Czech Technical University in Prague

Faculty of Electrical Engineering
Department of Cybernetics
Multi-robot Systems

Area Exploration by Unmanned Multirotor Helicopters using Probabilistic Quadtrees Algorithm

Bachelor's Thesis

Martin Koudelka

Prague, May 2022

Study programme: Open Informatics
Branch of study: Artificial Intelligence and Computer Science

Supervisor: Ing. Tomáš Báča, Ph.D.

Author statement for undergraduate thesis:

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date 20.5.2022 ....

Signature

Acknowledgments

Firstly, I would like to express my gratitude to my supervisor Ing. Tomáš Báča, Ph.D.for support and advice during the work and giving me the opportunity to transfer my work from my first supervisor. Although I could not finish the work with him, I would like to thank my first supervisor Ing. Hendrik de Bruin for advice during the initial phases of the work. Lastly I would like to thank my family for support and giving me the opportunity to study.

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Koudelka Martin**	Personal ID number:	**492275**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Open Informatics**		
Specialisation:	**Artificial Intelligence and Computer Science**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Area Exploration by Unmanned Multirotor Helicopters using Probabilistic Quadtrees Algorithm

Bachelor’s thesis title in Czech:

Prohledávání vymezené oblasti bezpilotními helikoptérami pomocí pravděpodobnostního kvadratického stromu

Guidelines:

This thesis aims to explore the problem of autonomous scanning of a designated ground or water surface for the desired object using multiple multirotor Unmanned Aerial Vehicles (UAVs). The goal is to extend an existing approach [1] from a single UAV to multiple UAVs, to implement it, and to verify it using realistic simulations in Gazebo/ROS simulator. Such an approach could serve, e.g., for autonomous search for surface litter in an ocean. Some form of autonomous object recognition should be utilized, in order to realistically model parameters of such computer vision-based detection. Realistic simulations using the MRS UAV System [2] should point to whether the implemented approach scales well with more UAVs and whether using multiple UAVs is time-effective. The goals of the thesis are as follows:
Study the published approach [1] for utilizing a single UAV for exploring a designated area using the probabilistic quadtree algorithm.
Familiarize yourself with the MRS UAV System for control, estimation, and simulation of multirotor helicopters [2].
Extend the approach published in [1] for multiple UAVs and attempt to solve possible drawbacks that such extension might bring, e.g., the possibility of mutual UAV collisions.
Implement the approach using the MRS UAV System [2] and utilize some of the existing tools for computer visual detection of objects on the ground (e.g., AprilTag2, Whycon, or the MRS’s Object Detect).
Verify the implemented approach using multi-robotic simulations and if possible, test it against a baseline (naive pre-planned systematic scanning using a single UAV).

Bibliography / sources:

[1] Carpin, Stefano, Derek Burch, and Timothy H. Chung. "Searching for multiple targets using probabilistic quadtrees." 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2011.
[2] Tomas Baca, Matej Petrlik, Matous Vrba, Vojtech Spurny, Robert Penicka, Daniel Hert and Martin Saska. The MRS UAV System: Pushing the Frontiers of Reproducible Research, Real-world Deployment, and Education with Autonomous Unmanned Aerial Vehicles. Journal of Intelligent & Robotic Systems 102(26):1–28, May 2021.

Name and workplace of bachelor’s thesis supervisor:

Ing. Tomáš Báča, Ph.D. Multi-robot Systems FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **25.01.2022** Deadline for bachelor thesis submission: **20.05.2022**

Assignment valid until: **30.09.2023**

Ing. Tomáš Báča, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Abstract

Thos thesis deals with using multiple Unmanned Aerial Vehicles (UAVs) to find targets inside a designated area. A modified quadtree data structure enhanced by probability property is used to achieve that. The objective of this work was to implement this approach to area exploration inside the Robot Operating System (ROS) and test it using multiple vision algorithms. The results were compared with a baseline naive exploration method.

Keywords Unmanned Aerial Vehicles, Path Planning, Probabilistic Quadtree, Area Exploration

Abstrakt

Tato práce se zabývá použitím několika bezpilotních helikoptér (UAVs) k nalezení cílů ve vymezené odlasti. K dosažení tohoto cíle je použita modifikovaná datová struktura kvadratického stromu rozšířená o pravděpodobnostní vlastnosti. Cílem práce je implementovat tento přístup k prohledávání oblasti v Robot Operating System (ROS) a otestovat ho za použití několika algoritmů pro strojové vidění. Výsledky byly porovnány s naivní prohledávací metodou.

Klíčová slova Bezpilotní helikoptéry, Plánování trasy, Pravděpodobnostní kvadratický strom, Prohledávání oblasti

Abbreviations

API	Application programming interface
FOV	Field of View
CTU	Czech Technical University in Prague
GPS	Global Positioning System
LiDAR	Light Detection and Ranging
MRS	Multi-robot Systems Group
ROS	Robot Operating System
UAV	Unmanned Aerial Vehicle
USV	Unmanned Surface Vehicle

Contents

Chapter 1

Introduction

Unmanned Aerial Vehicles (UAVs) have seen a great interest in last decades by both scientific and commercial sectors. The growth can be attributed to the development of used sensors, controls and energy storage which provide more accurate navigation, more reliable flight behaviour and longer flight times respectively. Modern UAVs used in research are usually multirotor helicopters equipped with a battery pack, sensors, an onboard computer and often a small camera. Based on the used hardware the UAVs can stay in the air for approximately 20 minutes or longer if lower powered components are used.

Area exploration is one of common use cases for a group of UAVs. Small size, relatively low price and independence make UAVs very capable at finding targets in both large areas as well as in narrow places which are usually hard to access. The targets of the search mission can be various, in general anything that the vision algorithm is configured or taught to recognize.

UAVs are not limited to only flying over ground. They can be configured and prepared for flights over bodies of water and even for landing on the water surface. In the case of aquatic missions the UAVs often work in combination with Unmanned Surface Vehicles (USVs) or regular boats. Boats can be used as a signal provider, a central node in a system and often as recharge stations. Common targets during aquatic search missions are for example lifeboats, makeshift boats, cargo, garbage floating on the water surface or oil spills.

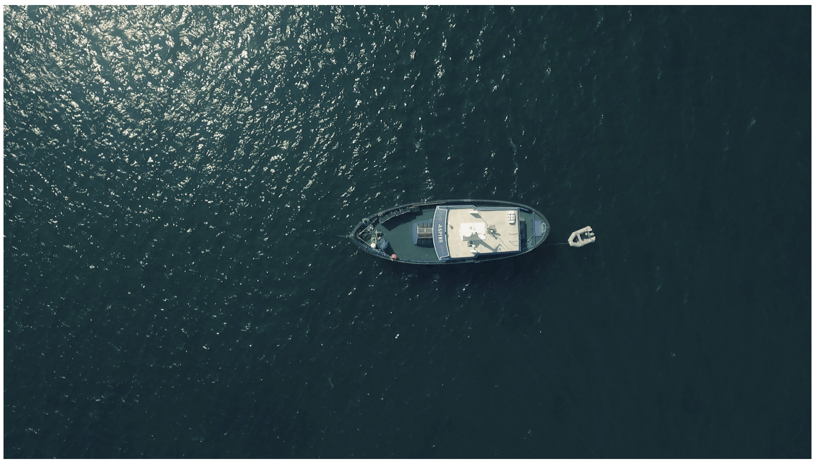


Figure 1.1: Example of a target of a search mission.[1]

Goals of the thesis have been set as:

- Study the published approach [1] for utilizing a single UAV for exploring a designated area using the probabilistic quadtree algorithm.

[1]Source: https://unsplash.com/photos/rI_Y_LL30E4

- Familiarize yourself with the MRS UAV System for control, estimation, and simulation of multirotor helicopters [2].
- Extend the approach published in [1] for multiple UAV and attempt to solve possible drawbacks that such extension might bring, e.g., the possibility of mutual UAVs collisions.
- Implement the approach using the MRS UAV System [2] and utilize some of the existing tools for computer visual detection of objects on the ground (e.g., AprilTag2, Whycon, or the MRS's Object Detect).
- Verify the implemented approach using multi-robotic simulations and if possible, test it against a baseline (naive pre-planned systematic scanning using a single UAV).

To achieve set goals the first chapter of the thesis will explain the Probabilistic quadtree approach introduced in [1]. In the second chapter the context of the implementation will be described. This includes used framework for robotics communication Robot Operating System (ROS) and particular system providing controls for the UAVs developed by Multi-robot Systems Group (MRS) of Faculty of Electrical Engineering on Czech Technical University in Prague. The third chapter focuses on vision algorithms, explains the basic requirements for an algorithm to be used inside the implemented system. The chapter also introduces two utilized vision algorithms — Object detect developed by MRS and AprilTag. The fourth chapter explains the methodology of experiments and result gathering which will be needed to evaluate the success of the thesis in the following last chapter. The results of the experiments are presented in the last chapter. The chapter also attempts to provide an explanation of achieved results and evaluates the success of the implementation.

1.1 State of the art

The topic of path planning for a group of UAVs is currently heavily studied. This comes at no surprise given the usefulness of UAV groups during area surveillance and Search and rescue task. The research is focused on finding the best possible path for the group ahead of time or dynamically adjusting the path based on the received information from sensors mounted on the UAVs. Many works also incorporate hardware limitations such as limited flight time due to the battery capacity or limited range of communications between the UAVs into the planning process.

Machine learning principles are employed in many current research papers to achieve the optimal dynamic path for the group. This can be seen in [3] as well as in [4]. Both papers use evolutionary optimization algorithm to train a path planning model over many generation. The signal range constraint is also incorporated into learning in [4].

Another approach chosen by many researchers in taking inspiration in swarm behaviour biological systems. Review article [5] calls such solutions to path planning as based on Swarm Intelligence. The article mentions examples such as Particle swarm optimization, Ant colony optimization and Artificial bee colony optimization. An example of ant colony optimization being used for path panning of a swarm can be seen in [6].

An example of a paper which considers the hardware limitations is [7] which works with the limited flight time of the UAVs. Instead of trying to explore the whole area of the mission, the introduced algorithm optimizes for search in more valuable areas to maximize gained information for a batter charge. Limited energy in also considered in [8] which takes a in-depth look at energy consumption and attempts to create an energy efficient path. The paper [4] mentioned above generates path in such way, that the connection between the UAVs stays strong enough at all times.

1.2 Problem definition

The approaches to path planning vary a lot as can be seen in Sec. 1.1. Many of the modern solutions to the problem rely on artificial intelligence which can be hard to interpret and requires significant time, computational power and resources to teach. The approach introduced in [1] uses the quadtree data structure instead, which is suitable for describing a plane area. It than uses probability to reason about position of targets in the space based on sensor readings and information gain to choose the optimal path based on gained information.

The main goal is to implement a path planning system based on the algorithm from [1] in the form of ROS nodes that can be easily added to any launch configuration in the MRS system. The implemented algorithm should be prepared for an exchange of used vision algorithm with minimal changes to the source code. This would allow the algorithm to be generalized to find any type of target. The algorithm should also work with a group of UAVs unlike the original algorithm from the paper [1] which worked with just one.

The goal of the thesis is not to create vision algorithm, instead it should use stand in targets and already implemented algorithms. Collision detection and avoidance is also not the goal of the thesis. The MRS system has its own system to handle those problems and will be relied on by the resulting algorithm implementation.

1.3 Mathematical notation

The following mathematical basis contains many reoccurring symbols which can sometimes be hard to follow. All of the notations are introduced later in the thesis with appropriate context. However the following table provides a simple place to find variable reference in case some are lost during the reading.

$\mathcal{T}$	Probabilistic quadtree instance
$\mathcal{L}(\mathcal{T})$	Leaf nodes of the quadtree
n	Node of the tree
X_n	Random variable of target being present in a area of node n
Z_n^t	Sensor reading at time t and node n
p_n, (q_n)	$P[X_n = 1]$, $(P[X_n = 0])$
$\alpha(d(n))$	Probability of false positive based on node depth $(P[Z_n = 1 \mid X_n = 0])$
$\beta(d(n))$	Probability of false negative based on node depth $(P[Z_n = 0 \mid X_n = 1])$
$H(\mathcal{T})$	Entropy of a tree $\mathcal{T}$
$I(n)$	Information gain of a node n
$\mathcal{U}(\mathcal{T})$	Stopping variable of the algorithm
C_n	Cost associated with a node n

Table 1.1: Mathematical notation, nomenclature and notable symbols.

Chapter 2

Probabilistic quadtree algorithm

The following section aims to describe and clarify an algorithm which was introduced in [1]. Firstly the entire algorithm is shown in a pseudo code and later each part is described in more detail. Sections of this chapter describe parts of the algorithm in detail. This is denoted in the beginning of each section by a reference to lines of the algorithm explained in the section.

Algorithm 1 Probabilistic quadtree algorithm as described in [1].

```
1: $\mathcal{T} \leftarrow$ InitializeTree(Prior)
2: searchDone $\leftarrow$ **false**
3: **while not** searchDone **do**
4:     $n \leftarrow \arg\max_n I'(n), n \in \mathcal{N}(\mathcal{T})$
5:     Get sensor reading $Z_n^t$ at location of $n$.
6:     Update $p_n$ and also decision for node $n$.
7:     Update $p$ and decision recursively for all ancestors and descendants of $n$.
8:     **if** $Z_n^t = 1$ **and** $n \in \mathcal{L}(\mathcal{T})$ **and** $d(n) < \mathcal{D}_{max}$ **then**
9:         Expand at node $n$ and initialize children.
10:    **end if**
11:    Compute $U(\mathcal{T})$
12:    searchDone $\leftarrow$ **true**
13:    **for all** $n \in \mathcal{L}(\mathcal{T})$ **do**
14:        **if** $-p_n log_2 p_n < \epsilon \cdot U(\mathcal{T})$ **then**
15:            searchDone $\leftarrow$ **false**
16:        **end if**
17:    **end for**
18: **end while**
19: **return** Decisions for all leaves of the tree.
```

The algorithm works in discrete steps. In each step the algorithm is selecting a node of the tree to explore by using the objective function $I'(n)$ which depends on the distance between UAV and the node and on the information which can be gained by exploring it. After gathering sensor data about the node, the algorithm than updates the tree and formulates a decision about each leaf node. If required the tree also expands at one of the leaf nodes. Than the algorithm computes stopping condition by using the value U and determines if the exploration should stop. If the exploration stops, than the decisions about each leaf node are returned.

2.1 Probabilistic quadtree data structure

The data structure used in the algorithm described in [1] is probabilistic quadtree. It is a variation of quadtree which is a tree data structure where each node has either zero or four child nodes.

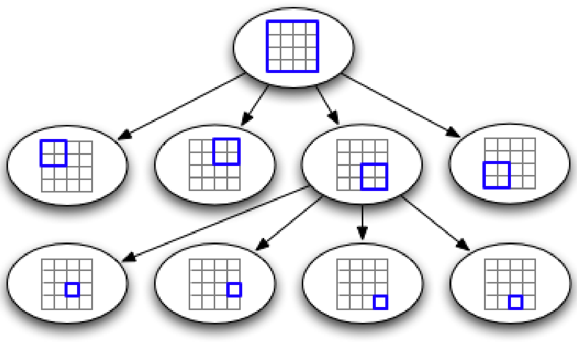


Figure 2.1: A quadtree data structure.[1]

Unlike a simple quadtree, a probabilistic quadtree also retains a probability value. Each node in a probabilistic quadtree represents square area and the posterior probability of a target being present in this area. The probability is constructed by incorporating multiple sensor readings. The mechanism of constructing the probability value is discussed in following sections.

In addition to that a node is also assigned an altitude in which a UAV has to fly in order for the onboard camera to see the entirety of the square area of the node. The altitude can be easily computed from the area size and a Field of View (FOV) of the camera. A FOV of a camera is an angle in which the camera is sensitive to light. The calculation of a flight altitude can be visualized by the following diagram.

[1]Source: https://ppujari.medium.com/quad-tree-and-spatial-data-indexing-88b12a87dfd9

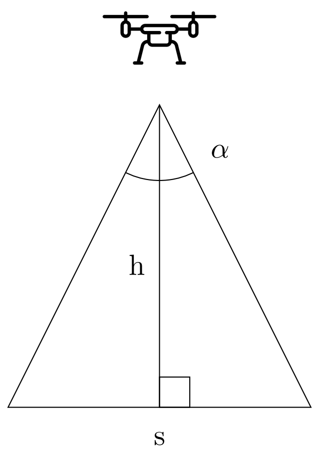


Figure 2.2: An altitude calculation for a Probabilistic quadtree node.[2]

In the Fig. 2.2 the h stands for the desired altitude, α stands for the camera FOV, usually in the shorter dimension, and s stands for the area side length. The resulting altitude can than be easily calculated using trigonometry as

$$h = \arctan(\frac{s}{2} \cdot \frac{\alpha}{2}). \tag{2.1}$$

Child nodes of a node represent equally sized areas which are created by equally dividing the area of the parent node.

Random variable X_n denotes the probability of at least one target being present in the area of node the n. For the simplicity of the notation the paper [1] proposes variables p and q of a node n which are defined as

$$p_n = P[X_n = 1],$$
$$q_n = 1 - p = P[X_n = 0].$$

We assume that the probabilities of a target being in two different areas are independent and thus

$$P[X_n = 1, X_m = 1] = p_n p_m.$$

This leads to another property between a node and its children which is derived from the fact that parent doesn't include a target if none of its children include a target:

$$q_n = q_1 q_2 q_3 q_4 \tag{2.2}$$

2.2 Detection model

A detection model needs to be defined to assume the probabilities of detecting a target under given conditions. These conditions are in our case mainly the distance between an UAV

[2]UAV image source: https://www.svgrepo.com/svg/52587/drone

and an area which is being searched. The paper [1] assumes that UAVs are limited to detecting only in the center of each area. Detection in node n in time t is denoted by random variable Z_n^t which is a Bernoulli variable which has a value of 1 only when at least one target is located in the detection area. The model also takes into consideration imperfect detection means. It assumes that each detection can be correct as well as false positive or false negative. This is denoted by variables α and β as

$$\alpha(d(n)) = P[Z_n = 1|X_n = 0],$$
$$\beta(d(n)) = P[Z_n = 0|X_n = 1].$$

As can be seen from notation, both values are dependant on the depth of the node and thus its height in a sense that detecting in higher altitude is more prone to be incorrect. *"Such detection probabilities can be determined empirically, as contained in search and rescue manuals, or from theoretical models for altitude-dependent detections"*[1].

2.2.1 Altitude dependant error functions

The paper [1] provides an article [9] as a source of a theoretical model for determining the values for detection error functions α and β. In fact the article [9] does provide a theoretical approach to the probability of successful detection based on the distance from the target. However it does not provide any specific values or formulas that could be inserted into the equations of the probabilistic quadtree algorithm. That is not an unexpected result however, as the specific values of α and β are in most cases very dependant on physical conditions. Many factors can influence the values all at once. In the example of using a camera to recognize the targets this could be weather conditions, brightness, sunlight, contrast of the searched target against the background and most importantly the vision algorithm itself. These factors will change with different detection methods and the detection environment.

For the algorithm it is however not necessarily required to obtain the perfect values for α and β. Given the usage of the values, which will be explained in the following sections, it is only necessary to model an approximate function, that will represent the real conditions well enough. This means that the resulting function should assign a higher detection error values to detections in higher altitudes. The form of the function can be chosen experimentally and the only requirement given is that

$$\alpha(d(n)) \geq \alpha(d(n')), d(n) < d(n'),$$
$$\beta(d(n)) \geq \beta(d(n')), d(n) < d(n').$$

2.3 Probabilistic quadtree updates

The following section explains in detail lines 6 and 7 of Alg. 1. During the exploration, the UAVs gather detections $Z_{1...T}$. The goal of the updates is to compute posterior probability of target being in an area $P[X_n|Z_{1..m}]$. For this computation the Bayesian inference for multiple observations is used. As follows:

$$p_n^{(t)} = \frac{P[Z_{n_t}^{(t)}|X_n = 1]p_n^{(t-1)}}{P[Z_{n_t}^{(t)}]}. \tag{2.3}$$

For the sake of clarity note that the equation (2.3) requires a computation of the value $P[Z_{nt}^{(t)}]$ which can be computed using Bayesian rule as:

$$P[Z_{nt}^{(t)}] = P[Z_{nt}^{(t)}|X_n = 1]p_n^{(t-1)} + P[Z_{nt}^{(t)}|X_n = 0]q_n^{(t-1)}. \tag{2.4}$$

The value $p_n^{(t)}$ computed by this equation can be than propagated up the tree using equation (2.2). When propagating the value down the tree the paper [1] uses following method:

$$\begin{aligned} k &= \frac{\ln q_n^{(t)}}{\ln q_n^{(t-1)}} \\ q_{n_i}^{(t)} &= (q_{n_i}^{(t-1)})^k, \end{aligned} \tag{2.5}$$

where the value k is calculated for every node going down and passed to its children.

2.3.1 Tree expansion

The following section explains in detail lines 8 to 10 of Alg. 1. In the paper [1] the tree is expanded, when the UAV has a positive detection $Z_n = 1$ while detecting in a leaf node which is not expanded yet. Also the current depth of the node has to be less than $\mathcal{D}_{max}$ which is a limit of how many layers should be in the tree. Value of $\mathcal{D}_{max}$ has to be set before the algorithm is started and can be different based on the particular mission. The main deciding factors in choosing the value of $\mathcal{D}_{max}$ is required precision in the mission since higher number of layers will lead to smaller areas in the result. On the other hand having higher number of layers generally leads to longer time required to finish the search although this time can be offset by other variables which will be discussed later. When the expansion happens, four new nodes are added to the tree with probabilities

$$q = \sqrt[4]{q_n}. \tag{2.6}$$

Expanding in this way ensures that equation (2.2) holds throughout the process.

2.4 Choosing next node to explore

The following section explains in detail line 4 of Alg. 1. There are multiple ways of choosing the next node where to check for targets. It is possible to use variety of heuristic methods which depend on the current probability of the node as well as the distance from the current node to the target one. The paper [1] shows one of purely heuristic methods which was used and showed good results. The heuristic maximises value J which is a function of both the distance from the current node in which UAV is located denoted as n' as well as the probability and the depth of the node.

$$J = \frac{p_n . 4^{d(n)}}{cost(n', n)}.$$

The paper however ends up using entropy of the node as its measure of how promising a node is. Although this is still only a heuristic, it can be argued, that using the entropy leads to more objective decision about the significance of a node. The entropy of a tree is calculated as:

$$H(\mathcal{T}) = -\sum_{n \in \mathcal{L}(\mathcal{T})} p_n \log_2 p_n, \tag{2.7}$$

where $\mathcal{T}$ stand for the tree and $\mathcal{L}(\mathcal{T})$ stands for leaves of the tree. Using similar logic the entropy can also be defined for any node of the tree in which case we get: $H(n) = -p_n \log_2 p_n$. This value is important in following sections. Another property of the node is information gain of choosing it. It is defined as

$$I(n) = H(\mathcal{T}) - E_{Z_n}[H(\mathcal{T}|n)], \tag{2.8}$$

where E_{Z_n} stands for the expected entropy over all possible detections. The algorithm for finding the optimal next node than maximises a weighted function

$$I'(n) = \gamma \frac{I(n)}{max_{n' \in \mathcal{T}} I(n')} - (1-\gamma) \frac{D(n*, n)}{max_{n' \in \mathcal{T}} D(n*, n')}, \tag{2.9}$$

where D is the distance between the UAV's current location and the node. The calculation also uses a weight γ to choose between prioritizing distance and information gain. The value of γ has to be set before the mission and can greatly influence the focus of search to either close nodes with lower focus on their information gain or even more distant nodes if they provide sufficient possible information gain.

2.4.1 Simulating a decision being made

When choosing the next node in Sec. 2.4 it is required to calculate the expected entropy $E_{Z_n}[H(\mathcal{T}|n)]$. The expected entropy in this case can be understood as the mathematical average of the tree entropy in case that the UAV detects a target and in case that the UAV does not detect a target. Both of these cases require the entropy of the tree after a detection is made. This detection however is not made on the tree structure and has to be instead only simulated. According to the definition of the entropy of a probabilistic quadtree (2.7) the entropy is only dependant on the probability p_n of it's leaf nodes $\mathcal{L}(\mathcal{T})$. Also, according to the independence in a probabilistic quadtree data structure (2.2), an update in one of the nodes of the quadtree can only affects the probability of the leaf nodes that are its direct descendants.

Thanks to these observations it is possible to declare that only descendant nodes need to be checked during the simulation of decision. The entropy of a probabilistic quadtree after a simulated detection can be calculated by calculating the update in each child node and moving the update down the tree until a leaf node is found. From a leaf node the change in entropy is noted. When the entropy change in all affected leaf nodes is gathered we can determine the entropy after a simulated detection and from that an expected entropy.

For this two approaches can be chosen. First approach simply averages the change of the entropy when the simulation assumed the detection would be negative and the one assuming a positive detection. The second approach takes into consideration probability p_n in the node n. The probability can be used as a weight of the results. When using the probability as a weight, the expected entropy is calculated as

$$E_{Z_n}[H(\mathcal{T}|n)] = p_n[H(\mathcal{T}|Z_n = 1)] + q_n[H(\mathcal{T}|Z_n = 0)]. \tag{2.10}$$

2.5 Stopping the algorithm

The following section explains in detail lines 11 to 17 of Alg. 1. The goal of any exploration algorithm is intuitively to stop the search as soon as enough information is gained to minimise the cost connected with exploring. In the paper [1] there is proposed that the gained information can be quantified by defining a variable U:

$$U(\mathcal{T}) = \frac{\sum_{n_i \in \mathcal{L}(\mathcal{T})} H(n_i)}{|\mathcal{L}(\mathcal{T})| H_{max}}, \tag{2.11}$$

where H is entropy of a node and H_{max} is maximum entropy out of all nodes in the tree $\mathcal{T}$. $L(T)$ refers to leaves in the tree in the same way as earlier.

Using obtained quantity we can decide that the algorithm should stop after all leaves of the tree have their entropy lower than $\epsilon \cdot U(T)$, where ϵ is a hyper-parameter, which can be optimised before the algorithm is started. Lowering the ϵ parameter forces the algorithm to search for longer to achieve lower entropy to satisfy the condition of $\epsilon \cdot U(T)$.

2.6 Making a decision

The following section explains in detail lines 6 and 7 of Alg. 1. For the task of formulating a decision about each leaf node, it is needed to take into consideration each type of error — missed detection and false positive. In most cases these types of error are not equally important which is why each of them is assigned individual cost. For each leaf node there is a binary decision D_n, that needs to be taken. $D_n = 0$ signalises that, according to exploration done so far, there isn't a target in the area of the node and vice versa. The cost associated with a node can be written as:

$$\begin{aligned} C_n = &P[D_n = 1|X_i = 1]P[X_i = 1]C_{11} + \\ &P[D_n = 1|X_i = 0]P[X_i = 0]C_{10} + \\ &P[D_n = 0|X_i = 1]P[X_i = 1]C_{01} + \\ &P[D_n = 0|X_i = 0]P[X_i = 0]C_{00}, \end{aligned} \tag{2.12}$$

where C_{ij} stands for the cost associated with making a decision $D_n = i$ while $X_n = j$. According to the [1] it is possible to derive a rule for choosing the decision D_n. The rule is based on the goal of minimising the cost C_n and using the form of (2.12). The rule used in [1] states that decision $D_n = 0$ should be chosen if

$$\frac{P[Z^m|X_n = 0]}{P[Z^m|X_n = 1]} > \frac{(C_{11} - C_{01})P[X_n = 1|Z^1...Z^{m-1}]}{(C_{00} - C_{10})P[X_n = 0|Z^1...Z^{m-1}]}, \tag{2.13}$$

otherwise $D_n = 1$ should be chosen. In this case $Z^1...Z^m$ stands for all sensor readings regarding the node n. $P[X_n = 1|Z^1...Z^{m-1}]$ can also be written as just p_n as is defined in the previous sections and $P[X_n = 0|Z^1...Z^{m-1}]$ can be written as q_n. The values of $P[Z^m|X_n = 0]$ and $P[Z^m|X_n]$ are obtained from the functions α and β. When update happens in a non-leaf node of the tree $\mathcal{T}$, than the decision is updated in leaf nodes when the update propagates to them from the top.

It is also necessary to derive the result in each iteration of the algorithm due to the dependence on Z^m. Deciding only at the end would force the implementation to remember the value Z^m at the time of the last update until the end of exploration.

Chapter 3

Algorithm inside MRS system

The simulation environment chosen for applying the algorithm introduced in the chapter 2 is the Multi-robot Systems Group (MRS) system built on the Robot Operating System (ROS)[1]. In this chapter the ROS environment is first introduced. Following that the MRS system is also introduced. The parts of the systems which are relevant for the correct understanding of the implementation of the Probabilistic quadtree algorithm are than highlighted and briefly explained.

3.1 Introduction to ROS

Robot Operating System (ROS) is an open-source widely used set of tools and middleware used to develop and use robots during various tasks. the ROS contains tools for building the projects, managing communication between parts of the robot, the simulation environment if the system is used in simulation or with other outside node such as a path planning algorithm. This section explains the core concepts of ROS. The concepts are ROS Nodes, Topics, Messages and Services.

3.1.1 ROS Nodes

ROS nodes are processes which are connected and communicate between each other using ROS provided means. Those means of communication with other nodes are using messages over topics or calling services provided by other nodes. In most robotic mission multiple ROS nodes are launched at the same time and communicate with each other. It is possible to view each communicating group of ROS nodes as a graph by connecting the nodes by topics and services.

3.1.2 ROS Messages

Messages in ROS serve as containers for data. They are defined in a standardized format and define exactly what types of data they can be filled with. Messages are declared in .msg files where all contained data types are described. During the building process of a ROS project the message files are compiled into usable libraries. Most notably they are compiled into Python script files containing classes and C++ libraries. After the messages have been build they can be used by ROS nodes to transfer data.

[1]https://www.ros.org/

3.1.3 ROS Topics

Topics in ROS are a way to establish a messaging connection between multiple ROS nodes. Some of the nodes can become publishers and some of them can become subscribers of data. Each node can be a subscriber to some topics and a publisher to other. When a node registers as either a publisher or a subscriber, than the ROS master node saves this information and creates a connection between each publisher and subscriber. Publishers can produce and send messages to a topic which will be received by all subscribers currently subscribing to the topic. The messages will be put into a queue and the subscriber can than receive and process them when it needs to. Each topic is generally associated with a single message type to match the type that is being sent by the publishers and that is expected by the subscribers.

3.1.4 ROS Services

Services in ROS are a system built on the ROS message architecture. They provide a way to simplify a request/response type of communication. Services are defined in .srv files in a similar way as ROS messages. Unlike messages a service contains a description of the data that is being sent and description of the data that is expected in a response. After building, the service can be provided by a ROS node. When a provider node is active, other nodes can send requests to it and receive responses.

3.2 MRS UAV system

The simulation environment chosen to perform the experiments is Multi-robot Systems Group (MRS) UAV System. It provides a high level Application programming interface (API) through the ROS message system. The system handles most of the low level management such as UAV rotor speed control, landing, takeoff, collision avoidance or hovering. It also collects data from various sensors on the UAV which it than uses to fulfil above mentioned control tasks as well as providing them through ROS topics to other ROS nodes. The sensors used are most notably camera, GPS, barometers, laser range detectors or Light Detection and Rangings (LiDARs). The next notable feature of the system is the option to receive commands through the ROS message API. Commands can be received in various forms such as coordinates or as velocity vector. The user can let the system manage the control aspect of flying the UAV and only provide the commands while using the data provided through the ROS topics.

Chapter 4

Vision algorithms

One of the goals of the thesis is to try different vision algorithms and assess their viability. First section of this chapter will describe the requirements for a vision algorithm for it to be used. It also introduces the general environment in which the algorithms will work. The following sections describe the algorithms that were tested. These algorithms are Object detect used in [10] developed by the MRS group and AprilTag developed by APRIL Robotics Laboratory at the University of Michigan. In the last section the limitations and viability of each algorithm is discussed.

4.1 General properties of the algorithms

Vision algorithms have to be compatible with the system in terms of own their own requirements and provided outputs. Compatibility is required due to the system only providing data in a given form and the path planning algorithm requiring certain information to function.

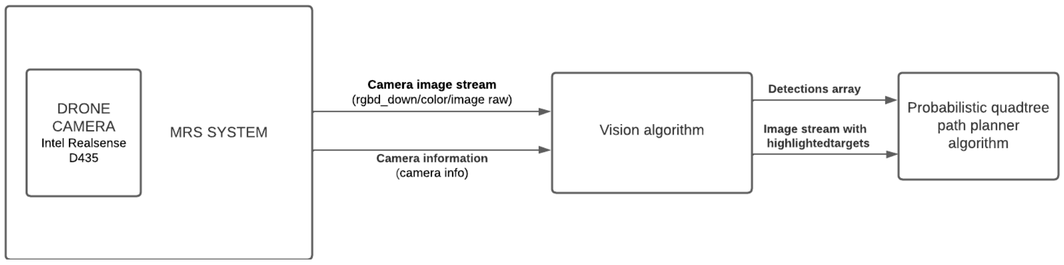


Figure 4.1: Vision algorithm general message visualization.

All vision algorithms require a video stream to function. UAVs in the experiments in this thesis are using a single downward facing Intel Realsense D435 camera. The video of the camera is than available through ROS topic provided by the MRS system. This image stream can than be used by the detection algorithms to evaluate the presence of targets in the currently visible area. Another data source which can be used by detection algorithms is camera info topic which can provide information about camera image dimensions, distortion and other general camera parameters. The algorithm can than use this information to better determine if the image contains a target.

Vision algorithms also need to provide output topics to which the path planning algorithm can subscribe and than use them to find the path for the exploration. The main output topic of a vision algorithm are detections. This topic should contain an array of detected targets as well as their position relative to the UAV. The other output topic should be image stream topic containing camera video with highlighted detections. The second output as well as position information of target relative to UAV is however not strictly necessary for the correct functioning of the path planning algorithm of the implementation and can be used mainly for visualization.

4.2 Object detect

The first used vision algorithm is object_detect[1] package of Multi-robot Systems Group of Faculty of Electrical Engineering of Czech Technical University in Prague (CTU). It is a segmentation algorithm. The algorithm was originally used for recognizing targets in a robotics competition MBZIRC where the targets were balloons suspended in the air.[10] The algorithm is capable of recognizing spherical objects of a single color using camera data.

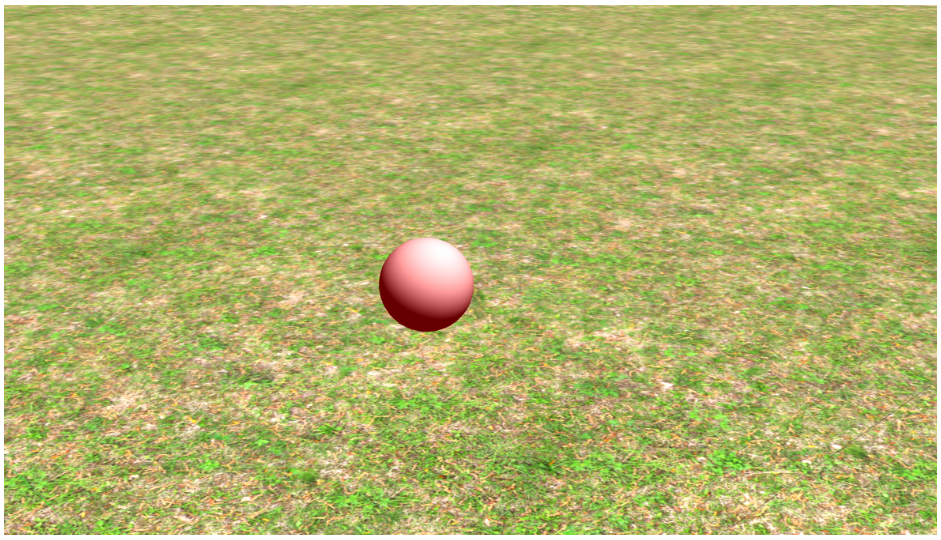


Figure 4.2: An example of a sphere used as target for Object detect.

The package allows the user to configure the size and the color of the targets which allows the algorithm to generalize to all colors and sizes of spheres. In addition to recognizing and segmenting the target in a video, the package is also capable of estimating the distance and relative position of the target. The detection of targets is however sufficient output for the purposes of this thesis and such the position approximation is not used.

[1]https://github.com/ctu-mrs/object_detect

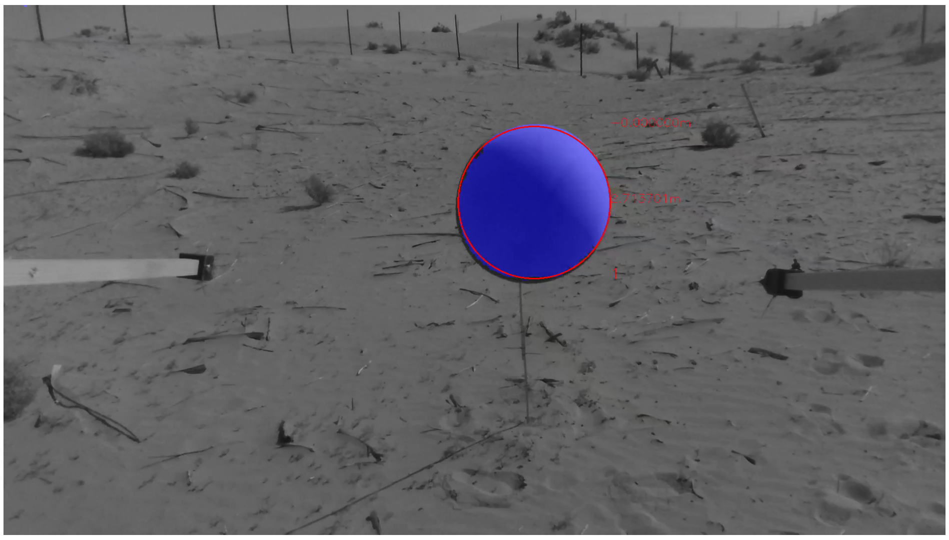
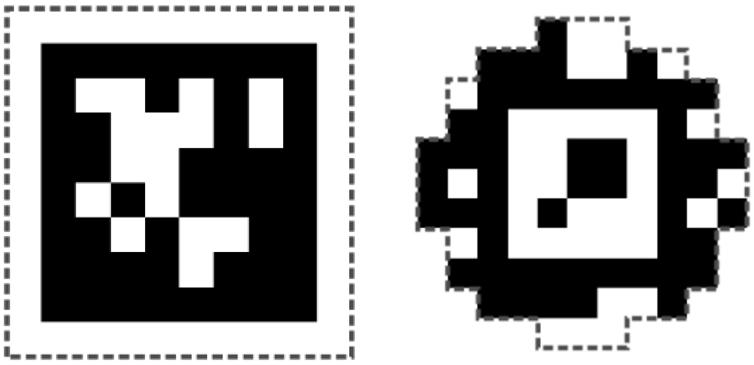


Figure 4.3: A sphere segmented out of the picture.[2]

4.3 AprilTag

The second algorithm used is AprilTag[11]. The detection targets of the AprilTag algorithm are special 2D tags. Tags look similar to well known QR tags used for example as links to websites or for an easy transfer of payment details. The tags can be of multiple visual families but the set used for a given mission has to be of only one family. Tag families differ in the shape and the general structure of the tags. For the experiments a Tag36h11 family was used, but any other family of tags could have been used almost interchangeably.



(a) Tag36h11 family (b) TagCircle49h12 family

Figure 4.4: An example of a tag used by AprilTag vision algorithm.[3]

Furthermore each tag in the set has to be unique for the algorithm to work. This requirement comes from the ability to distinguish between multiple tags during flight. This is redundant for given use case as a placeholder for the simulation but it could be used if the

[2]Source: https://github.com/ctu-mrs/object_detect#readme
[3]Source: https://april.eecs.umich.edu/software/apriltag

task was modified to not only find, but also distinguish the targets. Simulating for AprilTag is however quite easy given availability of the tags or even models of the tags for used Gazebo simulator.

AprilTag can however not be used well for general purpose tasks due to the dependence on a very specific target. This does not limit the usage in the implementation of the thesis given that the thesis is not a real usage but rather an evaluation of the correctness of the used approach on placeholder targets in the simulation.

4.4 Viability and limitations

Following section is giving an evaluation of each detection system when considering the particular goal of the thesis implementation. Both algorithms fulfil the requirements for a viable vision algorithm established in 4.1. Each of them however requires different detection target which has to considered when creating the experiment environments in the simulator or preparing experiments in the physical environment.

Main difference between the algorithms can be found in their detection capabilities. AprilTag performs much better in terms of being able to recognise target located further from the UAV's camera. Object detect fails to recognise the target reliably when the UAVs is located more than 10 meters above the ground in the simulation. AprilTag can still recognize the target reliably even when the UAV reaches altitude of around 30 meters. This observation has to be considered when preparing the experiments and evaluating the results of the experiments in the Results chapter.

Chapter 5

Implementation and expansion

5.1 Project structure and ROS communication

In Sec. 1.2 it was established, that the main goal of the project is to implement a path planning algorithm based on the Probabilistic quadtree algorithm defined in [1]. This section explains in detail the used parts of the system and explains the interaction among them.

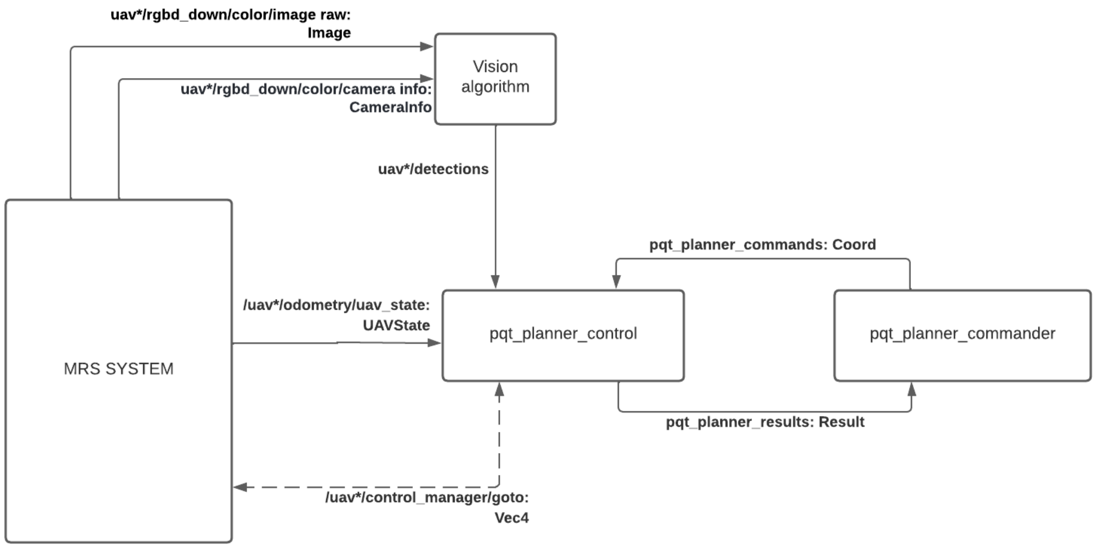

uav*/rgbd_down/color/image raw: Image
uav*/rgbd_down/color/camera info: CameraInfo
Vision algorithm
uav*/detections
pqt_planner_commands: Coord
/uav*/odometry/uav_state: UAVState
pqt_planner_control
pqt_planner_commander
MRS SYSTEM
pqt_planner_results: Result
/uav*/control_manager/goto: Vec4

Figure 5.1: Communication using ROS topics and services.

The path planning algorithm that is the goal of the implementation is divided into two parts. Each part of the system takes form of a ROS node.

First of the parts is the Commander node. It hold the instance of a Probabilistic quadtree data structure and handles the updates of the structure. It is also responsible for deciding about the next node that should be explored, checking for the stopping condition and once that is reached it outputs the resulting decisions into a predefined file.

Second part of the system is a Control node. It serves as a communication bridge between the Probabilistic quadtree path planning algorithm and the MRS system. The main purpose in that regard is managing the movement of the group of controlled UAVs. For that a location of each of the UAVs is obtained from the MRS system in form of a UAV state. The UAV state message contains, among other data, a position of the UAV as a set of coordinates given a reference point. The data for the UAV state is obtained by the MRS system by fusion of results of various position estimation hardware such as GPS, laser altitude estimator, barometer and more. The second step in managing the position of UAVs is changing their location. For that a goto service of control manager package of the MRS system is used. Goto service allows the Control node to issue a service requests using the Vec4 ROS service type which contains the desired position of the aircraft. In case the vehicle is able to fulfil the requirement, the service gives back a positive response message. In some cases such as ongoing takeoff or landing operation the UAV is not able to accept goto calling and the service call returns a negative response in that case. The control node has to make sure that the UAV control manager responded positive to the calling and repeat the calling until it is successful otherwise.

The last major responsibility of the Control node is receiving detection data from Vision algorithm of choice and evaluating the presence of targets from incoming data. The particular form of the received detection data depends on the vision algorithm used as the interface of each of them is not standardised.

The two aforementioned nodes communicate between each other using two ROS topics. First of them is published by the Commander node and contains the next locations for each manged UAV to explore. The message type used is Coord which contains the coordinates of the next node in reference to given base point and the number of the UAV which should fly there. The second used topic is published by the Control node which transmits through it the results of the detection in the location to which the UAV was sent. The messages in this topic are of a type Result and contain the number of the UAV and a boolean flag that signals if there was at least one target detected.

5.2 Using multiple UAVs

The expansion of the algorithm introduced in [1] is one of the main objectives of the thesis as the original algorithm was not originally introduced for a scenario with multiple UAVs. The MRS system provides a simple tools for introducing multiple UAVs to the environment at the same time using the drone spawner service. Each UAV has its own sensors and control inputs which are exposed through topics as was mentioned in 3.2.

Modification that has to be made is the access to the Probabilistic quadtree data structure instance. There are two ways to how to approach this challenge. It is possible to distribute the data structure between all UAVs and synchronize it using common distributed systems means or to have a central structure accessed by all the UAVs. The decision for the thesis was made to rely on single central instance of the Probabilistic quadtree data structure mainly due to distributed system not being the topic of the thesis and due to possibility to use the system in this way, since all the experiments are done in the simulator. However in a real scenario with large enough area it might not be possible to ensure a stable connection of all UAVs to a central control node or with each other. In such case the distributed system is required and means of ensuring its validity at all times have to be considered and implemented.

The last part of the multi UAV problem that is discussed in this section are collisions between the UAVs and ways of preventing them. Again there are multiple ways how to handle this problem. First of the solutions is provided by the MRS system. The UAVs are equipped with a collision detection system that takes control in case the UAVs are about to collide and stops them or diverts them. This approach was chosen for the implementation due to this problem not being the main topic of the thesis, having a low implementation complexity and good results.

A fully custom system for collision avoidance could be made since the system has full information about the positions of the UAVs at all times as well as the information about the their desired locations. A preemptive system could also be designed which would use the knowledge of a target coordinates for each UAV at a given time and the knowledge, that UAVs will try to get to the point in a straight line. The system than could insert intermediate waypoints ahead of time to make sure the paths of the drones never cross.

A more experimental solution based on modifying the objective function $I(n)$ declared in Sec. 2.4 is also a possibility for improving the performance. The function would be enhanced by another part, that would incorporate distance from already occupied nodes. The modified objective function would than become

$$I'(n) = \gamma \frac{I(n)}{max_{n' \in \mathcal{T}} I(n')} - (1-\gamma) \frac{D(n*, n)}{max_{n' \in \mathcal{T}} D(n*, n')} - \delta min_{n'' \in O(\mathcal{T})} D(n, n''), \tag{5.1}$$

where $O(\mathcal{T})$ stands for all nodes in the tree $\mathcal{T}$ that are currently occupied. This would also introduce another hyper parameter δ to optimize and in theory this would lead to choosing the nodes further from the nodes that are already being explored, thus minimizing the collision avoidance precautions. The modification would however need proper mathematical background as well as an intensive testing which was not done as part of this thesis.

5.3 Baseline method implementation

In the thesis goals it is stated that the Probabilistic quadtree algorithm implementation should be tested against a baseline method. The method that is chosen for that purpose is a simple zig zag pattern.

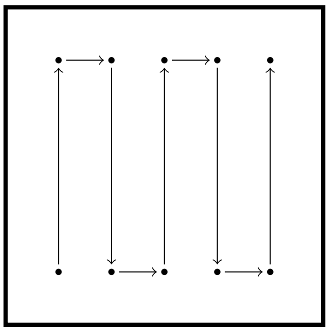


Figure 5.2: An example path of the baseline method.

As can be seen in Fig. 5.2 the final generated path consists of two rows of coordinate points situated on two opposite sides of the explored rectangular area. Than a simple pattern is applied to the points to create the final path.

The benefit of using this type of path is that it covers the entire area of the explored rectangle. For this it is however needed to properly choose the flight altitude so that the camera mounted on the UAV can see each part of the explored area. The calculation of the correct height is quite simple and was already described in Sec. 2.1. The calculation is exactly the same and uses trigonometric equation (2.1) with value that can be seen in Fig. 2.2.

To create a trajectory for the UAV from generated waypoints the Trajectory generation package[1] of the MRS is used. The package provides a ROS service and a ROS topic on each

[1]https://github.com/ctu-mrs/mrs_uav_trajectory_generation

drone in the simulation. The package generates an optimal trajectory when a Path message or a Path service, both containing an array of waypoints, is sent to the UAV. Once a trajectory is generated, the package takes care of managing the controls of the UAV and makes it fly along the trajectory.

Chapter 6

Experiments

Now that all necessary parts of the algorithm and the implementation are explained, it is possible to assess the performance of the implementation. This chapter is describing the methodology of the experiments and the way that they are evaluated. A used configuration is also described in this chapter.

6.1 Methodology of the experiments

All of the experiments share a few key properties. First of them is that all of the test are realized inside the Gazebo simulator inside ROS. All of the experiments share the same goal of finding a set of targets with randomly generated positions in a simple environment. The form of the target depends on the vision algorithm used in the experiment. The differences between the experiments are the size of the search area. Each experiment is performed multiple times with changing variables which are discussed later. Goal of each experiment is to gather results from the search mission, determine correctness of the gathered results and the errors. Than the results are compared with the results of the same experiment with different variables.

The first variable in the experiments is using two different searching algorithms. The algorithms are Probabilistic quadtree algorithm which is the subject of this thesis and in detail explained in chapter 1. The other is a baseline zig zag pattern algorithm explained in Sec. 5.3. The baseline method and the comparison strategy used are mentioned later in the chapter. The other variable is the vision algorithm used. As was discussed in chapter 4 there were two algorithms used: AprilTag and Object detect.

6.2 Gathering results

During the experiments, results are gathered from the algorithm and evaluated at the end. The Probabilistic quadtree algorithm outputs decisions in subsections of the initial search area. Gathered decisions from the Probabilistic quadtree are compared with the initial positions and sizes of the targets. Evaluation of each decision can be one of three cases. It can be a correct decision, a false positive, which means that the algorithm decides that there is a target in an area where no targets are present, or a false negative, when there is a negative decision in an area where there is at least one target.

6.3 Baseline method search results

To evaluate the viability of the area exploration using the Probabilistic quadtree algorithm a simple baseline method is used. The path generated by such algorithm is simple, however if the flying height is configured correctly, it does guarantee that the entire search area will be searched during a single flight. The mentioned correct configuration is further described together with the general implementation of such system in Sec. 5.3. Considering this guarantee it is possible to assume that an UAV that is flying slow enough along the given path will be able to detect each of the targets.

Although it would be possible to incorporate a vision algorithm to the baseline method, it is not necessary. It is sufficient to use the total duration of the mission to compare the performance of the baseline method to the performance of the Probabilistic quadtree algorithm.

It is worth noting that other measures could be used such as the total distance traveled. Getting the total distance traveled from the simulation run is on the other hand way more complicated and though it would provide values less dependant on for example the used simulation hardware, in the end it is not worth.

6.4 Used configuration

For the experiments a single configuration of the hyper-parameters was chosen. Note that the configuration of the parameters has a major influence on the course of the experiment and consequently on its result. This section is going through the list and assigning a value to each of the used hyper-parameters. Also a few changes had to be made in the way a next node is chosen in order to solve issues related to the vision algorithms used, the simulation environment and the case of using multiple UAVs. Those changes also have influence on the experiment and solving the issues in a different way can lead to a different results in the performed experiments.

6.4.1 Hyper-parameters

The Probabilistic quadtree algorithm as was introduced in [1] contains many hyper-parameters which have to be chosen before the search experiment is started. First of the parameters are error functions α and β introduced in Sec. 2.2. A function

$$f(x) = \frac{1}{x + 1.3}$$

was used for both of these variables. The function was chosen in this form to satisfy conditions for error functions of increasing error when depth decreases and also because it worked well in the experiments. The function is however not chosen by proper optimisation as it could be and the usage of different function can lead to significantly different results.

Next hyper-parameter is γ introduced in 2.4 which determines the weight of the information gain as opposed to the distance when choosing next node for exploration. The value was chosen as 0.5 same as in the paper [1] to give the same weight to the distance and the information gain.

Following hyper-parameter is ϵ used in Sec. 2.5. The usage of the parameter is to determine the required exploration necessary to end the search mission. A value of 1 was arbitrarily chosen for this parameter.

The costs of decisions C from Sec. 2.6 were set to the same values as in the paper [1]. This means that C_{00} and C_{11} for correct decisions were set to 1, C_{10} for false positive was set to 10 and C_{01} for missed target was set to 1000.

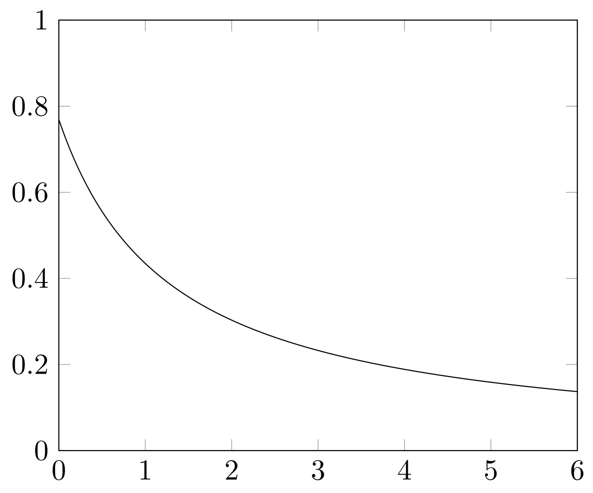


Figure 6.1: Graph of the used error function.

The last parameters that have to be determined before the start are the initial and the maximal depth of the tree and the initial probability. The initial probability was set to 0.5 in all leaf nodes to indicate that no information is available at the beginning of the exploration about the targets. The depths were set for each experiment differently depending on the size of the area. In every experiment the maximal depth was set in such way that the detections in deepest level of the tree reliably recognized the targets. The initial depth was in most experiments set to 1 to minimize explorations of unnecessary parts of the target area.

6.4.2 Node choosing modifications

In Sec. 2.4 a way of choosing next node is introduced. The introduced mechanism allows any node to be chosen at any time. In case the algorithm is using multiple UAVs however a node that is already about to be explored by a UAV can not be chosen due to the obvious collision between the UAVs. Also a detection in such node would not provide much benefit.

The next modification that was necessary solves the issue of the limitations of the Object detect vision algorithm that are presented in Sec. 4.4. The limitation of only reliably detecting in altitudes up to 10 meters can be solved by limiting the next node choice to only nodes located under that altitude. This leads to necessity to preemptively expand the tree ahead of time to create nodes that satisfy this new condition. This however limits the range in very significant extent and can lead to reduction of positive characteristic of the algorithm to expand only the parts of the tree that have a positive detections in them.

The last modification that had to be made is caused by the safety area limitation of the simulation. This limits the UAVs from flying above 30 meters in the simulation. Although the limitation can be removed, during the experiments it was present and had to be considered. The solution is the same as to the limitation described above concerning Object detect. This change however has only a small influence as only small number of nodes are located above the 30 meter boundary given the sizes of area in the experiments.

Chapter 7

Results

The following chapter presents achieved results and measurements of the realized experiments. All of the results are also discussed and a possible explanation of the results is provided. Other than that an evaluation of how successful was the implementation of the Probabilistic quadtree algorithm for area exploration. The success is discussed in terms of the implementation showing the positive characteristics of the algorithm as well as evaluating the results of using multiple UAVs.

7.1 Comparison to simple algorithms

As was established in Sec. 6.3 the Probabilistic quadtree algorithm was compared with baseline algorithm based on time. The results are following:

Algorithm used	10x10m	20x20m	30x30m	40x40m
Baseline	60s	112s	202s	352s
Single Object detect	213s	95s*	449s*	1056s*
Single AprilTag	148s	283s	460s	870s
3 UAV Object detect	151s	165s*	375*	834s*
3 UAV AprilTag	107s	424?s	371s	361s

Table 7.1: Achieved exploration times in the experiments.

It has to be noted, that some of the experiments had to be made under modified conditions from the other ones. The experiments marked with asterisk had tree expanded ahead of time to a greater depth due to requiring at least some of the nodes to be associated with a low enough altitude as was discussed in Sec. 6.4.2. This was done to solve problems with detection limitations of the Object detect algorithm. This combined with maximal depth chosen, so that the UAV is high enough to see the entire target located in the area, caused some of the experiments to be limited to only one or two tree layers for available nodes in which case the beneficial properties of the Probabilistic quadtree exploration algorithm are unfortunately lost.

From the table 7.1 it is possible to see, that the experiments using the baseline exploration method finished faster than the experiments using the Probabilistic quadtree method. On the other hand note, that the time difference gets shorter when the explored area increases. It is possible to imagine, that if the area increased even further, than the Probabilistic quadtree approach would surpass the baseline method.

The behaviour is anticipated since the main advantage of the Probabilistic quadtree approach is using the tree structure to minimize unnecessary exploration. The effect of using such approach should become more noticeable with increasing the size of search area as the approach is able to prune larger areas at a time.

7.2 Implementation performance

The performance of the implementation in experiments is measured in the number of correct and incorrect decisions. The results are following:

Algorithm used	10x10m	20x20m	30x30m	40x40m
Single Object detect	5/0/4	5/0/7	5/0/4	5/0/3
Single AprilTag	5/0/4	5/0/4	5/0/4	5/0/4
3 UAV Object detect	5/0/7	5/0/7	5/0/4	5/0/6
3 UAV AprilTag	5/0/0	5/0/4	5/0/4	5/0/16

Results order: Correct/Missed targets/False positives

Table 7.2: Performance of the approach in experiments.

It can be seen that the implementation performs very well. The algorithm was able to find all of the targets in all of the experiments. The presence of false positives in the experiments can be explained by usage of the unoptimized hyper-parameter mainly costs of errors C. There is a possibility that if optimized values were used, the algorithm would not make any mistakes at all. Note that some of the results of the Object detect can be biased, as the tree had to be limited.

7.3 Implementation evaluation

The implementation of the Probabilistic quadtree area exploration algorithm is successful. During a run of an experiment the tree stays in the correct state, properly updates and expands based on the detections made by UAVs. The performance can not be easily evaluated as it depends in many ways on the chosen hyper-parameters and the vision algorithm. The implementation also manages to overcome limitations of the vision algorithms, although in some cases that means giving up on tree structure which is able to skip the exploration of some part which are not likely to include targets.

Testing the implementation showed obstacles of the Probabilistic quadtree approach to area exploration. The idea behind using the tree structure stands on the ability to decrease the altitude to increase the detection capabilities in exchange for a lower detection area. This property however did not align with the characteristics of the vision algorithms used. The used algorithms are very good at recognizing targets up to certain distance from it and than the detection ability falls off very quickly. Ideally the Probabilistic quadtree algorithm would require a vision algorithm, that is able to recognize targets at many different distances.

7.3.1 Usage of multiple UAVs

The usage of multiple UAVs was proven to be beneficial to the speed of the exploration as can be seen in table 7.1. The extend of the speedup is however in many ways limited by using only basic precautions to prevent collisions which in many cases are not optimal. Not all information about the UAVs which is available to the path planning algorithm is used. The ways of improving the approach and incorporating available information into path planning is discussed in Sec. 5.2.

Chapter 8

Conclusion

This thesis achieved all of the goals set in the beginning. The paper [1] was thoroughly studied and the ideas were explained in detail in the chapter 2. The chapter also included the explanation of the topic mentioned in the paper [1] which were not explained and had to be interpreted. This includes error functions and the calculation of expected entropy.

The second goal was also achieved. The MRS UAV System and also the Robot Operating System (ROS) were introduced in the chapter 3. All of the important parts of these systems were explained in sufficient detail to understand their usage in the thesis implementation.

The Probabilistic quadtree algorithm was implemented using the MRS UAV System and was also connected to two vision algorithms — the Object detect and the AprilTag. The implementation also considered the limitations of the vision algorithms introduced in the chapter 4 and adapted the algorithm to overcome them.

Original idea of using the Probabilistic quadtree structure was extended to allow usage of multiple UAVs at the same time using a centralized node. The collision detection and prevention system was left to be handled by internal mechanisms of MRS UAV system and the goto service inside of it.

In the end the implementation was tested in a variety of different experiments using both of the vision algorithms. The exploration times were compared with the times a naive pre-planned algorithm.

An obstacle caused by used vision algorithms not satisfying the expected characteristics of a vision algorithm used in the Probabilistic quadtree was recognized and described. The approach could be evaluated differently if a proper vision algorithm is used.

Another obstacle of the algorithm is reliance on many hyper-parameters. An optimization of this amount of parameters can be hard and would require a complex optimization algorithm. In ideal case the parameters could be tested outside the simulation which would allow for much faster optimization. This was however in the time of thesis creation not possible.

Expanding the implementation to multiple UAVs decreased the time required to finish the mission, however in many cases the improvement was held back by the constant collision avoidance measures being used. A future work could expand the idea by more efficiently dispersing the UAVs across the search area to minimize possible collisions and improve the performance of the entire approach.

Chapter 9

References

[1] S. Carpin, D. Burch, and T. H. Chung, "Searching for multiple targets using probabilistic quadtrees," pp. 4536–4543, 2011.

[2] T. Baca, M. Petrlik, M. Vrba, *et al.*, "The MRS UAV System: Pushing the Frontiers of Reproducible Research, Real-world Deployment, and Education with Autonomous Unmanned Aerial Vehicles," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 26, pp. 1–28, 1 May 2021.

[3] M. G. Cimino, M. Lega, M. Monaco, and G. Vaglini, "Adaptive exploration of a uavs swarm for distributed targets detection and tracking.," in *ICPRAM*, 2019, pp. 837–844.

[4] L. Ruetten, P. A. Regis, D. Feil-Seifer, and S. Sengupta, "Area-optimized uav swarm network for search and rescue operations," in *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, 2020, pp. 0613–0618.

[5] A. Sharma, S. Shoval, A. Sharma, and J. K. Pandey, "Path planning for multiple targets interception by the swarm of uavs based on swarm intelligence algorithms: A review," *IETE Technical Review*, pp. 1–23, 2021.

[6] S. Konatowski and P. Pawłowski, "Ant colony optimization algorithm for uav path planning," in *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)*, 2018, pp. 177–182.

[7] H. Song, J. Yu, J. Qiu, *et al.*, "Multi-uav coverage planning with limited endurance in disaster environment," *CoRR*, vol. abs/2201.10150, 2022. arXiv: `2201.10150`. [Online]. Available: `https://arxiv.org/abs/2201.10150`.

[8] M. Monwar, O. Semiari, and W. Saad, "Optimized path planning for inspection by unmanned aerial vehicles swarm with energy constraints," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[9] B. O. Koopman, "The theory of search. ii. target detection," English, *Operations research*, vol. 4, no. 5, pp. 503–531, 1956.

[10] Y. Stasinchuk, M. Vrba, M. Petrlík, *et al.*, "A multi-uav system for detection and elimination of multiple targets," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 555–561.

[11] J. Wang and E. Olson, "Apriltag 2: Efficient and robust fiducial detection," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 4193–4198.