous UAV with vision based on-board decision making for remote sensing and precision agriculture, in: 2017 IEEE Aerospace Conference, 2017, pp. 1–12.
[22] M. Ono, B.C. Williams, L. Blackmore, Probabilistic planning for continuous dynamic systems under bounded risk, J. Artif. Intell. Res. 46 (2013) 511–577.
[23] P. Cohn, A. Green, M. Langstaff, M. Roller, Commercial drones are here: the future of unmanned aerial systems, McKinsey & Company.
[24] R. Nouacer, M. Hussein, H. Espinoza, Y. Ouhammou, M. Ladeira, R. Castiñeira, Towards a framework of key technologies for drones, Microprocess. Microsyst. 77 (2020) 103142.
[25] V. Sanelli, M. Cashmore, D. Magazzeni, L. Iocchi, Short-term human-robot interaction through conditional planning and execution.
[26] M. Fox, D. Long, Pddl2. 1: an extension to PDDL for expressing temporal planning domains, J. Artif. Intell. Res. 20 (2003) 61–124.
[27] S.J. Russell, P. Norvig, Artificial Intelligence: a Modern Approach, 3rd edition, Pearson, 2009.
[28] A. Sakai, D. Ingram, J. Dinius, K. Chawla, A. Raffin, A. Paques, Pythonrobotics: a python code collection of robotics algorithms, arXiv preprint, arXiv:1808.10703.