Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2022

Probability Based Path Planning of Unmanned Ground Vehicles for Autonomous Surveillance

– Through World Decomposition and Modelling of Target Distribution

Per Liljeström

LINKÖPING UNIVERSITY

Master of Science Thesis in Electrical Engineering

Probability Based Path Planning of Unmanned Ground Vehicles for Autonomous Surveillance: – Through World Decomposition and Modelling of Target Distribution

Per Liljeström

LiTH-ISY-EX--22/5459--SE

Supervisor: **Forskningsingenjör Jonas Nordlöf**
FOI
Förste forskare David Lindgren
FOI

Examiner: **Associate Professor Daniel Axehill**
ISY, Linköpings universitet

Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

To my family!

Abstract

The interest in autonomous surveillance has increased due to advances in autonomous systems and sensor theory. This thesis is a preliminary study of the cooperation between UGVs and stationary sensors when monitoring a dedicated area. The primary focus is the path planning of a UGV for different initial intrusion alarms. Cell decomposition, *i.e.*, spatial partitioning, of the area of surveillance was utilized, and the objective function is based on the probability of a present intruder in each cell. These probabilities were modeled through two different methods: *ExpPlanner*, utilizing an exponential decay function. *Markov planner*, utilizing a Markov chain to propagate the probabilities. The performance of both methods improves when a confident alarm system is utilized. By prioritizing the direction of the planned paths, the performances improved further. The Markov planner outperforms the ExpPlanner in finding a randomly walking intruder. The ExpPlanner is suitable for passive surveillance, and the Markov planner is suitable for ”aggressive target hunting”.

Acknowledgments

First and foremost, I would like to thank the Swedish Defence Research Agency, FOI, for the opportunity to write this thesis. A special thanks to my supervisors, David Lindgren at FOI and Jonas Nordlöf at FOI/Linköping University, for their patience, guidance, and constructive feedback. I also want to thank my examiner Daniel Axehill at Linköping University, for his proofreading and feedback. Finally, I thank my family for their continuous support and my friends for making my time at Linköping University the adventure it has been.

Linköping, 2022
Per Liljeström

Contents

Notation **xi**

1 Introduction **1**
1.1 Background & motivation . . . 1
1.2 Purpose . . . 3
1.3 Problem formulation . . . 3
1.4 Criteria . . . 4
1.5 Assumptions & delimitations . . . 4

2 Theory **5**
2.1 Discrete path planning . . . 5
2.1.1 Receding horizon control . . . 6
2.1.2 Search algorithms . . . 6
2.2 Informative path planning . . . 8
2.3 Model – Spacial and informative partition . . . 8
2.3.1 Holonomic and nonholonomic systems . . . 9
2.3.2 Spatial partition . . . 9
2.3.3 Quantifying information . . . 10
2.3.4 Graph configuration . . . 10
2.4 Exponential decay . . . 11
2.5 Markov processes . . . 12

3 Method **19**
3.1 Scene description . . . 20
3.1.1 Scene decomposition and probabilities . . . 20
3.1.2 Nodes and information handling . . . 21
3.1.3 Obstacle handling . . . 23
3.2 Target distribution . . . 25
3.2.1 Alarm system . . . 25
3.2.2 ExpPlanner, Exponential-based planner . . . 26
3.2.3 Markov planner . . . 28
3.2.4 Target's random walk . . . 32
3.3 Graph search . . . 33

3.3.1 Search method . . . 33
3.3.2 Search tree . . . 34
3.4 Objective function . . . 36
3.4.1 Failure probability . . . 36
3.4.2 Correction factor . . . 36
3.4.3 Algorithms . . . 38
3.5 Simulation environment . . . 40
3.6 Path planner evaluation . . . 41
3.6.1 Monte-Carlo simulations . . . 41
3.6.2 Evaluation of robustness . . . 41
3.6.3 Computation time . . . 41
3.6.4 Evaluation . . . 42

4 Results **45**
4.1 The time constant $\mathcal{T}$ . . . 45
4.2 Sparse obstacle placement . . . 47
4.2.1 Confident alarm system . . . 48
4.2.2 Uncertain alarm system . . . 54
4.2.3 Perimeter alarm system . . . 55
4.3 Simulations . . . 61
4.3.1 ExpPlanner, simulation . . . 61
4.3.2 Markov planner, simulation . . . 65

5 Discussion **69**
5.1 Results . . . 69
5.1.1 The time constant $\mathcal{T}$ . . . 69
5.1.2 Confident alarm system . . . 69
5.1.3 Uncertain alarm system . . . 70
5.1.4 Perimeter alarm system . . . 71
5.1.5 Comparison of alarm systems . . . 72
5.1.6 Comparison of planners . . . 73
5.2 Method . . . 73
5.2.1 Source criticism . . . 74

6 Conclusions **75**
6.1 ExpPlanner . . . 75
6.2 Markov planner . . . 75
6.3 Objective function . . . 76
6.4 General conclusions . . . 76
6.5 Future work . . . 77

A Tables **81**
A.1 Uncertain alarm system . . . 82

Bibliography **85**

Notation

Sets

Notation	Meaning
$\mathbb{X}$	Set of all conceivable states, i.e. a generic state space
$\mathbb{X}_{free}$	Set of all known states located within the *free*, unoccupied space
$\mathbb{X}_{occ}$	Set of all *known* invalid states located within obstacles or other invalid space
$\mathbb{U}(x_(t))$	Set of all possible actions from state $x(t)$
$\mathbb{U}$	All feasible actions over all states
$\mathbb{K}$	Index set of known, predefined, *free* nodes
$\mathbb{S}$	Index set of known, predefined *free* information regions
$\mathbb{S}_{x(t)}$	Index set of all *free* information regions observable from state $x(t)$
$\mathbb{S}_{perim}$	Index set of all *free* information regions along the perimeter of scene
$\mathbb{S}_{max}$	Index set of information region(-s) with maximum target probability
$\mathbb{S}_{+}$	Index set of all *free* information regions with non-zero probabilities
$\mathbb{M}_{x(t)}$	Index set of potential succeeding states that the UGV can move to from state $x(t)$
$\mathbb{H}_{x(t)}$	Set of possible paths from state $x(t)$, i.e. the whole search tree from state $x(t)$
$\mathbb{G}_{i}$	Index set of permitted subsequent information regions from region i
S	The theoretical Markovian state space

Symbols, functions & operators

Notation	Meaning
$x(t)$	The UGV's state at time t
$\tilde{x}(t)$	The UGV's intermediate state after the probability update but before the motion update at time t
$\varphi(t)$	The UGV's position, or coordinates, at time t
c_i	The coordinates to the centroid of region i
$u(t)$	An action that changes the UGV's state $x(t)$
$f_{next}(x(t), u(t))$	The motion update function, used in updating the UGV's state
$\gamma(t)$	The region for which the target is located at time t
$\lvert\mathbb{Z}\rvert$	The cardinality of set $\mathbb{Z}$, *i.e.*, the size of set $\mathbb{Z}$
N	Number of *free* information regions, *i.e.*, the cardinality of set $\mathbb{S}$
N_p	Number of *free perimeter* information regions, *i.e.*, the cardinality of set $\mathbb{S}_{perim}$
K	Number of *free* nodes, *i.e.*, the cardinality of set $\mathbb{K}$
$p_i(t)$, $p_i^{\dagger}(t)$	The probability of target being in information region i at time t, *prior resp. posterior* to observation
$\mathbf{p}(t)$, $\mathbf{p}^{\dagger}(t)$	The target distribution at time t, *prior resp. posterior* observation
$\mathbf{v}_k, \mathbf{v}_i$	Position vector to node k *resp.* centroid of region i
p_{max}	Index of the region with maximum target probability
Γ	The probability update algorithm
λ_i	The last time information region i was observed
$\mathcal{T}$	Time constant in seconds used as a tuning parameter
f_s	The samplings frequency in the number of steps the UGV takes per second
T_s	The sampling time in seconds between the UGV's steps
h	A path the UGV can move along, defined as a finite sequence of states
$h(t)$	A state in path h at time, or index t
m	The prediction horizon of each path h
$\mathcal{F}(h)$	The failure probability of path h
$V(h)$	The objective function
$h^{\star}(t)$	The optimal path for the chosen horizon
r	The detection radius of the UGV's ambiguous sensor configuration
$\mathcal{P}$	The transition probability matrix, or the transition matrix
$\mathbf{X}^{\intercal}$, $\mathbf{x}^{\intercal}$	Transpose of matrix $\mathbf{X}$ or vector $\mathbf{x}$
X_{ij}	Is the element at row i and column j in matrix $\mathbf{X}$
$gdc(\mathbb{Z})$	The greatest common divisor of the elements in set $\mathbb{Z}$

Acronyms & Abbreviation

Acronyms	Meaning
IPP	Informative path planning
RRT	Rapidly exploring random tree
UGV	Unmanned ground vehicle
4CG	Four-connected graph
FOI	The Swedish Defence Research Agency
sv.	The Swedish translation
AMÖS	*Autonom övervakning med samverkande sensorer* – a FOI-project
RHC	Receding horizon control
CPU	Central processing unit
FAB	Forward and backward

1 Introduction

The primary focus of this master's thesis is the development, implementation, and evaluation of path planners, with an appropriate objective function, collaborating with stationary alarm systems for the considered application of *informative path planning* (IPP) [1]. The problem is to plan a path for an unmanned ground vehicle (UGV) tasked to search for an intruder in an area of surveillance. This chapter presents the basis of this project done in collaboration with the *Swedish Defence Research Agency* (FOI) and begins with a brief background and motivation in Section 1.1. The purpose is presented in Section 1.2. The problem is formulated in Section 1.3 and the evaluation criteria are presented in section 1.4. Finally, the assumptions and delimitations used are presented in Section 1.5.

1.1 Background & motivation

The last couple of decades have seen evident progress in the research and development of autonomous platforms, *e.g.*, self-driving cars [2]. This progress can not only be seen in the civil sectors but also the military sectors, with the increased investments in the research and development of unmanned vehicles for military usage on an international scale [3].

The advantages of using UGV's for surveillance are numerous; one prominent advantage is removing direct danger for patrolling security personnel or soldiers in a military scenario. Deployed military personnel can focus on other important tasks during missions instead of having multiple personnel patrol, *e.g.*, the surroundings of an encampment. The UGV could be used to monitor vital installations (*sv.* skyddsobjekt) against possible intruders. Thus, it is necessary that

the UGV's path planning is robust and reliable. The UGV should monitor the dedicated area efficiently while simultaneously avoiding obstacles. The performance of the UGV should not vary much.

Research on the design of multi-sensor surveillance systems of collaborative mobile and stationary units has been conducted at FOI, in the autonomous surveillance project, *Autonom övervakning med samverkande sensorer* (AMÖS) [4]. The multi-sensor surveillance systems were designed to autonomously detect, classify and pursue threats, such as intruders, in a militarily challenging surveillance scenario. AMÖS examined, *e.g.*, different positioning methods for aerial multi-sensor systems and developed methods to detect and track moving objects, among other things.

AMÖS inspired this thesis to investigate how a path planner for an arbitrary UGV collaborating with a stationary alarm system should be designed. The certainty of alarm systems, *i.e.*, their ability to locate an intrusion, varies between alarm systems in general. Thus, alarm systems with different certainty are evaluated. The alarm systems provide the path planner with initial knowledge of the intruder's location. Subsequently, it is proposed that the intruder's location is updated through different probabilistic models. These models are designed depending on the established knowledge of the intruder's movement. Thus, a probabilistic objective function is a suitable choice. The models are: The *ExpPlanner*, modeling an exponential decay of the target distribution. The *Markov planner*, modeling the target distribution based on Markov chains.

The fundamental principles of Bayesian Search presented in [5] are another inspiration. In particular, the *one-sided search problem*, *search theory*, and the usage of Markov models to describe motion. The differences between [5] and this work are centered around the search method and the representation of the *target probability density*. The related theories are presented in Sections 2.2 and 2.3.3.

The notion of representing target position through Markovian states has been done before [6]. The idea of minimizing some function of the probability of not observing a target was also studied in [6], albeit not the same way as this thesis. [6] utilizes an iterative *Forward And Backward* (FAB) to compute the optimal search plan when the target's motion is modeled by a discrete space and time Markov chain.

In the case of monitoring the area of a permanent vital installation, the area's layout is predefined, *i.e.*, the positions of the obstacles are known. In the case of monitoring the perimeter around a temporary encampment, the area is most likely unknown to the UGV. Therefore, the UGV will need to explore the area of surveillance in addition to monitoring it, *i.e.*, the positions of the obstacles are unknown. Based on these two predominant cases, the "mission" is divided into different levels according to its complexity:

Level 1 – Monitoring the area of surveillance without obstacles.

Level 2 – Monitoring the area of surveillance with known obstacles that do not block the UGV's line of sight.

Level 3 – Monitoring the area of surveillance with known obstacles that block the UGV's line of sight.

Level 4 – Monitoring the area of surveillance with unknown obstacles that block the UGV's line of sight.

This thesis study only the first three levels of mission complexity.

As previously mentioned, the thesis's primary focus is the development, implementation, and evaluation of path planners collaborating with stationary alarm systems. Thus, the details of the UGV's controller were omitted.

1.2 Purpose

This master's thesis is done in collaboration with FOI in Linköping and can be seen as a continuation of AMÖS [4]. The purpose of this thesis is to serve as a possible initial basis for ambiguous IPP implementations and the UGV-aspect of FOI's continued evaluation and development of autonomous surveillance.

1.3 Problem formulation

Alongside the main purpose previously mentioned, the following problem statements will be answered:

- How should the path planner for an autonomous UGV be designed to ensure a sufficiently robust and efficient behavior when monitoring different (complex) scenarios?
  - How should the path planner's objective function be designed to ensure a sufficiently robust and efficient behavior?
- Which of the examined methods results in the highest likelihood to detect an intruder (or similar measurement) for the different examined complex scenarios?
  - Are there any improvements to the performance by introducing knowledge of the target's movement?

These problem statements will be answered by evaluating and comparing some criteria specified in Section 1.4 below.

1.4 Criteria

The possible criteria used to evaluate the produced path planners are listed below:

- The mean time it takes to detect an intruder and its standard deviation.
- Robustness: How consistent is the method's performance, *e.g.*, how common are outliers.
- The method's computation time.

1.5 Assumptions & delimitations

The following assumptions are made to simplify the circumstances. The simplification is due to the limited time.

- The UGV's position is known.
- The UGV will move on a flat surface, *i.e.*, a two-dimensional plane.
- The environment configuration for which the UGV is deployed guarantees motion without slip and no external interference.
- The specification of the sensor components used by the UGV is left ambiguous. The UGV has a 360° two-dimensional field of view, with a fixed radius: a detection radius, denoted r. The resulting circle centered on the UGV is called the detection area. The UGV's sensor configuration can detect obstacles and intruders within the detection area.
- The UGV's kinematic properties are that of a holonomic rigid body.
- There is at most one mobile target with an unknown location present in the area of surveillance.
- The individual probabilities of the target distribution are independent, and the developed methods are based on these.

2
Theory

The following chapter presents the theory on which this master's thesis is based. Different quantities, notations, and terminology used throughout the master's thesis are presented.

The chapter begins with an introduction to discrete path planning in Section 2.1, followed by the informative path planning (IPP) in Section 2.2. These are fundamental for the thesis, which is a form of application of discrete IPP. Section 2.3 presents the theory related to the developed model's spatial and informative partitioning. Section 2.4 presents the theoretical basis of the ExpPlanner. The ExpPlanner is used as a baseline for the final *Markov planner*. The theory related to the Markov planner is relayed in Section 2.5.

2.1 Discrete path planning

The considered systems throughout the thesis are discrete both in time and space. A straightforward approach to describe the basis of many planning problems is the discrete feasible planning model that uses state-space models [7, Chapter 2.1]. Different planning models could be described through reasonable extensions of the discrete feasible planning model. Each unique configuration of the "world" is called a *state*, denoted x, and *all* conceivable states form the set $\mathbb{X}$, called *state space*. The obtained *state space* must be countable – each state can be assigned a unique integer – for the discrete planning model to be feasible. It is also essential to ensure that only relevant information is included when defining the state space. Failing in doing so could lead to a solvable planning problem converting to an unsolvable one. $\mathbb{X}$ needs to be large enough to ensure that the task at hand is solvable. $\mathbb{X}$ is comprised of *the free state space*, $\mathbb{X}_{free}$ and *the occupied state*

space, $\mathbb{X}_{occ}$, *i.e.*,

$$\mathbb{X}_{free} \subseteq \mathbb{X},\ \mathbb{X}_{occ} \subset \mathbb{X} \Rightarrow \mathbb{X} = \mathbb{X}_{free} \cup \mathbb{X}_{occ}. \tag{2.1}$$

By applying *actions*, denoted u, the planner can transform the configuration of the "world". This configuration change can be seen as a new state $x(t+1)$ is generated from the current state $x(t)$ through a so-called *state transition function* f_{next}. Here, f_{next} is used as a *motion update function*. This motion update is done through the *motion update equation*:

$$x(t+1) = f_{next}(x(t), u(t)). \tag{2.2}$$

All actions applicable from state $x(t)$ form the *action space* for each state $x(t)$ denoted $\mathbb{U}(x(t))$ and $u(t) \in \mathbb{U}(x(t))$. The same action may be applicable in multiple consecutive states, *i.e.*, for specific $x(t)$, $x(t+1) \in \mathbb{X}$; $\mathbb{U}(x(t))$ and $\mathbb{U}(x(t+1))$ are not necessarily disjoint, $\mathbb{U}(x(t)) \cap \mathbb{U}(x(t+1)) \neq \emptyset$. Hence, it is beneficial to define the set $\mathbb{U}$, formed from all possible actions over all states:

$$\mathbb{U} = \bigcup_{x(t) \in \mathbb{X}} \mathbb{U}(x(t)). \tag{2.3}$$

Furthermore, a set of goal states $\mathbb{X}_{goal} \subset \mathbb{X}_{free}$ is often defined for planning problems. The planning algorithm is tasked to find a finite sequence of actions that transform the initial state x_0 to the goal to some goal state $\mathbb{X}_{goal}$.

2.1.1 Receding horizon control

Receding horizon control (RHC) is a common control strategy for industrial applications and is based on optimal control. RHC is widely acknowledged as a successful and applicable control theory in academia [8]. The fundamental concept of RHC revolves around a fixed finite prediction horizon m. At each time instance t, the optimal control sequence for the time interval $[t,\ t + m\,T_s]$ is obtained, where T_s is the sample time. Only the first control signal in the optimal control sequence is utilized as the control signal at time t, the rest is discarded. This is repeated for the subsequent time instances, *e.g.*, for the succeeding time instance $t + T_s$ is the first control signal from the optimal control sequence on the time interval $[t + T_s,\ t + (1 + m)\,T_s]$ used as the control signal at time $t + T_s$ [8].

2.1.2 Search algorithms

Graph search algorithms are commonly used when searching for feasible plans in a 2D grid. A common form of a grid graph in 2D is the "four-connected grid graph" (4CG) and is presented in Figure 2.1. A finite 4CG is known as a "square grid graph" [9].

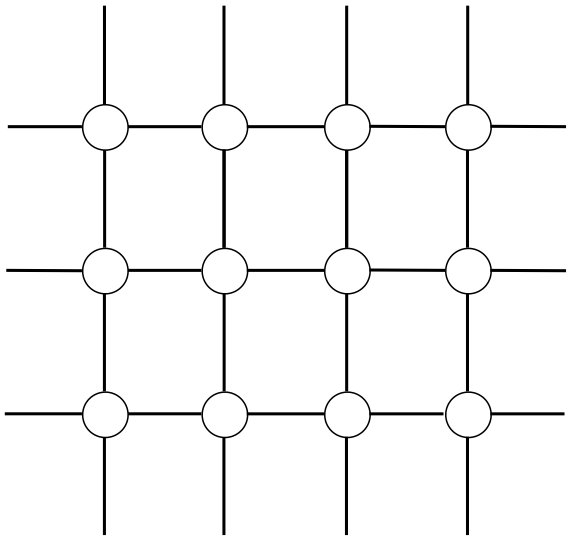


Figure 2.1: *An illustration of the connectivity between the* discrete points *of the 4CG.*

An important requirement for any search algorithm is to be *systematic*, including the search algorithms presented in this section below. Finite graphs are restricted either by a perimeter or a search horizon, *e.g.*, RHC. For finite graphs, the requirement of *systematic* search algorithms is that the algorithm will visit every reachable state while keeping track of visited states. It is sufficient to ensure that no redundant exploration is done for a search algorithm to be *systematic*. The following search algorithms are the two examined search algorithms that worked as a basis for the final search algorithm used in this thesis. These are both forward search type, with different sorting functions for the used queue q [7].

Breadth-first – First-In, First-Out

This search algorithm sorts q according to the first-come-first-served principle, often called *First-In, First-Out* (*FIFO*) queue. The breadth-first approach results in a uniform expansion, *i.e.*, all plans of depth k must be examined before any plan of depth $k + 1$ is examined [7].

Depth-first – Last-In, First-Out

This search algorithm sorts q according to the last-come-first-served principle, often called *Last-In, First-Out* (*FIFO*) queue. The depth-first approach results in an aggressive graph search; that quickly expands straight for the depth of the graph [7].

2.2 Informative path planning

When addressing the problem of planning suitable paths for autonomous vehicles dedicated to surveillance, it is a fundamentally sound approach to plan a path that yields the most ”information” about the monitored area and potential targets within. The problem of finding the path that yields the most ”information gain” is called *informative path planning* (IPP) [1]. It can be viewed as a subclass of the traditional motion planning problem, only with a different type of performance criteria. IPP seeks the path that would yield the most informative measurements, contrary to traditional motion planning, which usually plans towards a goal state or a set of goal states while continuously optimizing an objective function, *e.g.*, the shortest path to the goal states [10]. The prospective gain in knowledge or ”information” depends on what is already known or collected, *i.e.*, IPP makes path-related decisions based on the prediction of future ”information gain” from measurements [1].

Bayesian search

The Bayesian search in [5] uses a gradient search method (using gradient descent), and different particle filters represent the target probability density. In this thesis, graph search algorithms are utilized and the target probability distribution is represented as probabilities distributed between predefined discrete regions, as described in Section 2.3.

One-sided search problem is a category of search problems where only the searcher (here: the UGV) has a strategy while the target does not, *i.e.*, the target does not react to the search. Search theory aims to optimize the distribution of limited resources during a search for a target, given prior knowledge of the distribution of the target's position [5]. Planning based on maximizing the probability of detecting a target is comparable to IPP if the ”information gain” of IPP is chosen as the update of the (prior) target probability distribution after an observation, see Section 2.3.3.

2.3 Model – Spacial and informative partition

This section describes the model used as a foundation throughout this thesis. *Configuration space* is the state space for motion planning, and contrary to the state space $\mathbb{X}$ for discrete path planning, described in Section 2.1, the *configuration space* is a set of possible transformations applicable on the UGV [7]. The concept of holonomic and nonholonomic systems are presented in Section 2.3.1, followed by the theoretical basis of spatial partitioning or cell decomposition in Section 2.3.2. Theory concerning the definition and quantification of the information measurement is briefly mentioned in Section 2.3.3. Finally, in Section 2.3.4 is the essential graph theory relaid.

2.3.1 Holonomic and nonholonomic systems

A *robotic mechanism* is defined as a composition of perfectly rigid bodies connected ideally between themselves at *joints*. When describing the kinematics of a robotic mechanism, one common approach is to describe the *joint kinematics*, *i.e.*, the freedom of motion between bodies within the robotic mechanism. A joint where the connected bodies constrict their relative motion is called a *kinematic joint*. This constraint of relative motion (usually) stems from the connected bodies' geometric properties. The majority of these constraints associated with the *kinematic joints* can be expressed as equations of only the joint position variables, called *holonomic constraints*. Constraints that cannot be expressed solely with the joint position variables but also incorporate the time derivative of at least one of the position variables are called *nonholonomic constraints* [11, Chapter 2.3]. Concisely, the robotic mechanism is under holonomic constraints if its orientation is not considered. Thus, the robot can move in whichever direction; it is omnidirectional.

2.3.2 Spatial partition

One common approach to solve motion planning problems when the occupied *configuration space* is polygonal is cell decomposition [12–14]. *Cell decomposition* is defined as the partitioning of the "space", *i.e.*, the area of surveillance, into non-overlapping cells [7, Chapter 6.3]. "Suitable" cell decomposition must satisfy three criteria:

1. It should be trivially easy to connect two separated points within a cell. *E.g.*, with every cell being convex, a straight line can be drawn between any two points within each cell, with the whole line segment still within the cell.
2. Information about each cell's neighbors, *i.e.*, the cells' adjacency information, must be easily extracted to build a roadmap. A *roadmap* is defined as a graph with a sufficient number of paths such that any motion planning query is efficiently solvable.
3. Given two points ρ_1 and ρ_2, there exists an efficient way to determine which cells contain them.

If these criteria are satisfied, the cell decomposition reduces the motion planning problem to a graph search problem between the partitioned cells [7, Chapter 6.2.2]. After partitioning the surveillance area into cells, a suitable path, *e.g.*, a coverage path or a simple discrete path between measuring locations within each cell, can be computed to manage the internal travel or movement within each cell [12, 15].

A version of cell decomposition in 2D is a uniformly spatial partition of the world into non-overlapping, square cells. The difference comes in the usage of the partition. Contrary to how cell decomposition simplifies a motion planning problem to a graph search problem, spatial partitioning can also simplify the information gathering; by discretizing the measurements of the information quantities to a

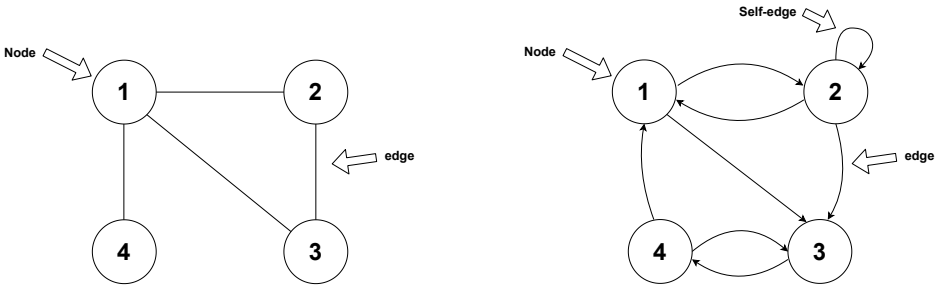
single value for each cell and approximating a complete collection of all information within the cell if the cell's centroid is observed, described in Section 3.1.

2.3.3 Quantifying information

As previously mentioned in Section 2.2, IPP revolves around finding the path that yields the most "information gain". The general convention of "information measurement" for IPP is the information matrix $\mathcal{I}$, which is the inverse of the covariance matrix of sensor measurements [10]. An alternative to this is simply the combination of the probability of a present target and cell decomposition. By modeling the probability of a present target distributed between the cells, an approximation of the target's location is obtained and used as information measurements.

2.3.4 Graph configuration

Network science treats how entities are connected. These connections are typically represented as graphs. The graphs are made up of nodes connected to other nodes through a link or *edge* [16]. Figure 2.2 shows examples of an undirected and a directed graph presented.



(a) Undirected graph *(b) Directed graph*

Figure 2.2: *Example of an undirected graph and a directed graph with basic terminology.*

Definitions

Definitions related to network science and graphs are presented below [16].

Definition 2.1 (Adjacency matrix). Consider a system with N nodes. Then the systems adjacency matrix, denoted $\mathbf{A}$, is an N-by-N matrix with its elements defined as;

$$A_{ji} = \begin{cases} 1, & \text{if there is a connection from node } j \text{ to node } i \\ 0, & \text{otherwise} \end{cases}. \tag{2.4}$$

For undirected graphs the matrix $\mathbf{A}$ is symmetric; *i.e.*, $\mathbf{A}^\intercal = \mathbf{A}$.

Note: The convention of $\mathbf{A}$ varies between literature. Here, *row node* j points to *column node* i.

The two graph examples in Figure 2.2 have the following adjacency matrices:

$$(a) \quad \mathbf{A}_a = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \qquad (b) \quad \mathbf{A}_b = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}.$$

Note that for undirected graphs, the elements in the main diagonal of the adjacency matrix $\mathbf{A}$ are zero. This characteristic is also true for directed graphs without self-edges. The elements in $\mathbf{A}$'s main diagonal are A_{ii}, $i = 1, .., N$, *i.e.*, *node* i *points to node* i, as per (2.4).

Definition 2.2 (Strongly connected component). A group of nodes within a directed graph that fulfills the requirement that any node is reachable from any other node by following the directions of the edges are called a *strongly connected component*.

Definition 2.3 (Irreducible adjacency matrix). If the whole directed graph forms a strongly connected component, the adjacency matrix $\mathbf{A}$ is said to be *irreducible*.

Definition 2.4 (Absorbing node). A node with no outgoing edges is called an *absorbing node*.

The directed graph in Figure 2.2b is also a *strongly connected component* and thus is $\mathbf{A}_b$ *irreducible*. Edges can have an associated weight attached to them. These weights could symbolize a cost to traverse that specific edge. Weights are often depicted as a (numeric) value beside the associated edge in the graph [16].

2.4 Exponential decay

A common approach when processing and handling temporal data is deteriorating the importance of the collected data based on its age, *i.e.*, how much time has passed since it was recorded, using a *decay function* [17]. One popular decay function is the exponential decay

$$f(a) = e^{(-\beta a)}, \tag{2.5}$$

where a is the data's age and β is the decay rate. With the mean lifetime of the data as τ, then $\beta = 1/\tau$ [17].

2.5 Markov processes

In modern probability theory, it is well-established that previous outcomes influence the prediction of future events and should be considered in most real-world cases for stochastic processes. Markov processes are fundamental in understanding stochastic processes and the relation between past and future events [18]. Markov processes are used in various fields, *e.g.*, meteorology, DNA analysis, and economic phenomena [18–21].

A Markov process is, in fact, a stochastic process that fulfills the *Markov property*, described in Definition 2.5 [22].

Definition 2.5 (Markov property). The Markov property declares that if the state of a stochastic process at a given time instance t is known, then its past and future states, relative to this time instance t, are independent. That is; if the state $x(t)$ of the process is known at a given time instance t, then the prediction of its future state in regards to time instance t, denoted $x(t+1)$, does not change due to any additional information about the past states $x(\tilde{t})$, $\tilde{t} < t$.

The distinction between Markov chains and Markov processes is incoherent in the studied literature, and it seems to be an established flaw within the field [23, 24]. What is clear is that *Markov chains* are defined as a discrete version of the Markov process. The vague distinction is whether the time [24] or the state space (index set) [20, 22] is discretized. There is no particular agreed-on restriction on the considered state space. Thus, the state space can be ambiguous, but a finite or countably infinite state space is usually used. This thesis consider a *countable finite* state space. A discrete-time Markov chain on a countable (finite or countably infinite) state space is defined in Definition 2.6.

Definition 2.6 (Discrete-time Markov chain). A *discrete-time Markov chain* is defined as a stochastic process, *i.e.*, a sequence of stochastic variables: $\{X_t,\ t \in \mathbb{N}_0\}$ on a *countable* state space S, henceforth known as the *Markovian state space*. For $s_i \in S, \forall i \in \{0, 1, ..., t-1, t\}$ and $t \geq 1$ it holds that:

$$Pr(X_t = s_t \mid X_{t-1} = s_{t-1}, ..., X_0 = s_0) = Pr(X_t = s_t \mid X_{t-1} = s_{t-1}),$$

where $Pr(X_t = s_t \mid X_{t-1} = s_{t-1})$ is the *transition probability – conditional probability* – from state s_{t-1} to s_t, in accordance with Definition 2.5.

Transition probability matrix

The probability notation for Markov chains is condensed through the use of a *transition probability matrix*, denoted $\mathcal{P}$, defined in Definition 2.7 [18].

Definition 2.7 (Transition probability matrix). The transition probability matrix, $\mathcal{P}$, also known as the *transition matrix*, is a square matrix with the number of rows and columns equal to the cardinality (number of elements), $|S|$, of the *Markovian state space* S. Each element is a non-negative real number representing the *conditional* probability to transition from one state to another. The sum of the transition probabilities from a state i to all states (including itself) must be 1. $\mathcal{P}$ is also called the stochastic matrix or the Markov matrix.

Transition matrices occur as three different types [18];
(1) A *right* transition matrix, with the elements $Pr(X_t = j \mid X_{t-1} = i) = \mathcal{P}_{ij},\ i, j \in S$. $\mathcal{P}_{ij}$ is the transition probability from state i to state j. The requirement that the sum of the transition probabilities from state i equals one results in the sum of each row equal to one,

$$\sum_{j \in S} \mathcal{P}_{ij} = 1,\ \forall i \in S.$$

(2) A *left* transition matrix is the transpose of the *right* transition matrix.

(3) A *doubly* transition matrix is a square matrix where the sum of both rows and columns equals 1.

Probability vector

The more common convention in the literature concerning stochastic processes – or more specific Markov chains – of the previously mentioned three is the right transition matrix accompanied with a *stochastic vector*; or *probability vector* $\mathbf{p}(t) = [p_1(t), p_2(t), ..., p_{|S|}(t)]$. $\mathbf{p}(t)$ is a non-negative real row vector, with its elements summing to one, that depict the probability distribution of the *Markovian states*, *i.e.*, it specifies the probability that the system (Markov chain) is in each *Markovian state* [18, 22, 25, 26].

As the sum of the probability vector equals one, the following holds,

$$\sum_{i=1}^{|S|} p_i(t) = 1,\ \forall t. \tag{2.6}$$

The update of $\mathbf{p}$ occurs each time instance according to:

$$\mathbf{p}(t+1) = \mathbf{p}(t)\,\mathcal{P}. \tag{2.7}$$

State diagram

It is common to use state diagrams to visualize a Markov chain (stochastic process) [18, Chapter 1.3]. There are some similarities between state diagrams and graphs, as previously mentioned in Section 2.3.4. Figure 2.3 portrays an example of a state diagram for a two-state Markov chain, for which the transition matrix is given in (2.8).

$$\mathcal{P} = \begin{array}{c} \\ A \\ B \end{array} \begin{array}{c} \begin{array}{cc} A & B \end{array} \\ \begin{bmatrix} 0.3 & 0.7 \\ 0.8 & 0.2 \end{bmatrix} \end{array}. \tag{2.8}$$

Markovian states are represented as ”nodes”, and the transition probabilities between Markovian states are represented through arrows – with accompanied probability – between the Markovian states. Note the added labels for the rows and columns of the transition matrix (2.8); these denote the two states in Figure 2.3.

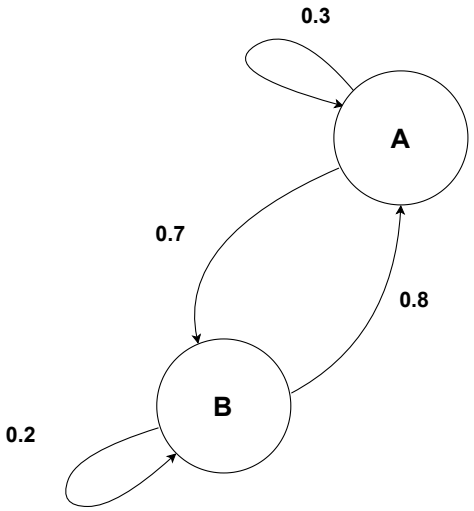


Figure 2.3: *State diagram for a two-state Markov Chain. The nodes representing the Markovian states are depicted as circles, and the arrows depict the transition probabilities.*

To describe Markov chains, some fundamental properties of Markov chains need to be defined. Subsequent definition is used to describe the Markov chain utilized by the Markov planner.

Homogeneous Markov chains

One time-based property of Markov chains is *homogeneity*. The (time-)*homogeneous* Markov chain is defined in Definition 2.8 [25].

Definition 2.8 (Homogeneous Markov chains). A discrete-time Markov chain, $\{X_t, t \in \mathbb{N}_0\}$, defined as per Definition 2.6, is said to be (time-)*homogeneous*, if for $\tau_1 < \tau_2 < ... < \tau_r < \tau < \tau + n$ the probability:

$$Pr(\, X_{\tau+n} = s \mid X_{\tau_1}, X_{\tau_2}, ..., X_{\tau_r}, X_\tau) = Pr(\, X_{\tau+n} = s \mid X_\tau), \text{ where } s \in S \text{ and } n \in \mathbb{N}$$

depend only on the time step n and not on the current time t.

Thus, for *homogeneous* Markov chains, a transition from one given state to another during a unit of time $n = 1$ does not depend on the time, only between which two states the transition is done. Furthermore, the transition probability matrix can be constant, $\mathcal{P}(t) = \mathcal{P}$. This property of time-invariance of $\mathcal{P}$ for homogeneous chains facilitates the computation of the n-step transition probability matrix as the n-th power of the transition matrix, $\mathcal{P}^n$ [25].

Irreducible

A Markov chain can be irreducible as per Definition 2.9 [25].

Definition 2.9 (Irreducible Markov chain). If all states in a Markov chain are reachable from any other state, the Markov chain is said to be irreducible.

Consequently, the related state diagram, which can be seen as a graph, is a *strongly connected component*, as per Definition 2.2, in its whole. Then the associated transition probability matrix $\mathcal{P}$ is *irreducible*, as per Definition 2.3 [22].

Recurrence and transience

Let $\tau(j)$ denote the number of transitions needed to arrive at Markovian state j, where $\tau(j)$ is defined as

$$\tau(j) = \inf\{n \geq 1 \mid X_n = j\},$$

where inf is the infimum and $\tau(j) = \infty$ if the set is empty. The probability that the first visit to state j starting in state i occurs after n transitions, denoted $f_{i,j}^{(n)}$, is defined $\forall i, j \in S$ and $n \geq 1$ as

$$f_{i,j}^{(n)} = Pr(\, \tau(j) = n \mid X_0 = i) = Pr(\, X_n = j, X_k \neq j, 1 \leq k \leq n-1 \mid X_0 = i). \quad (2.9)$$

Naturally, $f_{i,i}^{(n)}$ denote the probability that the first revisit of state i, from state i, occurs after n transitions. Consequently, $\forall i, j \in S$ and $\forall n \geq 1$, it holds that

$$(\mathcal{P}^n)_{ij} = Pr(\, X_n = j \mid X_0 = i) = \sum_{k=1}^{n} f_{i,j}^{(k)} \, (\mathcal{P}^{n-k})_{jj}.$$

Let $f_{i,j}$ denote the probability that arriving at state j from state i occurs in a finite number of transitions and is defined as

$$f_{i,j} = Pr(\, \tau(j) < \infty \mid X_0 = i) = \sum_{k=1}^{\infty} f_{i,j}^{(k)}. \tag{2.10}$$

The properties recurrent and transient are related to whether it is possible to revisit a Markovian state or not. The two properties are defined in Definition 2.10 and Definition 2.11 below [22].

Definition 2.10 (Recurrent state). A Markovian state $i \in S$ is said to be *recurrent* if, starting from i, the probability that the subsequent return to i occurs at a *finite time*, $f_{i,i} = 1$. In other words, the Markovian state i is revisited an *infinite* number of times when time $t \to \infty$.

The Markovian state i is *recurrent* if and only if:

$$\sum_{n=1}^{\infty} (\mathcal{P}^n)_{ii} = \infty \Rightarrow \lim_{n\to\infty} (\mathcal{P}^n)_{ii} \neq 0. \tag{2.11}$$

Definition 2.11 (Transient state). A Markovian state $i \in S$ is said to be *transient* if, starting from i, the probability of returning to i at a *finite time*, $f_{i,i} < 1$. In other words, the Markovian state i is revisited a *finite* number of times when time $t \to \infty$.

If the Markovian state j is *transient* then, $\forall i \in S$,

$$\sum_{n=1}^{\infty} (\mathcal{P}^n)_{ij} < \infty \Rightarrow \lim_{n\to\infty} (\mathcal{P}^n)_{ij} = 0. \tag{2.12}$$

If all Markovian states for a Markov chain are recurrent, respectively transient, then the Markov chain is said to be recurrent, respectively transient, itself.

For a recurrent Markovian state, the first return time will be a stochastic variable called the *recurrence time*. The *mean recurrent time* for state i is

$$\mu_i = \sum_{n=1}^{\infty} n \, f_{i,i}^{(n)}, \tag{2.13}$$

and if the μ_i is finite, then state i is said to be *positive-recurrent* [25].

Periodicity

The periodicity of a *Markovian state* is closely related to recurrence and is defined in Definition 2.12 [22].

Definition 2.12 (Period of Markovian state). Markovian state i has the period

$$d(i) = gdc(n > 1 \mid (\mathcal{P}^n)_{ii} > 0), \tag{2.14}$$

where $gdc(X)$ is the greatest common divisor of X. $d(i)$ is essentially how often it is possible to return to state i.

Note: $(\mathcal{P}^n)_{i,i} = 0 \Rightarrow d(i) = 0,\ \forall n \geq 1$.

State i is said to be *aperiodic* if $d(i) = 1$. If all states in a Markov chain are aperiodic, then the Markov chain is said to be aperiodic.

If all Markovian states in a Markov chain $X = \{X_i, i \in S\}$ have "self-edges"; $\mathcal{P}_{ii} \neq 0,\ \forall i \in S$, the Markov chain is then aperiodic.

Ergodicity

Ergodic processes include or visit all states in the state space given an adequate amount of time, enabling a statistical representation of the process through a fairly large collection of states [27]. Ergodicity from a Markovian perspective is defined in Definition 2.13 [25].

Definition 2.13 (Markovian ergodicity). A Markovian state is said to be ergodic if it is *aperiodic*, Definition 2.12, and *positive-recurrent*, (2.13).

If *all* states in an irreducible Markov chain are ergodic, is the whole Markov chain said to be ergodic.

Stationary Markov chains

Ergodic finite Markov chains have a unique probability vector π called *equilibrium distribution* that denotes the limitations on the probability distribution and is defined in Definition 2.14 [25].

Definition 2.14 (Equilibrium (stationary) distribution). For an ergodic Markov chain, the *equilibrium distribution* π is defined by

$$\lim_{n\to\infty} \mathcal{P}^n = \begin{pmatrix} \pi \\ \pi \\ \vdots \\ \pi \end{pmatrix}$$

π is the *stationary distribution* that a homogeneous (finite) Markov chain converges to independently of the initial conditions, such as the initial probability

distribution $\mathbf{p}(t = 0)$. If the initial probability distribution is π, then the probability distribution for all the subsequent time instances remain π, *i.e.*, $\pi \mathcal{P} = \pi \Rightarrow \pi \mathcal{P}^n = \pi$.

Random walk

Random walk is just that, a walk where each step is taken in a random direction. Random walk is one of the most straightforward stochastic processes out there. Assume a person initially is located in state X_0. X_0 can be seen as a physical point. At each time instance t, the person takes a step U_t, where $\{U_t, t \in \mathbb{N}\}$ is a stochastic process of mutually independent, identically distributed stochastic variables. A Random walk that satisfies the Markov property in Definition 2.5 is a Markov chain [25].

One version of Random walk utilizes a uniform distribution of the feasible steps from each state X_i, $\forall i \in \mathbb{N}_0$, *i.e.*, all feasible steps from each state are equally likely.

3
Method

This chapter describes the considered scenario and the approach used to develop the objective functions for two different path planners and the associated search algorithms. Important quantities, notations, and terms are distinguished through *emphasis*. The developed method can be partitioned into six main components:

- Section 3.1: The description of the scenario and the configuration of the simulation environment.
- Section 3.2: The different approaches to representing the probability distribution of a present intruder. These representations are the main difference between the two examined models.
- Section 3.3: The considered search methods for operating the UGV and the expanded search tree are presented.
- Section 3.4: The developed objective function that is mutually used as the foundation for the path search by both models. The two fundamental algorithms for traversal and path search are outlined.
- Section 3.5: A short description of the developed simulation environment used for evaluation and presentation.
- Section 3.6: The methods used to evaluate the different path planners are presented.

The UGV's and the surveillance area's configurations are kept as ambiguous as possible to accommodate a broader range of different implementations and sensor platforms.

3.1 Scene description

The following sections present the configuration of the simulation environment and the premise upon which this thesis builds.

3.1.1 Scene decomposition and probabilities

The area of surveillance, henceforth known as the *scene*, is assumed to be known before deployment. The scene is uniformly decomposed (partitioned) into N non-overlapping cells, henceforth known as *information regions*, or simply *regions*, see Figure 3.1. The partitioning is done before deployment as the scene's dimensions are known. Each region is assigned an index $i \in \mathbb{S}$, where $\mathbb{S}$ is the index set of known, predefined, *free* regions with the size $|\mathbb{S}| = N$. A *free* region is defined as a region with an unoccupied centroid.

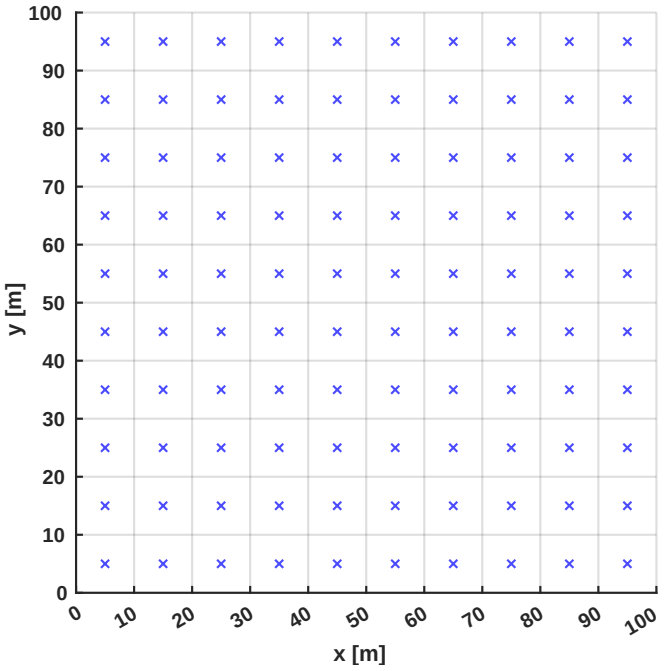


Figure 3.1: *An example of the uniform decomposition of the* scene. *The example depicts a decomposition with:* $N_x = N_y = 10 \Rightarrow N = N_x \times N_y = 100$. *The dimension of each region is thus* 10×10 *meters. The blue, transparent crosses represent each* information region's *centroid.*

It is assumed that if the UGV observes the centroid of an information region, then the whole information region is observed, and complete knowledge of its content is obtained. This complete knowledge includes the knowledge of whether or not a target is located in the *information region*. The probability that a target at time

$t \in \mathbb{N}_0$ is in the *information region* $i = 1, 2, 3, ..., N$ is given by $p_i(t)$.

Consequently, $p_i(t)$ is the probability *prior* to any observation made at time t. The probability vector $\mathbf{p}(t) = [p_1(t), p_2(t), ..., p_N(t)]$ depicts the probability distribution of the target's presence in the regions at time t. This distribution is called the *target distribution*. At $t = 0$, the target distribution is initialized as $\mathbf{p}(0)$.

3.1.2 Nodes and information handling

Independent of which path planner is used, the UGV plans its path between fixed discrete points, or coordinates, henceforth known as *nodes*, see Figure 3.2. The *nodes* are assigned an index $k \in \mathbb{K}$, where $\mathbb{K}$ is the index set of known, predefined, *free* nodes, with the size $|\mathbb{K}| = K$. Let $\varphi(t) = [x, y]$ denote the UGV's position at time t. The UGV's state at time t is denoted $x(t) = [\varphi(t), \mathbf{p}(t)]^\intercal$, and $x(t) \in \mathbb{X}_{free}, \forall t$. The UGV is seen as a *holonomic* system, capable of moving in any direction. At any given time t, the UGV is located at some node k. Therefore, it is equivalent to saying that the UGV plans its path between states. Furthermore, the actual movement between the discrete states is not treated, *i.e.*, only the straight path between states is planned, not the UGV's motion along the path. At $t = 0$, the UGV's state is initialized as $x(0) = [\varphi(0), \mathbf{p}(0)]^\intercal$.

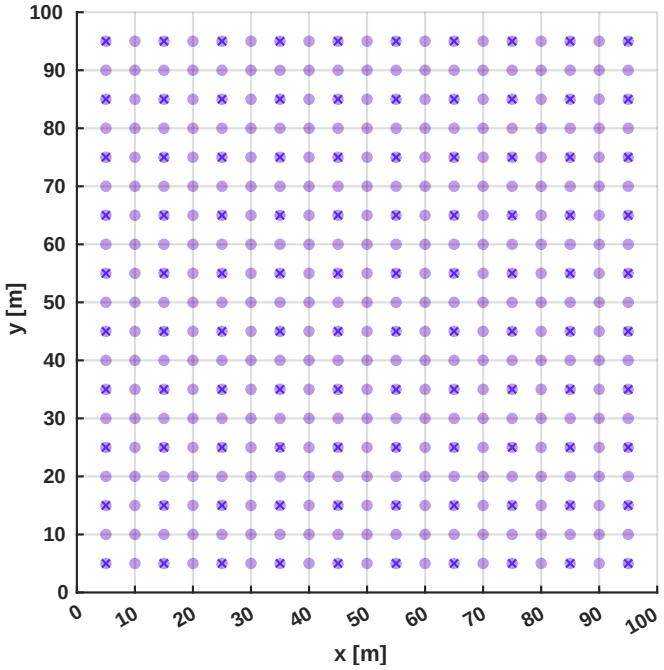


Figure 3.2: *An example of the uniform decomposition of the* scene *overlaid with* nodes. *The example depicts a quadratic decomposition of* $N = 100$ *and* $K = 361$. *The translucent dots represent each* node.

Suppose a robot moves on a 2D grid described by the previously mentioned 4CG, depicted in Figure 2.1. The robot moves between discrete grid points or nodes denoted by integer coordinates (i, j). The possible action the robot can take at each grid point is taking a discrete step in one of the four directions; up, right, down, left. Resulting in either increments or decrements of one of the integer coordinates i and j. The *action space*, given in (2.3), is in this case $\mathbb{U} = \{(0,1),(1,0),(0,-1),(-1,0)\}$. An adjacency matrix of the connectivity can be formulated using Definition 2.1. The resulting graph is undirected. The cost of traveling along an edge is equal for all edges. The *motion update function* f_{next} uses the obtained adjacency matrix $\mathbf{A}$ to represent the *action space* $\mathbb{U}$ from (2.3).

The *state update* consists of two parts; the probability update and the motion update. First, when the UGV has arrived at a new position, it observes its surrounding. Thus, updating the target distribution $\mathbf{p}(t)$. The details of the update depend on the used model, but the basis of the update of $\mathbf{p}(t)$ is an observation of the neighboring information regions from the UGV's current state $x(t) = [\varphi(t), \mathbf{p}(t)]^\mathsf{T}$. The observed regions from state $x(t)$ are represented by the set $\mathbb{S}_{x(t)}$, defined as

$$\mathbb{S}_{x(t)} = \{i \in \mathbb{S} \mid \|c_i - \varphi(t)\| \leq r\}, \tag{3.1}$$

where c_i is the coordinates to the centroid of region i, $c_i = [x, y]_i$. The UGV observes the surrounding through its "ambiguous" sensor configuration, defined only by a detection radius r. This observation is done through a simple ray-casting approach in 2D space based on [28], described further in Section 3.1.3. Each separate observation of the neighboring *information regions* $i \in \mathbb{S}_{x(t)}$ have two possible *posterior* probabilities; If a target is found in *information region* i, then $p_i^\dagger(t) = 1$, else $p_i^\dagger(t) = 0$, where the dagger symbol, $\dagger$, indicates that the probability is posterior an observation.

Let Γ be the probability update algorithm. The update of $\mathbf{p}(t)$ can thus be generalized as

$$\mathbf{p}(t+1) = \Gamma(\mathbf{p}(t),\ \mathbb{S}_{x(t)}). \tag{3.2}$$

A temporary state $\tilde{x}(t)$ is thus obtained after the probability update as

$$\tilde{x}(t) = [\varphi(t),\ \Gamma(\mathbf{p}(t),\ \mathbb{S}_{x(t)})]^\mathsf{T} = \begin{bmatrix} \varphi^\mathsf{T}(t) \\ \mathbf{p}^\mathsf{T}(t+1) \end{bmatrix}. \tag{3.3}$$

The second and final part of the state update is the motion update, where the UGV moves to a new position. The position is updated as

$$x(t+1) = f_{next}(\tilde{x}(t), u(t)) = \begin{bmatrix} \varphi^{\mathsf{T}}(t) + u(t) \\ \mathbf{p}^{\mathsf{T}}(t+1) \end{bmatrix} = \begin{bmatrix} \varphi^{\mathsf{T}}(t+1) \\ \mathbf{p}^{\mathsf{T}}(t+1) \end{bmatrix}, \tag{3.4}$$

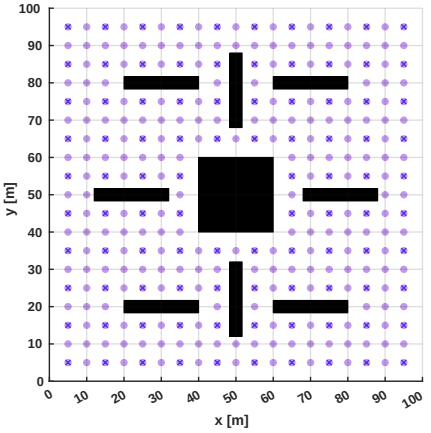
where $u(t) \in \mathbb{U}(\tilde{x}(t))$.

3.1.3 Obstacle handling

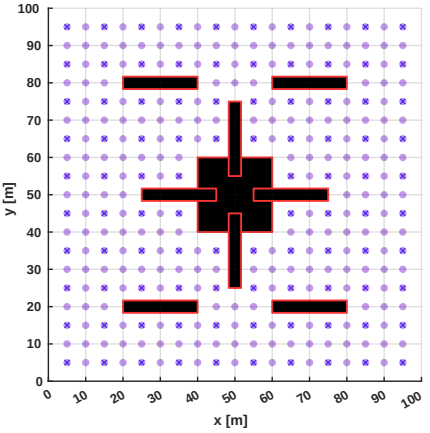
The definition of obstacles and the different effects they have on the studied scenario are presented in this section. For the first three mission complexity levels mentioned in Section 1.1, the UGV knows the positions of the obstacles in the scene.

Definition of obstacles

Each individual obstacle is defined as a convex closed four-sided, black polygon[1], with its sides parallel to the scene's coordinate axes, see Figure 3.3a. A concave closed obstacle is created by combining intersecting convex polygons[2], and Figure 3.3b shows an example.



(a) *An example of the scene's configuration with obstacles (black polygons). The original configuration was $N = 100$ and $K = 361$, but with the obstacles are $N = 96$ and $K = 300$.*



(b) *An illustration of how multiple convex closed obstacles overlap to create a concave closed obstacle. The separate polygons' borders are emphasized as red lines.*

Figure 3.3: *Example of different obstacle placements and how to create a concave obstacle.*

[1] MATLAB object `polyshape`

[2] By applying the `polyshape` object method `union`

Occupied regions and states

An information region is deemed occupied if an obstacle covers its centroid. Similarly, a node is deemed occupied if it is located within an obstacle. The occupied state space $\mathbb{X}_{occ}$ is represented by the combined occupied space comprised of *all* obstacles as one whole (disjoint) polygon[1]. A check[3] whether a point is inside the $\mathbb{X}_{occ}$ is made for all centroids and nodes before deployment. All points located within $\mathbb{X}_{occ}$ are removed from the corresponding sets $\mathbb{S}$ and $\mathbb{K}$. Note that $N = |\mathbb{S}|$ and $K = |\mathbb{K}|$; consequently, N and K decrease with every removed region and node, respectively. Thus, $\mathbb{K}$ is kept as the index set of the free nodes and states, and $\mathbb{S}$ is kept as the index set of the free regions. This leads to $\mathbb{X}_{free} \cap \mathbb{X}_{occ} = \emptyset$ and $\mathbb{X}_{free} \cup \mathbb{X}_{occ} = \mathbb{X}$, where $\mathbb{X}$ is the set of all conceivable states, *i.e.*, the complete state space, which coincides with (2.1).

Occultation

Obstacles obstruct the UGV from observing regions within the detection radius r, *i.e.*, an occultation of regions occurs. Different methods to handle this obstruction of the UGV's sight-lines were examined. The examined preexisting MATLAB function[4] does not classify the UGV's sight-lines as obstructed when tangent to an obstacle's corner. However, obstacles' corners should obstruct the UGV from observing regions.

Thus, the occultation of regions is implemented through a simple ray-casting approach in 2D space. The UGV's sight-lines are lines drawn from the UGV to each region's centroid within the detection radius. Subsequently, all line segments in the scene are extracted: the borders of all obstacles and the perimeter. Regions for which the corresponding sight-line from the UGV to its centroid intersects any of the extracted line segments are deemed blocked by said line segment, thus not observed.

As previously mentioned, the implementation is based on [28]. Two minor alterations to the method presented in [28] are made:

- $0 \leq T_1 \leq 1$ and $0 \leq T_2 \leq 1$ are used instead of $T_1 > 0$ and $0 < T_2 < 1$, respectively. This alteration is done to ensure that the region is included in the observation when the UGV is located at the said region's centroid. The alteration also accommodates that the rays from the UGV (the aforementioned sight-lines) are limited to the detection radius.
- T_1 and T_2 are split into corresponding components along each respective coordinate axis. Thus, becoming more manageable, especially when the sides of all obstacles are parallel to the coordinate axes.

The scene's layout is known before deployment for the first three mission complexity levels mentioned in Section 1.1. Thus, the mentioned calculations are

[3] By applying the `polyshape` object method `intersect`

[4] The `polyshape` object method `intersect`

done before deployment for *all* of the UGV's possible positions, *i.e.*, all free nodes, $\mathbb{K}$.

Obstruction of movement

The nodes located within an obstacle are not reachable from any other node. The UGV cannot move to a node located within or on the other side of an obstacle. This obstruction of movement also applies to the propagated probability used by the Markov Planner and by the identifiable target's random walk. This obstruction is implemented by modifying the adjacency matrix $\mathbf{A}$, zeroing the elements for which the associated edge is blocked. Similarly, the transition matrix $\mathcal{P}$ used by the target and the propagation of probabilities is modified before normalizing the transition probabilities.

3.2 Target distribution

The only differences between the *ExpPlanner* and the *Markov planner* are the representation and handling of the *target distribution*. Both planners make use of an "alarm system" described in Section 3.2.1. The first method models the target distribution between *all* free regions based on an exponential expression, described in Section 3.2.2. The second method uses a Markov chain to represent the target distribution and propagates it through the scene, described in Section 3.2.3.

Let $\mathbf{p}^{\dagger}(t) = [p_1^{\dagger}(t), p_2^{\dagger}(t), ..., p_N^{\dagger}(t)]$ denote the *posterior* probability vector, which depicts the probability distribution of the target's presence in the regions at time t, *posterior* to an observation. It is worth clarifying that the UGV can only identify a target from its "true" position, *i.e.*, observations of the target is not available during the path search phase. Hence, a target cannot be identified during the path search phase, resulting in only one possible *posterior* probability, $p_i^{\dagger}(t) = 0$, see Algorithm 3 and Algorithm 4. The *posterior* probability $p_i^{\dagger}(t) = 0$ is thus utilized as the notion that the *information region* i is observed.

3.2.1 Alarm system

The alarm systems inform that an intrusion has occurred by providing the initial target distribution $\mathbf{p}(0)$ with different degrees of certainty depending on the alarm system. The three different utilized alarm systems are given below.

One specific alarm region

The intruder enters the scene from the outside setting off an alarm in the perimeter region $\alpha \in \mathbb{S}_{perim}$, that it enters through, where $\mathbb{S}_{perim}$ is the set of free perimeter regions, with the size $N_p = |\mathbb{S}_{perim}|$. This alarm can be interpreted as an alarm from, *e.g.*, a seismic sensor in that α, indicating the presence of a *potential* intruder. There exist two versions of this alarm system; The *confident* and the *uncertain* alarm system. The difference between these is the initialization of

the probabilities (*target distribution*). The *confident alarm system* initializes the probabilities as,

$$p_i(0) = \begin{cases} 1, \ i = \alpha \\ 0, \ \text{otherwise} \end{cases}. \tag{3.5}$$

The *confident alarm system* initiates all probability in α; there is no doubt that the intrusion occurred in region α.

The *uncertain alarm system* initializes the probabilities as,

$$p_i(0) = \begin{cases} 0.9, \ i = \alpha \\ \frac{1-0.9}{N-1} = \frac{0.1}{N-1}, \ \forall i \neq \alpha \end{cases}. \tag{3.6}$$

The *uncertain alarm system* initiates one region as the most probable of having a target present with the probability $p_\alpha(0) = 0.9$ and distributes the remaining 0.1 between *all* other free regions.

Uniformly along the perimeter

The intruder enters the scene from the outside setting off an alarm uniformly along the perimeter $\mathbb{S}_{perim}$. This alarm can be interpreted as an alarm that "something has entered the scene", without specifying where it entered. This alarm system is called the *perimeter alarm system* and initializes the probabilities as,

$$p_i(0) = \begin{cases} \frac{1}{N_p}, \ \forall i \in \mathbb{S}_{perim} \ and \ N_p = |\mathbb{S}_{perim}| \\ 0, \ \text{otherwise} \end{cases}. \tag{3.7}$$

3.2.2 ExpPlanner, Exponential-based planner

The exponential-based planner *ExpPlanner* is used as a baseline and reference when examining the usefulness of the Markov Planner described in Section 3.2.3. For the ExpPlanner, it is assumed that $p_i(t)$ is independent of $p_j(\tilde{t})$ whenever $t \neq \tilde{t}$ or $i \neq j$, as per the last assumption in Section 1.5. Furthermore,

$$\sum_{i=1}^{N} p_i(t) = 1, \ \forall t. \tag{3.8}$$

Prior to observation, the target probability $p_i(t)$ is proposed to follow

$$p_i(t) = \frac{1 - e^{\frac{(\lambda_i - t)}{T f_s}}}{\sum_{j=1}^{N} \left(1 - e^{\frac{\lambda_j - t}{T f_s}}\right)}, \tag{3.9}$$

where $\lambda_i < t,\ i = 1, 2, ..., N$ denote the last time *information region* i was observed, the tuning parameter $\mathcal{T}$ is a time constant in seconds designed based on the estimated mobility of the target, and f_s is the sampling frequency in the number of steps the UGV takes per second. Let $T_s = 1/f_s$ denote the sampling time, *i.e.*, the time in seconds between the UGV's steps. For a more compact notation $\lambda = [\lambda_1, \lambda_2, ..., \lambda_N]$. The initial value of $\mathbf{p}(0)$ from equations (3.5), (3.6) and (3.7) results in λ being initialized according to either:

$$\lambda_i(0) = \begin{cases} -\infty,\ i = \alpha \\ 0,\ \forall i \neq \alpha \end{cases}, \tag{3.10}$$

$$\lambda_i(0) = \begin{cases} \mathcal{T} \ln(0.1),\ i = \alpha \\ \mathcal{T} \ln(1 - \frac{0.1}{N-1}),\ \forall i \neq \alpha \end{cases}, \tag{3.11}$$

$$\text{or } \lambda_i(0) = \begin{cases} \mathcal{T} \ln(1 - \frac{1}{N_p}),\ i \in \mathbb{S}_{perim} \\ 0, \text{ otherwise} \end{cases} \tag{3.12}$$

respectively.

The probability $p_i(t)$ depends on λ_i, the last time *information region* i was observed, as stated in expression (3.9). Thus, expression (3.9) can be called the "last-seen" function. Let h denote the planned path and let h be defined as a finite sequence of states. λ_i depends in turn on the previous *states* in the path h through the relation

$$\lambda_i(t) = \max_{\tilde{t}<t} \tilde{t} \quad \text{s.t.} \quad i \in \mathbb{S}_{h(\tilde{t})}, \tag{3.13}$$

where $\mathbb{S}_{h(\tilde{t})}$ is the set of observed regions from state $h(\tilde{t})$. The definition of a path h is further explained in Section 3.3.2. The expression (3.9) was proposed because it is "well-behaved" and describes an exponential increase in the probability of the target relocating to *information region* i. Furthermore, the resemblance between (3.9) and a normalized version of the exponential decay function (2.5) is not a coincidence. (3.9) can be seen as modeling the UGV's uncertainty that a previously observed region is still empty.

For the ExpPlanner, the probability update algorithm Γ in (3.2) is proposed to adhere to Algorithm 1.

Algorithm 1 Probability update algorithm (ExpPlanner)

Require: $\mathbb{X}$, $\mathbb{X}_{free}$, $\mathbb{X}_{occ}$, $\mathbb{S}$ and T_s
Observation is done from position $\varphi(t)$
Input: time t, observed regions $j \in \mathbb{S}_{x(t)}$ and λ
Output: $\mathbf{p}(t + T_s)$ and prior target distribution $\mathbf{p}_{obs}(t)$

1: $\mathbf{p}_{obs}(t) = \{p_j(t) \mid \forall j \in \mathbb{S}_{x(t)}\}$ ▷ Save prior target distribution
2: $\mathbf{p}(t + T_s) \leftarrow$ Update $\mathbf{p}(t)$ as per (3.9)
3: **for all** $j \in \mathbb{S}_{x(t)}$ **do**
4: $\quad \lambda_j = t$ ▷ Update λ for observed regions j
5: **end for**
6: **return** $[\mathbf{p}(t + T_s), \mathbf{p}_{obs}(t)]$

Note that the posterior probability $p_j^{\dagger}(t) = 0$, where $j \in \mathbb{S}_{x(t)}$ is done indirectly through the update of λ_j in line 4 in Algorithm 1.

The tuning parameter $\mathcal{T}$ can be seen as a *"stress parameter"* for the UGV when it travels through the scene using the ExpPlanner. This parameter regulates how often the UGV revisits regions. With the previously mentioned similarities between (3.9) and the exponential decay function given in [17], $\mathcal{T} * f_s$ can be seen as the mean lifetime of the "certainty" that the target has not returned to a previously observed region. Thus, the lower the $\mathcal{T}$, the more "stressful" the UGV behaves.

3.2.3 Markov planner

The second path planner *Markov planner* utilizes a Markov chain to model the target's possible position, *i.e.*, the target distribution. The modeling takes the form of propagation of the probability of a present target between neighboring regions. Consequently, the Markov planner's modeling of the target distribution is equivalent to modeling the target's estimated movement. Thus, it can be seen as an introduction of knowledge of the target's movement compared to the ExpPlanner in Section 3.2.2.

Single target handling

The target is represented by a homogeneous discrete-time Markov chain $\{X_t,\ t \in \mathbb{N}_0\}$ on the *Markovian state space*. Here, the Markovian state space is the *finite* index set for all *free information regions*, $\mathbb{S} = \{1, 2, ..., N\}$, where N is the total number of *free* regions. Each free region is seen as a *Markovian state*. The N-by-N *transition matrix* $\mathcal{P}$ is constant due to the homogeneity of the Markov chain, and its dimensions derive from $\mathbb{S}$ being finite. $\mathcal{P}$ is a right transition matrix; thus, the sum of each row in $\mathcal{P}$ is equal to one, as per

$$\sum_{j=1}^{N} \mathcal{P}_{ij} = 1, \ \forall i \in \mathbb{S}. \tag{3.14}$$

The transition probabilities from region i are chosen to be uniformly distributed between all permitted subsequent regions (Markovian states) from region i, *i.e.*, all transitions from i are equally likely. This includes the "transition" to i itself, *i.e.*, staying at i. $\mathbb{G}_i$ denotes the set of all permitted subsequent regions from i. Thus, a permitted subsequent region $j \in \mathbb{G}_i$ is defined as a region where an *unobstructed straight line segment* can be drawn between the centroid of i and the centroid of j. In other words, there must be an absence of obstacles between two regions' centroids for them to be permitted subsequent regions to each other. The same connectivity used between the nodes for 4CG is utilized between regions, see Figure 2.1.

$\mathcal{P}$ for a Markov chain with uniform transition probabilities is obtained by taking the related adjacency matrix $\mathbf{A}$ for the grid of regions and normalizing the elements with the sum of the corresponding row. All non-zero elements in $\mathbf{A}$ are unweighted and thus equal to one. Thus, this is equivalent to normalizing each row with the number of non-zero elements in each row, as per

$$\mathcal{P}_{ij} = \frac{A_{ij}}{\sum_{j=1}^{N} A_{ij}}, \ \forall i \in \mathbb{S}. \tag{3.15}$$

All free regions are made reachable from any other free region; thus, the Markov chain and its transition matrix $\mathcal{P}$ is irreducible as per Definition 2.9. As per Definition 2.12 and (2.13), all reachable regions are *aperiodic* and *positive-recurrent*. Thus, the entire Markov chain is *ergodic*, as per Definition 2.13.

Figure 3.4 depicts a state diagram for the associated Markov chain of a scene uniformly decomposed into nine non-overlapping regions yielding a 3-*by*-3-grid. The related transition matrix is given in (3.16), with the zero entries omitted.

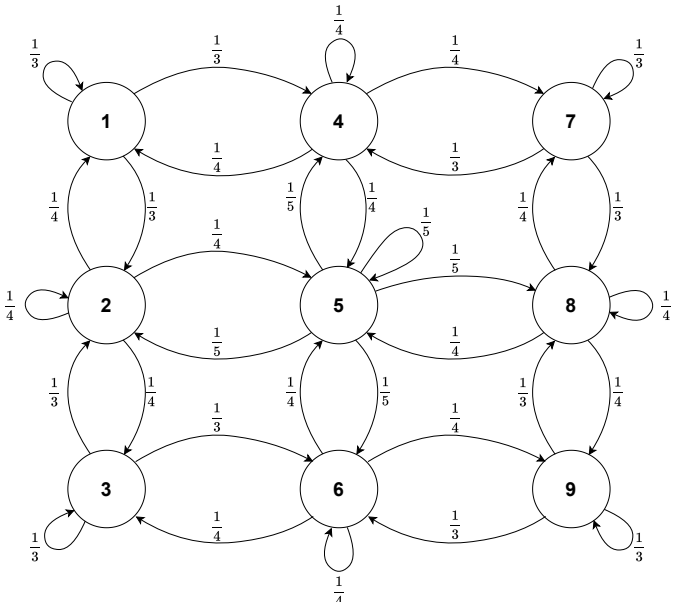


Figure 3.4: *The state diagram for a Markov chain with uniform transition probabilities and* $\mathbb{S} = \{1, 2, ..., 9\}$.

$$\mathcal{P} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & & \frac{1}{3} & & & & & \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & & \frac{1}{4} & & & & \\ & \frac{1}{3} & \frac{1}{3} & & & \frac{1}{3} & & & \\ \frac{1}{4} & & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & & & & \\ & \frac{1}{5} & & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & & \frac{1}{5} & \\ & & \frac{1}{4} & & \frac{1}{4} & \frac{1}{4} & & & \frac{1}{4} \\ & & & \frac{1}{3} & & & \frac{1}{3} & \frac{1}{3} & \\ & & & & \frac{1}{4} & & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ & & & & & \frac{1}{3} & & \frac{1}{3} & \frac{1}{3} \end{bmatrix} \tag{3.16}$$

The probability vectors $\mathbf{p}(t) = [p_1(t), ..., p_N(t)]$ and $\mathbf{p}^\dagger(t) = [p_1^\dagger(t), ..., p_N^\dagger(t)]$ describe the probability distribution that the target is present in one of the information regions at time t, *i.e.*, the target distribution, prior respectively posterior an observation, as previously mentioned in Section 3.1.1 and Section 3.1.2. It is assumed that a target is present somewhere in the scene, characterized by the sum of the probability vector always equals 1, as per (2.6).

After the initial target distribution from the "alarm systems" previously mentioned in Section 3.2.1, the probability propagates according to the previously mentioned Markov chain with the *uniform transition matrix* $\mathcal{P}$ described in (3.15). The target distribution is represented in the probability vector $\mathbf{p}(t)$. The propa-

gation of the probability vector $\mathbf{p}(t > 0)$ – the *update* of its individual elements, $p_i(t > 0)$, $i = 1, 2, ..., N$ – are described in Algorithm 2. Thus, the probability update algorithm Γ in (3.2) is proposed to adhere to Algorithm 2.

One fundamental theorem in probability theory is *Bayes' law* which describes how prior knowledge influences the conditional probability of an event happening. *Bayes' law* is presented in Theorem 3.1 below [29].

Theorem 3.1 (Bayes' law). *Let $H_1, H_2, ..., H_n$ be pairwise disjoint events and $H_1 \cup H_2 \cup ... \cup H_n = \Omega$, where Ω is the sample space, i.e., only one event H_i, $i \in \{1, 2, ..., n\}$, can happen during a trial. Then with an event A, the conditional probability that H_i is true if A is true is given by:*

$$Pr(H_i \mid A) = \frac{Pr(H_i)Pr(A \mid H_i)}{\sum_{j=1}^{n} Pr(H_j)Pr(A \mid H_j)} \tag{3.17}$$

.

Let $\gamma(t)$ denote the actual region of the target at time t. The observation at time t is done from state $x(t) = [\varphi(t), \mathbf{p}(t)]^\intercal$. *Bayes' law* is used to update, or normalize, the non-zero posterior probabilities $p_i^\dagger(t)$ with the following definitions:

$H_i = \{\gamma(t) = i\}$, which describes the event that *"the target is in the non-observed region $i \in \mathbb{S}_+$"*, where $\mathbb{S}_+ = \{s \in (\mathbb{S} \setminus \mathbb{S}_{x(t)}) \mid p_s^\dagger(t) > 0\}$. $A = \{\gamma(t) \neq j\}$, which describes the event that *"the target is* not *in the observed region $j \in \mathbb{S}_{x(t)}$"*.

This update results in the allocation of the probability of the observed region $j \in \mathbb{S}_{x(t)}$ *prior* to the observation at time $t > 1$, to the *non-observed* regions $i \in \mathbb{S}_+$ with the *non-zero posterior probability*, $p_i^\dagger(t) \neq 0$, as

$$\begin{aligned} p_i^\dagger(t) &= Pr(\,\gamma(t) = i \mid \gamma(t) \neq j) \\ &\overset{(1)}{=} \frac{Pr(\,\gamma(t) = i)\, Pr(\,\gamma(t) \neq j \mid \gamma(t) = i)}{\sum_{k=1}^{N} Pr(\,\gamma(t) = k)\, Pr(\,\gamma(t) \neq j \mid \gamma(t) = k)} \\ &\overset{(2)}{=} \frac{Pr(\,\gamma(t) = i)}{\sum_{k \in \mathbb{S}_+} Pr(\,\gamma(t) = k)}. \end{aligned} \tag{3.18}$$

The second equality (1) in (3.18) is *Bayes' law*, as per Theorem 3.1, for the conditional probability $Pr(\,\gamma(t) = i \mid \gamma(t) \neq j)$, which denotes *"the probability that the target is in region i if it is known that the target is not in region j"*. At the third equality (2) in (3.18) is $Pr(\,\gamma(t) \neq j \mid \gamma(t) = k)$, which denote *"The probability that*

the target is not in region j if it is known that the target is in region k" with

$$Pr(\gamma(t) \neq j \mid \gamma(t) = k) = \begin{cases} 1, & \text{if } j \neq k \\ 0, & \text{if } j = k \end{cases}, \ \forall j, k \in \mathbb{S}.$$

This is done in Algorithm 2 on line 6.

Algorithm 2 Probability update algorithm (Markov Planner)

Require: $\mathbb{X}$, $\mathbb{X}_{free}$, $\mathbb{X}_{occ}$, $\mathbb{S}$, T_s and $\mathcal{P}$
Observation is done from position $\varphi(t)$
Input: Time t and observed regions $j \in \mathbb{S}_{x(t)}$ and $\mathbf{p}(t)$
Output: $\mathbf{p}(t + T_s)$ and prior target distribution $\mathbf{p}_{obs}(t)$

1: $\rho(t) = \mathbf{p}(t)\,\mathcal{P}$ ▷ Propagate the target distribution $\mathbf{p}(t)$
2: $\mathbf{p}^{\dagger}(t) = \rho(t)$
3: **for all** $j \in \mathbb{S}_{x(t)}$ **do**
4: $\quad p_j^{\dagger}(t) = 0$
5: **end for**
6: Update $p_i^{\dagger}(t)$ according to (3.18) ▷ $i \in \mathbb{S}_+$
7: $\mathbf{p}_{obs}(t) = \{\rho_j(t) \mid \forall j \in \mathbb{S}_{x(t)}\}$ ▷ Save prior target distribution
8: $\mathbf{p}(t + T_s) = \mathbf{p}^{\dagger}(t)$
9: **return** $[\mathbf{p}(t + T_s), \mathbf{p}_{obs}(t)]$

Intuitively, the Markov planner prefers an initial target distribution that is as confident as possible to model the target movement more accurately. Thus, the *confident alarm system* from (3.5) is predicted to be preferred.

3.2.4 Target's random walk

As previously mentioned, $\gamma(t)$ denotes the region for which the target is located at time t. The random walk performed by the identifiable target is executed using the uniform transition probability matrix $\mathcal{P}$ given in (3.15) for *both* ExpPlanner and Markov planner. The probabilities $\mathcal{P}_{\gamma(t),j}$, $\forall j \in \mathbb{S}$, act as weights for the target's random walk. Let $\varsigma_{\gamma(t)}(j)$ denote the cumulative sum of $\mathcal{P}_{\gamma(t),j}$, $\forall j \in \mathbb{S}$. $\varsigma_{\gamma(t)}(j)$ is defined as

$$\varsigma_{\gamma(t)}(j) = \sum_{k=1}^{j} \mathcal{P}_{\gamma(t),k}. \tag{3.19}$$

Subsequently, by generating a random value distributed uniformly on the interval $[0, 1]$, and selecting the target's succeeding region $\gamma(t+1)$ based on the boundaries of $\varsigma_{\gamma(t)}$. Trivially, it could be seen as picking a random element (region) uniformly from the set $\mathbb{G}_{\gamma(t)}$.

3.3 Graph search

The considered graph search methods are presented below. These methods are used to find the path that optimizes the objective function. The examined search methods are presented in Section 3.3.1. The expanded search tree is described in Section 3.3.2.

The graph search's general structure and basic implementation are based upon existing code from the repository provided in the course TSFS12, *Autonomous Vehicles: Planning, Control and Learning Systems*, given at Linköping University [30]. Some alterations include (but are not exclusive to): A more efficient memory usage for the two search strategies. The search is done for a prediction horizon instead of the whole graph. The objective function is based on a failure probability instead of a travel cost. The target distribution, which is the basis of the objective function, changes between iterations is also an implemented modification. The mentioned alternations are presented in Section 3.3.1.

3.3.1 Search method

Two strategies of the *Forward search* were considered for the basis of the graph search; *breadth-first* and *depth-first*. Both strategies perform an *exhaustive search* of the given depth m within the graph, *i.e.*, they both constructs a complete search tree of depth m. Thus, both methods are *systematic* – all reachable nodes will be visited. The difference between the two strategies is the sorting function used for the queue q, as mentioned in Section 2.1.2.

The *breadth-first* approach specifies q as a FIFO queue, and the *depth-first* approach specifies q as a LIFO queue, defined in Section 2.1.2. Some alternations are done to the two strategies to adjust them to the considered scenario. Visited nodes are tracked but not excluded from being revisited; however, reversing to the previous node is only allowed if *no other* action is feasible. This alteration is done due to the target distribution changing between iterations, thus inducing the possibility that revisiting a node could yield a path with the lowest objective value. An exhaustive search is essential to ensure that all viable paths are examined due to the target distribution is continuously changing, and the path search relies on probabilities.

Both methods lead to at least *local optimality* for the chosen prediction horizon m as they ensure that all reachable nodes are visited. The performance difference lies within the memory usage and the execution time. The *breadth-first* approach results in a uniform expansion from the initial *root* node, *i.e.*, all plans of depth k must be examined before any plan of depth $k + 1$ is examined. This approach could be interpreted as "unnecessary" data being saved between each increment of depth during the path search phase and thus is the memory usage suboptimal.

A difference in the target distribution between branches of the same depth is virtually guaranteed. The extent of the difference between branches is related to how recent – measured in depth – they branched off from each other. The

more significant the difference in depth between the branch-off and the examined branches' nodes, the greater is the difference between the two branches' target distributions. Therefore, evaluating each branch to its end is a more efficient usage of the memory than uniformly evaluating each branch, *i.e.*, it is more efficient to use the *depth-first* approach than the *breadth-first* for the considered scenario. It is significantly more efficient, borderline necessary to use the depth-first due to the propagation of the target distribution of the Markov planner in Section 3.2.3. Thus, the search method is based on the *depth-first* approach chosen.

3.3.2 Search tree

From each newly visited node at time t_0, a search tree of potential paths $h \in \mathbb{H}_{x(t_0)}$ is expanded from the UGV during the path search. $\mathbb{H}_{x(t_0)}$ is the set of all allowed paths from state $x(t_0)$, *i.e.*, the whole search tree from state $x(t_0)$.

The complete *path* h that originates from state $x(t_0)$ is defined as a finite sequence of states, $h(t) \in \mathbb{X}_{free}$, where $t = \{t_0 + \tau T_s \mid \tau = 0, 1, 2, ..., m\}$ and m is the number of states in h, *i.e.*, the prediction horizon. The set $\mathbb{M}_{x(t)}$ is the succeeding states that the UGV can move to from state $x(t)$, that is, with $x(t) = h(t) \Rightarrow h(t + T_s) \in \mathbb{M}_{h(t)}$. The *state update*, or the transition from state $x(t)$ to state $x(t + T_s)$ is described in Section 3.1.2. Once again, it is emphasized that it is equivalent to say that the UGV plans its path between nodes or states.

The *branching factor* b, where no reversing and idling is allowed, is defined in (3.20), even though the graph's connectivity is that of 4CG in Figure 2.1.

$$b(t) = |\mathbb{M}_{h(t)}| = \begin{cases} \leq 3, \textit{ for } t > 1 \\ \leq 4, \textit{ for } t = 1 \end{cases}. \tag{3.20}$$

That is, only at the initial branching at $t = 1$ can b be equal to 4 if none of the neighboring nodes are obstructed by an obstacle. The average branching factor $\bar{b}$ for $t > 1$, where reversing or idling is not allowed, is obtained through the adjacency matrix $\mathbf{A}$ as

$$\begin{aligned} \mathbf{A}_i &= \sum_{j=1}^{N} \mathbf{A}_{i,j}\,, \\ \bar{b} &= \frac{\sum_{i=1}^{N} (\mathbf{A}_i - 1)}{N}. \end{aligned} \tag{3.21}$$

The size of the search tree $\mathbb{H}_{x(t_0)}$ follows

$$|\mathbb{H}_{x(t_0)}| = 1 + \sum_{M=0}^{m-1} \prod_{t=t_0}^{t_0+MT_s} b(t). \tag{3.22}$$

Where m is the prediction horizon and t_0 is the base time when the path search is initiated. The average branching factor in (3.21) can be seen as the average increase of the search tree $\mathbb{H}_{x(t_0)}$.

3.4 Objective function

An objective function related to the UGV's movement in the scene is utilized during each decision-making stage of the path search. The objective function is minimized by minimizing the probability of failing to find a target. This probability is called *failure probability* and is described in Section 3.4.1. The improved and final objective function is presented in Section 3.4.2. Both path planners use the presented objective function. At any time t, the UGV is located at a node with position $\varphi(t)$. In Section 3.4.3 the two algorithms for respective graph traversal and path search are outlined, see Algorithm 3 and Algorithm 4, respectively.

3.4.1 Failure probability

Each path h has a failure probability $\mathcal{F}(h)$, which is the probability that the UGV fails to find the target through its sensor when it moves along the path h. $\mathcal{F}(h)$ is defined as

$$\mathcal{F}(h) = \prod_{t=t_0}^{t_0+mT_s} \prod_{i \in \mathbb{S}_{h(t)}} 1 - p_i(t). \tag{3.23}$$

Note that $p_i(t)$ is the prior probability. Initially, to minimize the risk of failure to find the target, the path with the lowest probability of failure

$$h^\star = \underset{h \in \mathbb{H}_{x(t_0)}}{\arg\min}\ \mathcal{F}(h), \tag{3.24}$$

will be chosen for the UGV to follow. Where $h^\star$ denote the optimal path for the chosen horizon. In the unlikely case of multiple paths being optimal for the chosen horizon, *i.e.*, multiple $h^\star$ are obtained, a prioritizing "protocol" will be initiated. This "protocol" prioritizes paths with the longest initial straight line; and if there still exist multiple $h^\star$, one is chosen at random uniformly amongst them. This is outlined in Algorithm 4 on line 29.

3.4.2 Correction factor

There is a *possible* drawback with the Markov planner's representation of propagating the target probability when using the *confident alarm system* in (3.5). The possible scenario that the UGV and the target start at a sufficient distance from each other, that none of the paths in the initial search tree $\mathbb{H}_{x(0)}$ reaches a node from which the UGV can observe a region with non-zero probability within its detection radius r. This results in the dilemma of deciding what to do when all paths have an equal failure probability $\mathcal{F}(h) = 1,\ \forall h \in \mathbb{H}_{x(0)}$.

Initially, the *uncertain alarm system* in (3.6) was implemented to prevent this dilemma. A *semi*-heuristic, where the UGV moves towards the alarm region α

if $\mathcal{F}(h) = 1,\ \forall h \in \mathbb{H}_{x(0)}$, was introduced as an alternative. Some simulations of this gave promising results suggesting an improved performance if the direction of travel is considered. Thus, the *improved* objective function with a directional prioritization is suggested as

$$V(h) = \Bigg(1 + \underbrace{\varepsilon \frac{|\mathbf{v}_{max} - \mathbf{v}_{h(t_0+mT_s)}|}{|\mathbf{v}_{max} - \mathbf{v}_{h(t_0)}|}}_{Q}\Bigg) \prod_{t=t_0}^{t_0+mT_s} \prod_{i \in \mathbb{S}_{h(t)}} (1 - p_i(t)) = (1 + Q)\,\mathcal{F}(h). \tag{3.25}$$

The parameter ε is a scaling factor ensuring that the added *correction factor* Q is kept relatively small (around 0) so that it does not become too invasive on $\mathcal{F}(h)$. $\mathbf{v}_{h(t_0)} = [x, y]_{h(t_0)}$ is the position vector to state $h(t_0)$. $\mathbf{v}_{max} = [x, y]_{p_{max}}$ is the position vector to the region with the maximal probability at the initial search time t_0,

$$p_{max} \in \mathbb{S}_{max} = \arg\max_{i \in \mathbb{S}} \mathbf{p}(t_0). \tag{3.26}$$

The variable p_{max} is a random element selected uniformly from $\mathbb{S}_{max}$. Q is only applied once for each path h and resembles a form of heuristic, where movement in the direction of the initial maximal target probability is prioritized. Thus, $h^\star$ is obtained through,

$$h^\star = \arg\min_{h \in \mathbb{H}_{x(t_0)}} V(h). \tag{3.27}$$

Note that (3.23) and (3.25) coincide when $\varepsilon = 0$; and consequently, (3.24) and (3.27) also coincide. Henceforth, if nothing else is mentioned, the scaling factor is considered $\varepsilon = 0$ in this thesis.

When the UGV arrives at the first state in $h^\star$ at time t_1, $x(t_1) = h^\star(t_1)$, the remaining states in $h^\star$ will be discarded, and a new search tree $\mathbb{H}_{x(t_1)}$ will be generated from state $x(t_1)$; and a new $h^\star$ is generated through the new path search phase. This is repeated indefinitely or until an intruder is found, similarly to RHC described in Section 2.1.1.

The decision to use a prediction horizon was made because searching for a plan from a starting point to an endpoint at a reasonable distance from the starting point requires extensive computational power. Especially in the case of no identifiable target present in the scene, as the absence of a possible endpoint causes the path planners to run indefinitely. With an identifiable target present, path planning is terminated when: the target is found, a predefined termination time is reached, or a manual termination is received. Thus, m can be seen as an essential computational necessity.

3.4.3 Algorithms

A general outline of the developed algorithms are presented below. Algorithm 3 outline the UGV's graph traversal and Algorithm 4 outline the path search algorithm.

Algorithm 3 Graph traversal algorithm

Require: $\mathbb{X}$, $\mathbb{X}_{free}$, $\mathbb{X}_{occ}$, $\mathbb{S}$, $\mathbb{S}_{perim}$, T_s and $\mathcal{P}$

Initiate:

Alarm system as per (3.5), (3.6) or (3.7)

$t = 0$

$\gamma(0) \in \mathbb{S}_{perim}$, as per the initiated alarm system ▷ Target's true region

$x(0) = [\varphi(0), \mathbf{p}(0)]^\intercal \in \mathbb{X}_{free}$

$\mathcal{F}(x(0)) = 1$ ▷ Initiate failure probability

SUCCESS = $false$

1: **repeat**
2: $\mathbb{S}_{x(t)} \leftarrow$ Observe from state $x(t) = [\varphi(t), \mathbf{p}(t)]^\intercal$
3: **if** $\gamma(t) \in \mathbb{S}_{x(t)}$ **then** ▷ Target found?
4: $p^{\dagger}_{\gamma(t)}(t) = 1$
5: SUCCESS = $true$
6: **else**
7: $h^\star \leftarrow$ PATHSEARCH$(x(t), t)$ ▷ Call the search algorithm, see Algorithm 4.
8: $\tilde{x}(t) = [\varphi(t), \mathbf{p}(t + T_s)]^\intercal$ ▷ Update the $\mathbf{p}(t)$
9: **Define $u^\star(t)$ as the action to move to the first state in $h^\star$**
Move to the first state in $h^\star$:
10: $x(t + T_s) \leftarrow f_{next}(\tilde{x}(t), u^\star(t))$ ▷ $u(t) \in \mathbb{U}(\tilde{x}(t))$
Random walk of target:
11: $\gamma(t + T_s) \leftarrow$ RANDOMWALK$(y(t))$ ▷ As per Section 3.2.4
12: $t = t + T_s$
13: **end if**
14: **until** SUCCESS

The SUCCESS criterion on line 5 in Algorithm 3 is chosen to be the event that the target is *actually* found and *identified*. Since the modeling of the target distribution, previously mentioned in Section 3.2, is the basis of the path planning, the path planners can run indefinitely, as there is no "need" for an *identifiable* intruder to be present.

Algorithm 4 Path search algorithm

Require: $\mathbb{S}$, m, T_s
Input: Time of calling t_0 and initial state $x(t_0)$
Output: $h^\star$
Initiate:
$x(t_0).depth = 0$ ▷ Initiate the depth parameter to 0
$\hat{\mathbf{p}}(t_0) = \mathbf{p}(t_0)$ ▷ Save initial target distribution
Position vectors for Q **in** (3.25)
$\mathbf{v}_{x(t_0)} = [x, y]_{x(t_0)}$
$\mathbb{S}_{max} = \arg\max_{i \in \mathbb{S}} \mathbf{p}(t_0)$
$p_{max} \leftarrow$ PICKRANDOMFROM($\mathbb{S}_{max}$) ▷ Handle case of $|\mathbb{S}_{max}| > 1$
$\mathbf{v}_{max} = [x, y]_{p_{max}}$
$q = LIFO()$ ▷ LIFO queue
$\mathbb{H}_{x(t_0)} = \emptyset$ ▷ Initiate the search tree from state $x(t_0)$ as empty
$q.Insert(x(t_0))$ ▷ Place the initial state $x(t_0)$ on queue
while $q \neq \emptyset$ **do**
  $\hat{x} \leftarrow q.PopLast()$ ▷ The UGV's hypothetical state $\hat{x} \in \mathbb{X}_{free}$
  $t = t_0 + \hat{x}.depth\, T_s$ ▷ Update the local time t
  $h(t) = \hat{x}$
  if $\hat{x}.depth \geq m$ **then**
    $\mathbb{H}_{x(t_0)} = (\mathbb{H}_{x(t_0)} \cup h)$ ▷ If horizon m is reached, save branch h to search tree $\mathbb{H}_{x(t_0)}$
  else
    $\mathbb{S}_{h(t)} \leftarrow$ observe from state $h(t) = [\varphi(t), \mathbf{p}(t)]^\intercal$
    $[\hat{x}.\mathbf{p}, \mathbf{p}_{obs}(t)] \leftarrow$ Update the local target distribution $\hat{x}.\mathbf{p}$ as per Algorithm 1 or 2
    Update objective value $V(h)$ as per (3.25) using $\mathbf{p}_{obs}(t)$
  Generate succeeding states $\hat{x}(t + T_s) \in \mathbb{M}_{\hat{x}(t)}$ **:**
    for all $u(t) \in \mathbb{U}(\hat{x}(t))$ **do**
      $\hat{x}(t + T_s) \leftarrow f_{next}(\hat{x}(t), u(t))$
      $\hat{x}(t + T_s).\mathbf{p} = \hat{x}(t).\mathbf{p}$ ▷ Inherit the parent state's target distribution
      $\hat{x}(t + T_s).depth = \hat{x}(t).depth + 1$
      $q.Insert(\hat{x}(t + T_s))$
    end for
  end if
end while
$h^\star = \arg\min_{h \in \mathbb{H}_{x(t_0)}} V(h)$ ▷ As per (3.27)
if multiple($h^\star$) **then**
  Prioritize the path with the longest initial straight line.
  If still multiple $h^\star$ pick one uniformly.
end if
return $h^\star$

3.5 Simulation environment

The simulation environment that was developed alongside the path planners is presented here. The simulation environment can handle different decompositions and dimensions of the scene. It is not necessary for the decomposition of the nodes and regions to overlap. The user has considerable freedom when creating a scene related to its complexity, for example, the scene depicted in Figure 3.5. The target distribution is illustrated by a (cluster) *heatmap* where each region is colored according to the probability that the target is present in said region. The mapping of probability to color goes from *white/light blue* to *dark grey* with an increasing probability value. an descriptive colorbar is given alongside the heatmap. Examples of simulation environment are given in Section 4.3. Note that the UGV has moved to the first node of the previously obtained $h^\star$ for every presented example. After finding the target, it is possible to generate the path the UGV traversed to find said path.

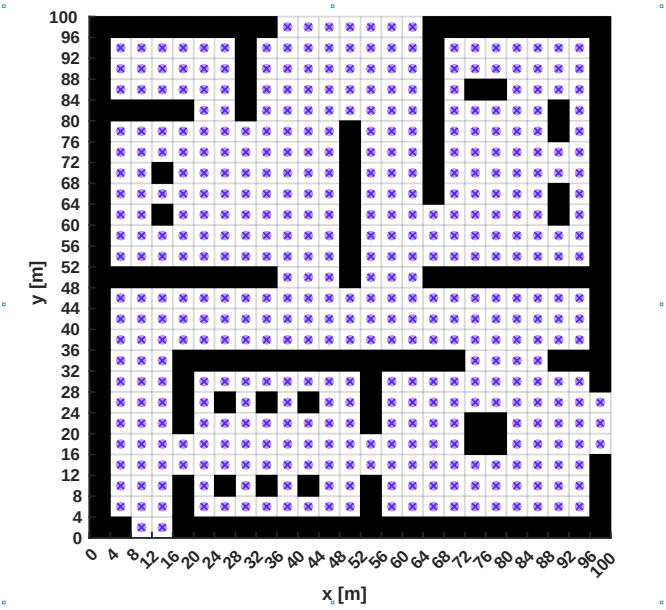


Figure 3.5: *An example of a possible complex indoor scene. The decomposition is* 25×25 *resulting in regions with dimension* 4×4 *meters.*

3.6 Path planner evaluation

Different methods to evaluate path planners exists, one of which is the Monte-Carlo simulations described in Section 3.6.1.

3.6.1 Monte-Carlo simulations

Monte-Carlo simulations are essentially the method of extracting statistical properties, such as the mean and variance, of a stochastic process from a sufficiently large collection of random simulations, in accordance with the *(weak) law of large numbers*. The law of large numbers declares that the *arithmetic mean*,

$$\bar{t}_n = \frac{1}{n} \sum_{i=1}^{n} t_i, \tag{3.28}$$

of a sequence of mutually independent and identically distributed stochastic variables $\{t_i\}$ converges toward the expected value of each stochastic variable $E(t_i) = \bar{t}$, $\forall i = 1, 2, ..$ for large enough n [29]. As such, the mean time to locate the target, the *mean detection time* $\bar{t}$, and the standard deviation σ can be obtained as evaluation parameters. As stated in Section 3.2.3, the used Markov chain is ergodic, and thus, the process can be represented statistically.

3.6.2 Evaluation of robustness

The general concept of *robustness* within control engineering is often related to how the controlled system handles model errors [31]. Although a possible approach for evaluation, *e.g.*, evaluating of how well the Markov planner can handle "model errors" by altering either the target's random walk or the transition matrix $\mathcal{P}$ so that they differ from each other, and analyzing the performance. This thesis defines *robustness* as; *"the consistency of the method's performance."* The number of outliers among the Monte-Carlo simulations quantifies the consistency of the performance. *Outliers* are defined as simulations that take an unreasonably large amount of iterations to find the target compared to the mean detection time $\bar{t}$. What defines an "unreasonably large amount of iterations" depends on the size of the world and the density of the cell decomposition. These outliers derive from unwanted behavior, such as; moving back and forth, moving irrational, *e.g.*, moving away from the target, only partially exploring the scene.

3.6.3 Computation time

The computation time t_{CPU} is also an evaluation parameter. Hence, the developed methods cannot be too computationally heavy and thus impossible to implement on a real UGV in the future.

3.6.4 Evaluation

The developed path planners are evaluated based on parameters obtained primarily through *Monte-Carlo simulations* described in Section 3.6.1. The primary approach of the simulations is to introduce an *identifiable* fictive target for which the movement follows a typical *random walk*, described in Section 2.5. 1000 simulations are done for each examined configuration and path planner if nothing else is mentioned. The configuration is defined as:

- The scene's layout.
  - The placement of obstacles.
  - The scene's decomposition and dimensions.
- The scaling factor ε in (3.25).
- The prediction horizon m.
- The detection radius r.
- The tuning parameter $\mathcal{T}$, used by the ExpPlanner.
- The transition matrix $\mathcal{P}$, used by the Markov planner.

Both path planners use the same 1000 starting positions for the UGV and the target; an array of these starting positions is henceforth referred to as *starting array*. Each simulation is executed until the target is found and identified, at which point the time t, or more accurately, the number of iterations it took to find the target is stored. When all 1000 simulations are done, the mean detection time $\bar{t}$ and the standard deviation σ of the number of iterations it took to find the target are calculated.

The Monte-Carlo simulations are outlined in Algorithm 5. The sample time $T_s = 1$ is chosen for simplicity and represent an arbitrary discrete time step to the next time instance, in each simulation.

It is worth emphasizing the fact that the random walk performed by the target is based on the same transition matrix $\mathcal{P}$ used by the Markov planner to propagate the target distribution. Hence, the Markov planner has no model error when it propagates the target distribution.

Algorithm 5 Monte-Carlo simulations

Require: $\mathbb{X}$, $\mathbb{X}_{free}$, $\mathbb{X}_{occ}$, $\mathbb{K}$, $\mathbb{S}_{perim}$, $\mathcal{P}$ and ε
Input: Number of simulations: $nSamples$
Output: Mean: $\bar{t}$ and standard deviation: σ
Initiate:
$results = zeros(nSamples, 1)$

1: **for** i=1:nSamples **do**
Initiate for each simulation:
2: Alarm system as per (3.5), (3.6) or (3.7)
3: $t = 0$
Spawn Target:
4: $\gamma(0) \leftarrow$ PICKRANDOMFROM($\mathbb{S}_{perim}$) ▷ Target spawns in random free perimeter region, $\gamma(0)$
Spawn UGV:
5: $x(0) \leftarrow$ PICKRANDOMFROM($\mathbb{X}_{free}$) ▷ UGV spawns in random free state, $x(0) = [\varphi(0), \mathbf{p}(0)]^\intercal$
6: **while** true **do**
7: $\mathbb{S}_{x(t)} \leftarrow$ Observe from state $x(t) = [\varphi(t), \mathbf{p}(t)]^\intercal$
8: **if** $\gamma(t) \in \mathbb{S}_{x(t)}$ **then** ▷ Target found?
9: $p^{\dagger}_{\gamma(t)}(t) = 1$
10: **break**
11: **else**
12: $h^\star \leftarrow$ PATHSEARCH($x(t)$, t) ▷ Call the search algorithm, see Algorithm 4.
Move to the first node in $h^\star$:
13: $x(t+1) = h^\star(t+1) \leftarrow f_{next}(x(t), u(t))$ ▷ $u(t) \in \mathbb{U}(x(t))$
14: **end if**
Random walk of target:
15: $\gamma(t+1) \leftarrow$ RANDOMWALK($y(t)$) ▷ As per Section 3.2.4
16: $t = t + 1$ ▷ Iterate one step
17: **end while**
18: $results(i) = t$
19: **end for**
20: Mean: $\bar{t} = Mean(results)$ ▷ As per (3.28)
21: standard deviation: $\sigma = \sqrt{Var(results)}$
22: **return** $[\bar{t}, \sigma]$

4
Results

Here are the results of the thesis presented. Section 4.1 presents the results used as the basis of the determination of $\mathcal{T}$. The main results presented are primarily centered around the evaluation of the developed methods to model the target distribution for a specific scene and are presented in Section 4.2.

4.1 The time constant $\mathcal{T}$

The variable $\mathcal{T}$ depends on the target's mobility, the number of steps it takes compared to the number of steps the UGV takes per second (f_s); and also the size and other "dimension-related parameters" of the *scene*, *e.g.*, the ratio between the number of free nodes K and free regions N. This makes it very difficult to produce a function over how $\mathcal{T}$ relate to all of these parameters. Thus, a suitable $\mathcal{T}$ is determined through an empirical study, by simulating an intrusion into an empty scene 1000 times for incremental values of $\mathcal{T}$, with the UGV starting from a random node and the intruder entering from a random perimeter region, $\gamma(0) = \alpha \in \mathbb{S}_{perim}$. The *uncertain alarm system*, given in (3.6), is used. The *decomposition* is 20×20 for both regions and nodes; the nodes share the coordinates with the regions' centroids. These simulations are repeated for three different starting configurations.

Table 4.1 presents the results from the empirical study to determine a suitable value for the time constant $\mathcal{T}$. The mean number of iterations it took to find target $\bar{t}$ and the standard deviation σ were used as evaluation parameters. Three different sets of starting positions (*starting array*) for the UGV and target are used to ensure the robustness of the empirical study. The determination of $\mathcal{T}$ is done in an empty scene as seen in Figure 4.1.

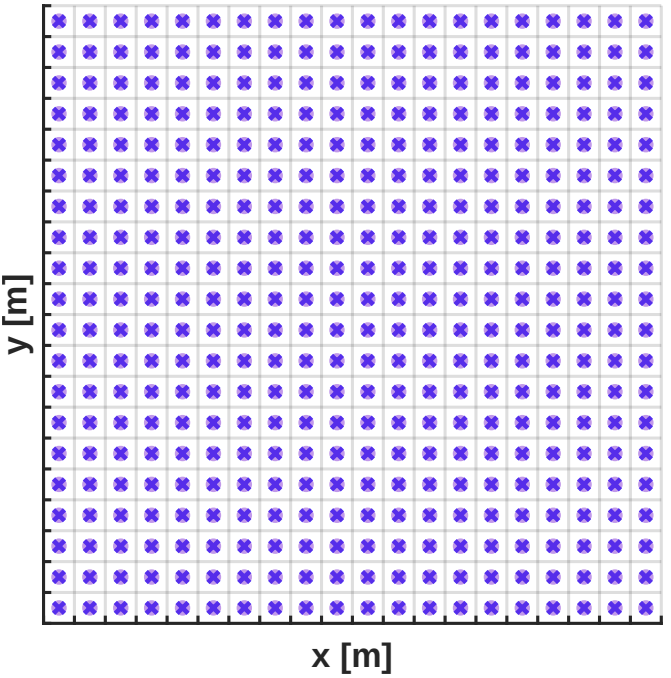


Figure 4.1: *The empty scene used to determine $\mathcal{T}$, the decomposition is $N_x = N_y = 20 \Rightarrow N = N_x \times N_y = 400$ for both regions and nodes. The nodes overlap with the regions' centroids. Regions have dimensions 5×5 meters.*

Table 4.1: *The results from the empiric study to determine a suitable $\mathcal{T}$. 1000 simulations were done for each starting array. $\bar{t}$ is the mean detection time, and σ is the standard deviation. The results for the chosen $\mathcal{T} = 450$ are highlighted.*

$\mathcal{T}$ $[s]$	***starting array* 1**		***starting array* 2**		***starting array* 3**	
	$\bar{t}$	σ	$\bar{t}$	σ	$\bar{t}$	σ
50	30.368	25.029	30.798	26.028	31.297	26.902
100	28.810	24.970	28.329	24.658	28.891	25.873
150	26.804	24.538	26.435	23.327	25.836	23.778
200	25.759	23.590	26.262	22.576	24.749	22.994
250	24.302	20.906	24.918	21.607	24.132	21.869
300	24.323	21.300	25.470	22.190	23.795	21.552
350	24.121	21.172	25.191	21.926	23.816	21.640
400	24.004	21.012	25.279	23.153	23.509	21.525
450	24.254	21.015	24.744	22.357	23.386	20.533
500	24.768	21.490	25.429	21.923	24.084	22.447
550	24.110	21.447	25.545	23.461	23.393	20.450

A general decrease of both $\bar{t}$ and σ can be seen with increasing $\mathcal{T}$, for each *starting array* up to around 350, after which $\bar{t}$ and σ start to fluctuate. $\bar{t}$ and σ for *starting array* 2 and 3 reach their minimum (for the examined $\mathcal{T}$) at $\mathcal{T} = 450$.

4.2 Sparse obstacle placement

Figure 4.2 depicts the examined configuration, *Sparse obstacle placement*. The *decomposition* used was 20×20 for both regions and nodes. The ExpPlanner used $\mathcal{T} = 450$ from Section 4.1, and the Markov planner used the uniform transition matrix $\mathcal{P}$ described in Section 3.2.3. The detection radius is $r = 16\,m$.

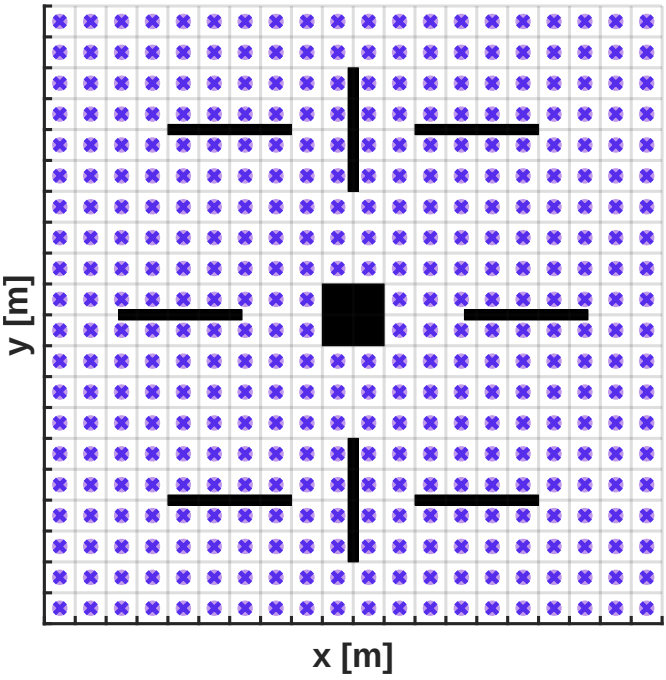


Figure 4.2: *Obstacle layout and cell decomposition of the* Sparse obstacle placement $N = K = 396$.

Three different *starting arrays* of 1000 starting positions for the UGV and the target were examined to ensure a large enough sample collection to evaluate the methods' robustness. The evaluation parameters are the mean number of iterations it took to find the target $\bar{t}$, the standard deviation σ, and the computation time t_{CPU}. Thus, $\bar{t}$ is the *mean detection time*. (3.21) gives *Sparse obstacle placement* the average connectivity for $t > 1$, $\bar{b} \approx 2.62$, which could be seen as the average increase of the search tree's size when increasing the horizon m for $m > 1$ as per (3.22). For each combination of alarm system and scaling factor ε, incremental values of the prediction horizon m are examined. Tables 4.2-4.6 present

the results.

4.2.1 Confident alarm system

The results for the confident alarm system given in (3.5) are presented in Tables 4.2, 4.3, and 4.4. Figure 4.3 illustrates the confident alarm system's initial target distribution and possible starting positions for the UGV and the target.

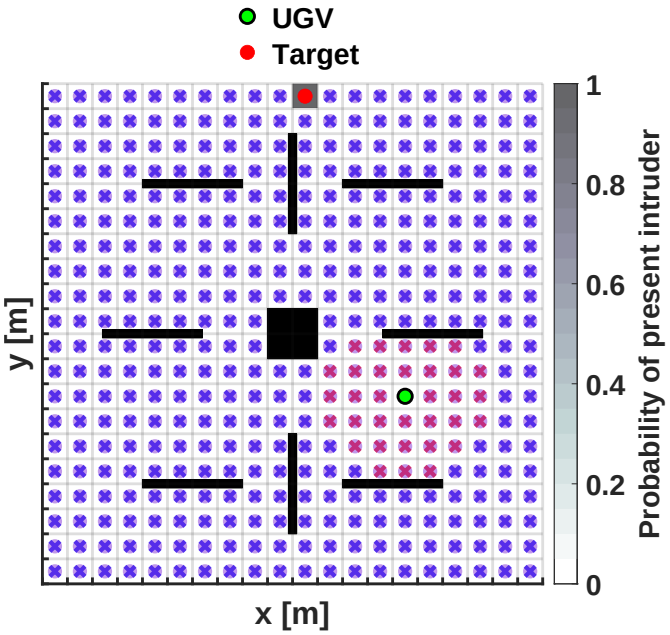


Figure 4.3: *Possible initialization of the confident alarm system. Here, the UGV starts in state* $x(0) = [\varphi(0), \mathbf{p}(0)]^\intercal$ *with position* $\varphi(0) = [72.5, 37.5]$ *and the target starts in region* $\gamma(0) = 201$*. The red crosses around the UGV are the centroids of the observable regions from position* $\varphi(0)$*, i.e.,* $\mathbb{S}_{x(0)}$*.*

Table 4.2: *The results for the confident alarm system* (3.5) *with incremental* m. t_{CPU} *is the computation time for* 1000 *simulations in seconds. The scaling factor used was* $\varepsilon = 0$.

			$m = 4$			
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	42.378	33.706	186.21	19.066	14.144	64.16
2	42.124	34.310	186.72	18.513	14.583	62.62
3	42.454	34.898	187.83	18.630	14.266	63.02
			$m = 5$			
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	36.996	32.083	408.81	17.306	13.113	148.96
2	40.384	35.381	450.83	16.837	12.855	143.33
3	37.409	32.228	417.80	17.112	12.901	146.99
			$m = 6$			
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	32.662	30.017	952.99	16.093	12.892	361.49
2	34.851	32.786	1002.22	15.621	11.541	343.73
3	32.424	29.376	924.68	15.703	11.275	350.57
			$m = 8$			
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	27.942	26.558	6177.30	14.294	9.876	2371.40
2	30.083	28.761	6412.17	14.004	9.037	2286.28
3	27.928	27.005	6047.76	14.422	9.618	2393.01

Both planners exhibit an improvement in performance with increasing horizon m, which is consistent with the theory of RHC, see Section 2.1.1; the further ahead the planning is done, the better the path planning will be able to predict and adjust. The results in Table 4.2 show that the Markov planner outperforms the ExpPlanner substantially for every examined horizon, resulting in a mean detection time $\bar{t}$ and standard deviation σ approximately half of the ExpPlanner's corresponding parameters.

The computation time (for 1000 simulations) t_{CPU} for the Markov planner is almost a third of the computation time for the corresponding ExpPlanner.

Table 4.3: *The results for the confident alarm system* (3.5) *with incremental* m. t_{CPU} *is the computation time for* 1000 *simulations in seconds. The scaling factor used was* $\varepsilon = 0.01$.

				$m = 4$		
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	27.734	27.794	118.06	13.254	7.727	43.90
2	28.793	28.555	126.07	13.030	7.707	43.26
3	26.770	26.822	115.18	13.546	8.589	45.03
				$m = 5$		
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	33.775	33.427	376.48	13.469	9.304	114.69
2	35.980	34.630	392.51	12.920	7.421	109.81
3	33.572	32.471	374.68	13.534	8.431	115.65
				$m = 6$		
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	27.262	27.451	789.27	13.529	10.041	301.72
2	31.128	30.536	893.64	13.083	7.805	290.49
3	28.358	27.703	827.16	13.551	8.478	299.22
				$m = 8$		
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	21.869	23.253	4901.79	13.554	9.377	2307.75
2	23.650	26.099	5359.32	13.223	7.859	2282.86
3	21.946	24.417	5018.19	13.579	8.320	2283.04

As seen in Table 4.3, compared to the case of $\varepsilon = 0$, see Table 4.2, both planners' performance is improved for every horizon with the introduction of Q with $\varepsilon = 0.01$. The ExpPlanner's performance follows the expected trend of RHC except for a deterioration in performance when increasing the horizon from $m = 4$ to $m = 5$. The performance of the Markov planner does not improve with the increasing horizon, contrary to the expected trend of RHC. The Markov planner's $\bar{t}$ and σ are essentially consistent for every horizon.

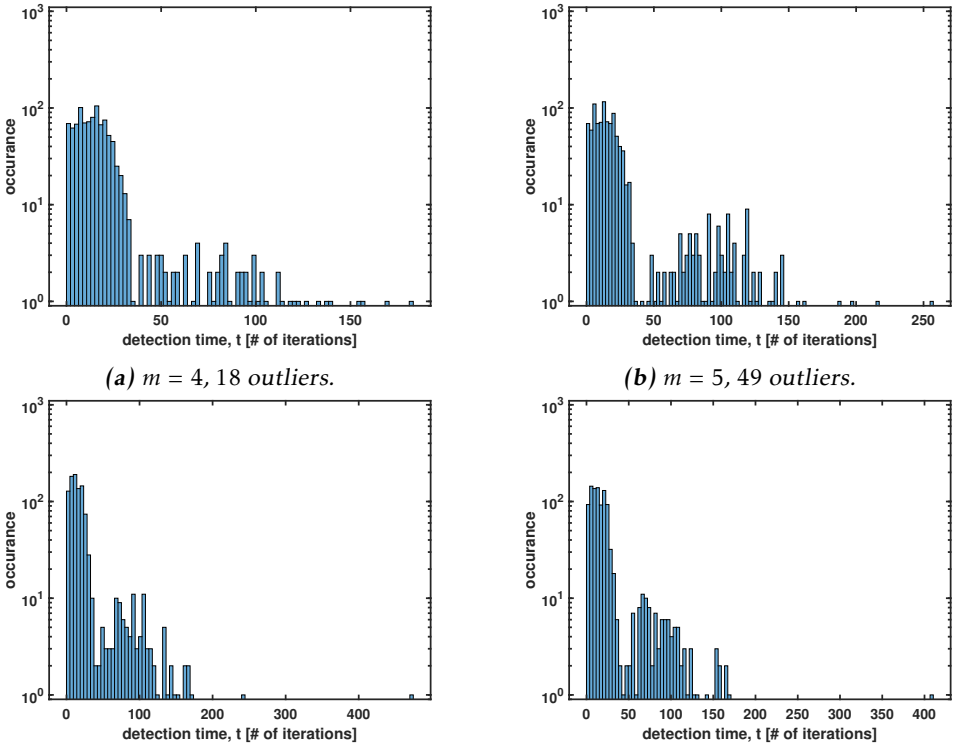
The computation time t_{CPU} is improved for both planners when the correction factor Q is introduced, which is to be expected with the improved performance of both planners. Nevertheless, the t_{CPU} for the Markov planner is half of that of the corresponding ExpPlanner.

Table 4.4: *The results for the confident alarm system* (3.5) *with incremental* m. t_{CPU} *is the computation time for* 1000 *simulations in seconds. The scaling factor used was* $\varepsilon = 0.1$.

				$m = 4$		
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	17.299	18.673	75.11	13.346	8.105	42.74
2	18.409	21.353	77.83	12.985	7.568	42.13
3	17.917	20.631	81.02	13.503	8.321	44.29
				$m = 5$		
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	20.445	25.376	227.54	13.266	7.970	110.08
2	23.248	30.276	261.24	12.976	7.578	108.27
3	20.435	23.881	224.87	13.568	8.445	113.54
				$m = 6$		
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	19.877	23.778	583.73	13.403	9.004	294.70
2	22.556	31.288	632.91	13.154	8.203	300.32
3	20.041	26.099	566.13	13.606	8.571	308.73
				$m = 8$		
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	21.544	25.856	4762.66	13.553	9.310	2308.48
2	22.898	29.405	5037.75	13.245	7.879	2236.54
3	21.263	25.767	4670.20	13.664	8.725	2320.70

Compared to the case with $\varepsilon = 0.01$ in Table 4.3, the ExpPlanner's performance is improved further for every horizon except $m = 8$ with $\varepsilon = 0.1$ in Table 4.4. Although its behavior does not follow the expected trend of RHC, see Section 2.1.1 and fluctuate seemingly arbitrarily between the different horizons. The performance of the Markov planner is virtually unchanged compared to the case of $\varepsilon = 0.01$ in Table 4.3. The t_{CPU} for the Markov planner is half of that of the corresponding ExpPlanner.

Figure 4.4 shows the distribution of the detection time t for *starting array* 2 when the ExpPlanner is used with the confident alarm system and $\varepsilon = 0.1$. Figure 4.5 shows the corresponding results for the Markov planner. A simulation with a detection time $t > 100$ is deemed an outlier, *i.e.*, a failure, per the definition in Section 3.6.2.



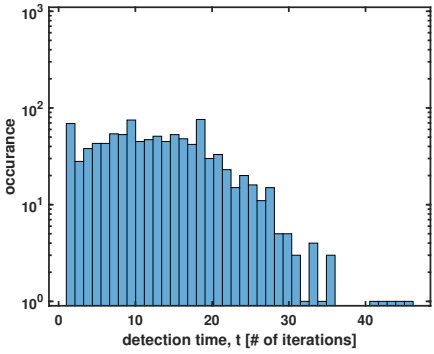
(a) $m = 4$, 18 *outliers.*

(b) $m = 5$, 49 *outliers.*

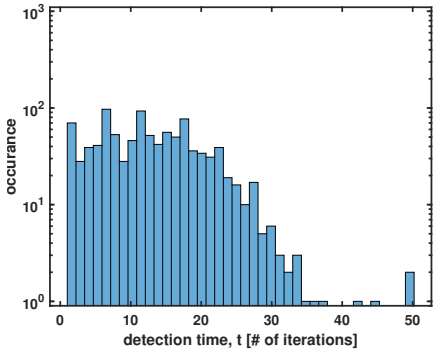
(c) $m = 6$, 39 *outliers.*

(d) $m = 8$, 33 *outliers.*

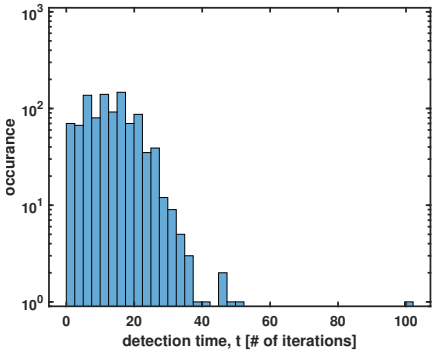
Figure 4.4: *The distribution of the detection time t for starting array 2 when the ExpPlanner is used with the confident alarm system and $\varepsilon = 0.1$ for the different horizons.*



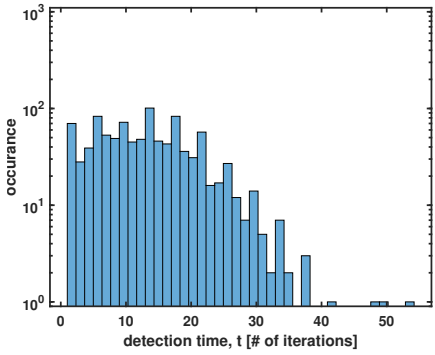
(a) *$m = 4$, 0 outliers.*



(b) *$m = 5$, 0 outliers.*



(c) *$m = 6$, 1 outlier.*



(d) *$m = 8$, 0 outliers.*

Figure 4.5: *The distribution detection time t for starting array 2 when the Markov planner is used with the confident alarm system and $\varepsilon = 0.1$ for the different horizons.*

4.2.2 Uncertain alarm system

The results for the *uncertain alarm system* given in (3.6) are presented in Tables A.1, A.2, and A.3. Figure 4.6 illustrates the initial target distribution for the *uncertain alarm system* and possible starting positions for the UGV and the target.

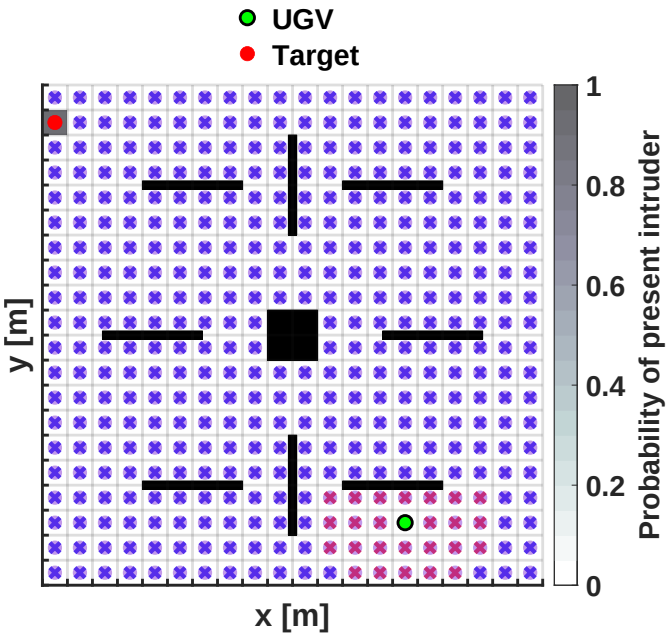


Figure 4.6: *Possible initialization of the* uncertain alarm system. *Here, the UGV starts in state* $x(0) = [\varphi(0), \mathbf{p}(0)]^\intercal$ *with position* $\varphi(0) = [72.5, 12.5]$ *and the target starts in region* $\gamma(0) = 2$. *The red crosses around the UGV are the centroids of the observable regions from position* $\varphi(0)$, *i.e.,* $\mathbb{S}_{x(0)}$.

Note that in Figure 4.6, the probabilities for all regions except the alarm region $\alpha = 2$ are $p_i(0) = \frac{0.1}{N-1} \approx 2.53 \cdot 10^{-4}$, $\forall i \in \{\mathbb{S} \setminus \alpha\}$ as per (3.6).

The performance of the ExpPlanner remains unchanged compared compare to the confident alarm system in Table 4.2. The performance of the Markov planner has improved slightly compared to the confident alarm system with $\varepsilon = 0$ in Table 4.2. Comparing Table A.2 with Table 4.3, and Table A.3 with Table 4.4, the performance of both planners is virtually the same as their performance with the confident alarm system with $\varepsilon = 0.01$ and $\varepsilon = 0.1$, respectively.

Both planners follow the convention of RHC with improved performance with increasing horizon m. There is no apparent difference in the computation time (t_{CPU}) between the confident alarm system with $\varepsilon = 0$ in Table 4.2 and the *uncertain alarm system* in Table A.1 for both planners. The Markov planner's t_{CPU} is still a third of the ExpPlanner's t_{CPU}.

4.2.3 Perimeter alarm system

The results for the perimeter alarm system given in (3.7) are presented in Tables 4.5, 4.6, and 4.7. Figure 4.7 illustrates the initial target distribution for the perimeter alarm system and possible starting positions for the UGV and the target.

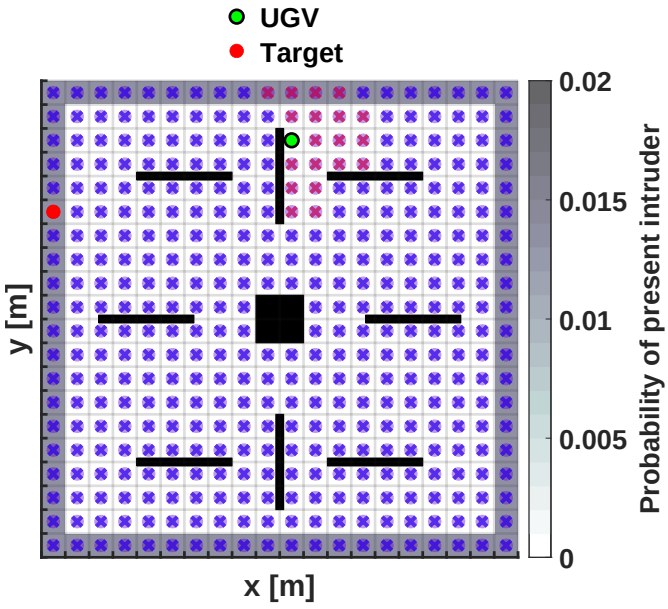


Figure 4.7: *Possible initialization of the perimeter alarm system. Here, the UGV starts in state* $x(0) = [\varphi(0), \mathbf{p}(0)]^\intercal$ *with position* $\varphi(0) = [52.5, 87.5]$ *and the target starts in region* $\gamma(0) = 6$*. The red crosses around the UGV are the centroids of the observable regions from position* $\varphi(0)$*, i.e.,* $\mathbb{S}_{x(0)}$*.*

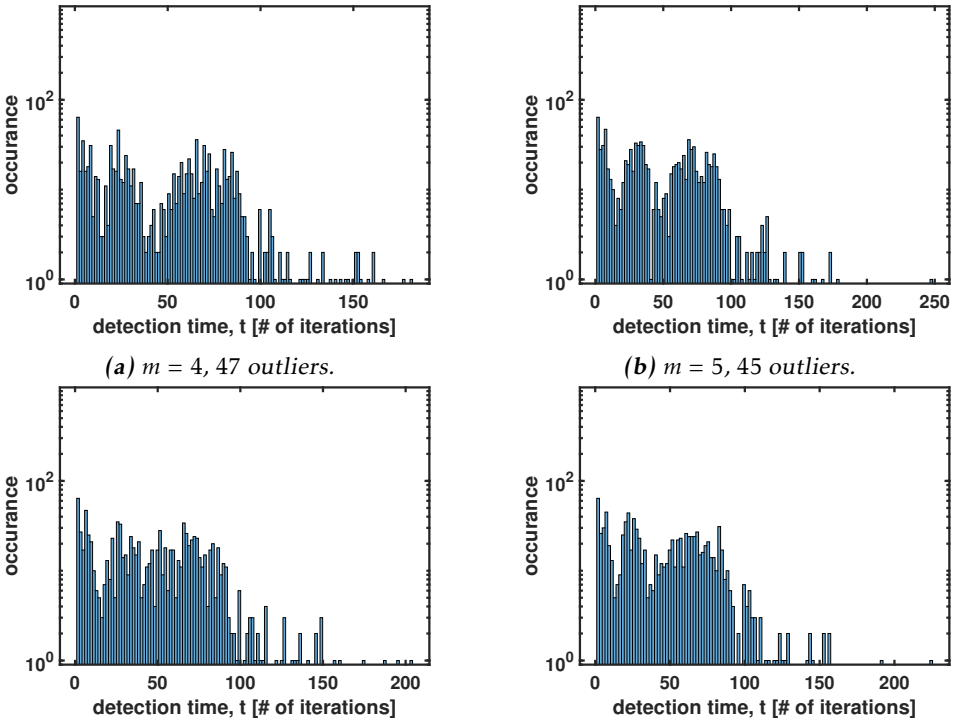
Note that the scale of the heatmap over the probability of a present intruder in Figure 4.7 is different from that in Figure 4.3 and Figure 4.6. This is because the target probabilities for the perimeter regions are $p_i(0) = \frac{1}{N_p} = \frac{1}{76} \approx 1.32 \cdot 10^{-2}$ as per (3.7).

Table 4.5: *The results for the perimeter alarm system* (3.7) *with incremental* m. t_{CPU} *is the computation time for* 1000 *simulations in seconds. The scaling factor used was* $\varepsilon = 0$.

				$m = 4$		
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	45.940	33.576	198.65	31.510	23.281	105.30
2	47.382	34.150	207.39	33.788	24.767	112.71
3	45.342	32.760	195.11	30.728	23.222	102.17
				$m = 5$		
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	47.367	34.203	512.84	31.939	24.406	266.80
2	48.671	34.365	523.98	33.849	24.711	284.29
3	46.267	32.072	508.13	30.534	22.782	255.73
				$m = 6$		
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	44.245	30.316	1249.69	31.499	23.860	671.80
2	46.729	32.962	1310.57	33.605	24.836	708.89
3	43.383	30.276	1205.39	30.437	22.878	648.48
				$m = 8$		
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	45.683	32.165	9328.25	31.417	22.760	4752.24
2	44.851	31.989	9074.50	33.391	24.476	5065.04
3	45.120	31.325	9113.08	30.790	23.665	4672.99

As seen in Table 4.5, the performance of both planners does not follow the convention of RHC and does not improve with an increasing horizon; $\bar{t}$ and σ practically stay unchanged. By examining the mean $\bar{t}$ and σ of all three starting arrays for each horizon, the lowest mean values of $\bar{t}$ and σ were obtained with $m = 6$, not $m = 8$, for both planners.

Figure 4.8 shows the distribution of the detection time t for *starting array* 2 when the ExpPlanner is used with the perimeter alarm system and $\varepsilon = 0$. Figure 4.9 shows the corresponding results for the Markov planner. A simulation with a detection time $t > 100$ is deemed as an outlier, *i.e.*, a failure, per the definition in Section 3.6.2.



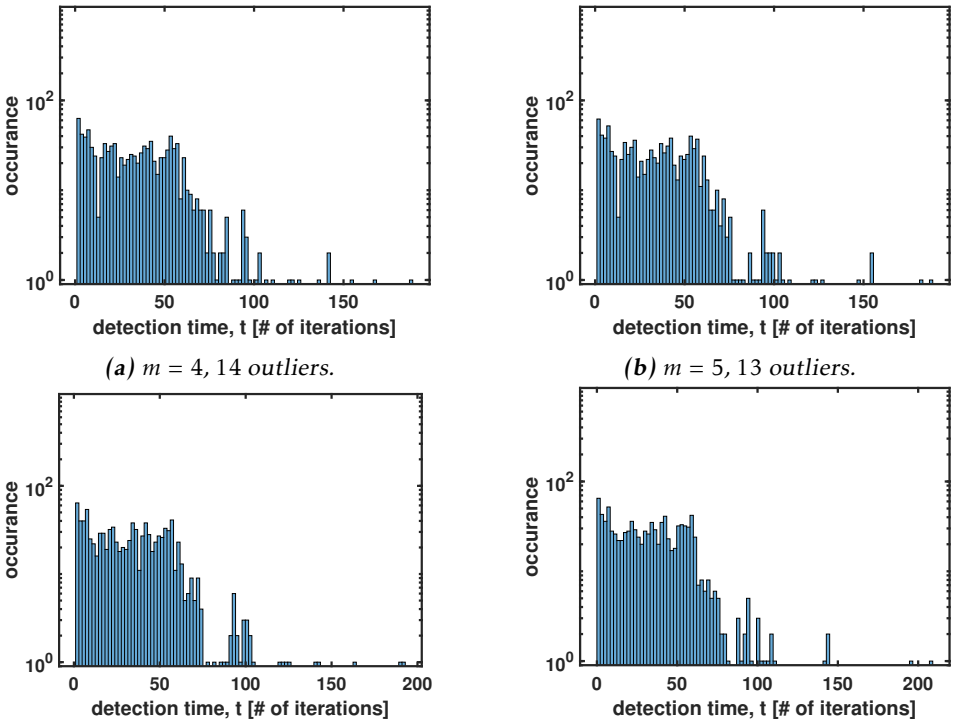
(a) *$m = 4$, 47 outliers.*

(b) *$m = 5$, 45 outliers.*

(c) *$m = 6$, 40 outliers.*

(d) *$m = 8$, 40 outliers.*

Figure 4.8: *The distribution of the detection time t for starting array 2 when the ExpPlanner is used with the perimeter alarm system and $\varepsilon = 0$ for the different horizons.*



(a) $m = 4$, 14 *outliers.*

(b) $m = 5$, 13 *outliers.*

(c) $m = 6$, 14 *outliers.*

(d) $m = 8$, 12 *outliers.*

Figure 4.9: *The distribution of the detection time t for starting array 2 when the Markov planner is used with the perimeter alarm system and $\varepsilon = 0$ for the different horizons.*

Table 4.6: *The results for the perimeter alarm system* (3.7) *with incremental* m. t_{CPU} *is the computation time for* 1000 *simulations in seconds. The scaling factor used was* $\varepsilon = 0.01$.

			$m = 4$			
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	45.468	32.858	183.31	32.166	25.810	102.74
2	44.643	32.002	185.53	34.056	24.630	109.51
3	44.771	32.894	185.77	31.380	23.882	100.69
			$m = 5$			
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	46.322	32.503	492.68	32.124	24.488	258.63
2	47.930	33.807	500.57	33.693	24.694	271.41
3	45.469	33.248	468.91	30.711	23.039	246.85
			$m = 6$			
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	43.766	31.265	1205.98	31.419	24.129	644.99
2	49.155	35.489	1330.29	33.534	24.626	690.75
3	45.214	31.976	1244.89	30.518	22.672	626.04
			$m = 8$			
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	44.115	30.619	9016.17	31.420	23.293	4599.91
2	45.804	32.517	9292.06	32.638	24.200	4790.03
3	43.182	30.895	8789.20	30.936	23.572	4548.19

Comparing Table 4.6 with Table 4.5 shows that the performance is virtually unchanged for both planners with the introduction of Q with $\varepsilon = 0.01$.

Table 4.7: *The results for the perimeter alarm system* (3.7) *with incremental* m. t_{CPU} *is the computation time for* 1000 *simulations in seconds. The scaling factor used was* $\varepsilon = 0.1$.

			$m = 4$			
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	52.035	34.525	213.41	32.240	24.354	101.09
2	55.749	38.865	221.12	34.054	25.067	106.53
3	53.807	37.972	215.33	32.733	24.788	103.02
			$m = 5$			
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	54.006	38.093	552.53	33.101	25.398	266.69
2	56.007	38.900	593.81	34.302	25.503	274.87
3	54.956	39.329	577.42	32.528	25.802	265.74
			$m = 6$			
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	53.718	38.351	1463.19	32.957	25.103	678.43
2	55.244	36.925	1495.85	34.723	25.800	714.85
3	56.737	44.146	1530.46	32.125	23.636	657.83
			$m = 8$			
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	58.029	56.916	11987.38	32.699	24.885	4804.45
2	56.929	38.714	11780.22	34.960	25.530	5128.23
3	56.681	39.673	11709.00	35.086	27.426	5157.83

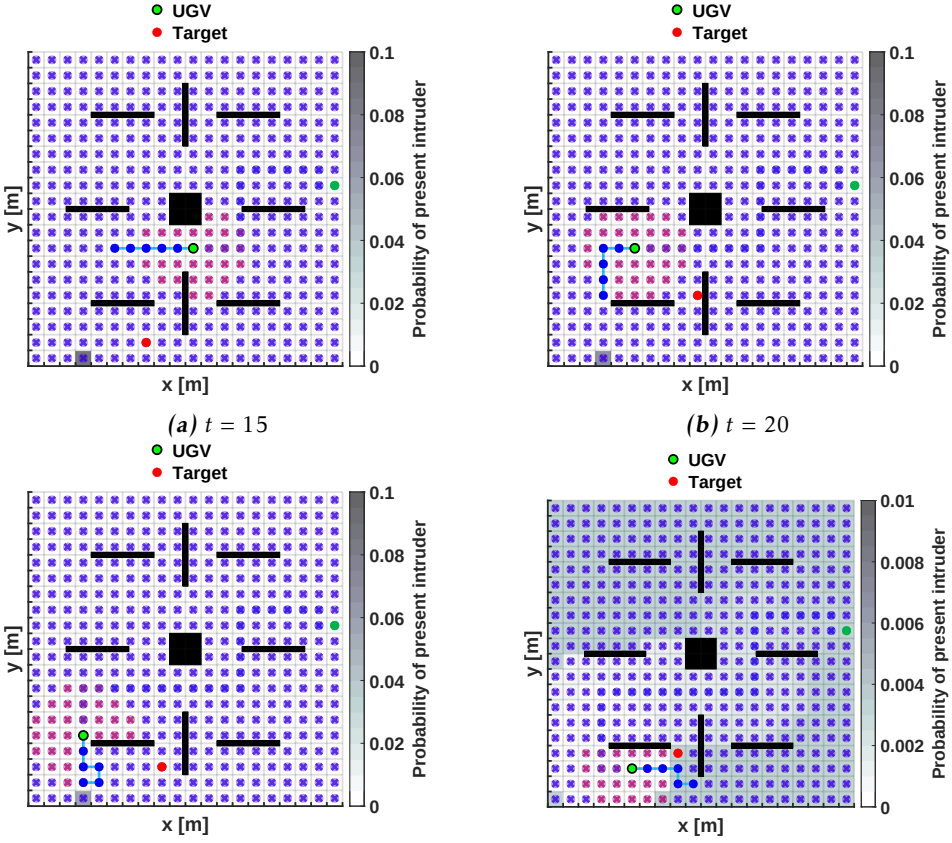
Comparing Table 4.7 with Table 4.6 shows that the performance of ExpPlanner deteriorates with the introduction of a correction factor Q with $\varepsilon = 0.1$. Furthermore, the performance of the ExpPlanner deteriorates with an increasing horizon, contrary to the convention of RHC. The Markov planner's performance does not change much between the different values of ε; furthermore, it remains consistent for all horizons, contrary to the convention of RHC.

4.3 Simulations

Examples of the simulations in the developed simulation environment are presented in this section.

4.3.1 ExpPlanner, simulation

As previously mentioned in Section 3.2.2, the ExpPlanner model the target distribution between all free regions; that is, it allocates a probability that the target is present in each region based on the last time the region was observed λ. Figure 4.10 shows an example of how the simulation could look when the ExpPlanner is used together with the confident alarm system (3.5), $\varepsilon = 0.1$, and $m = 6$. The apparent difference in the heatmap's color scheme between 4.10c and 4.10d is due to the rescaling of the colorbar's limits.

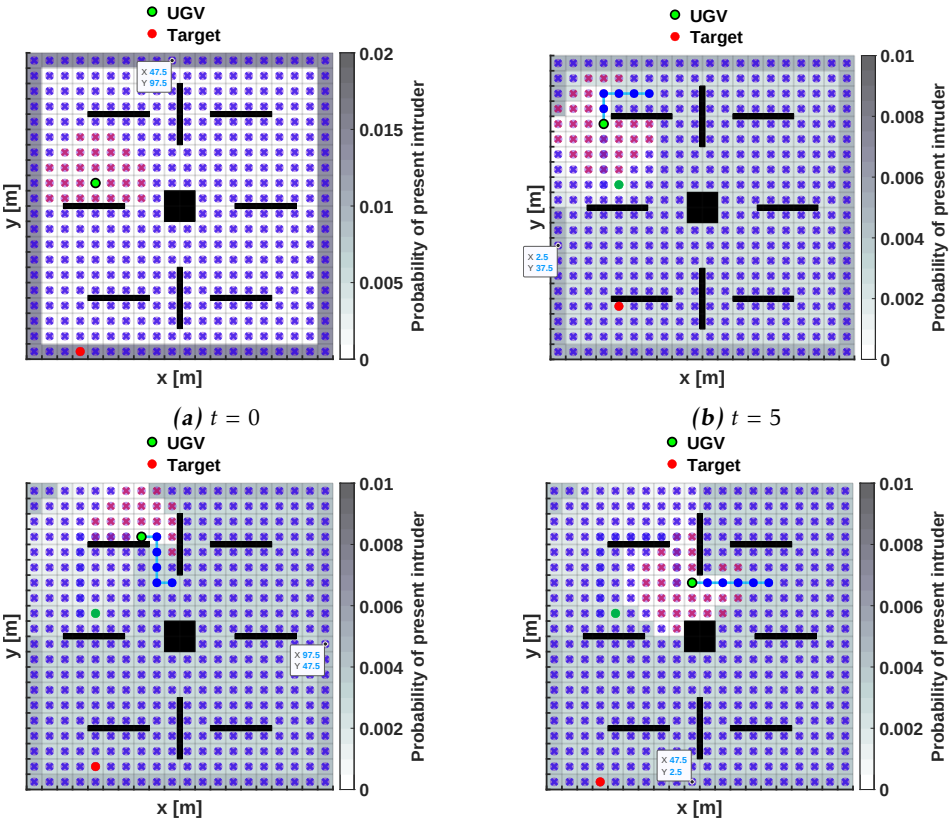


(a) $t = 15$ ***(b)*** $t = 20$

(c) $t = 25$ ***(d)*** $t = 29$, *the target is found.*

Figure 4.10: *Example of extracted time instances from a simulation using the ExpPlanner with the confident alarm system* (3.5), $\varepsilon = 0.1$, *and* $m = 6$. *The red crosses around the* UGV *are the observable regions, i.e.,* $\mathbb{S}_{x(t)}$. *The target is found at* $t = 29$.

Figure 4.11 illustrates an example of how the simulation could look like using the ExpPlanner together with the perimeter alarm system (3.7), $\varepsilon = 0.01$, and $m = 6$.

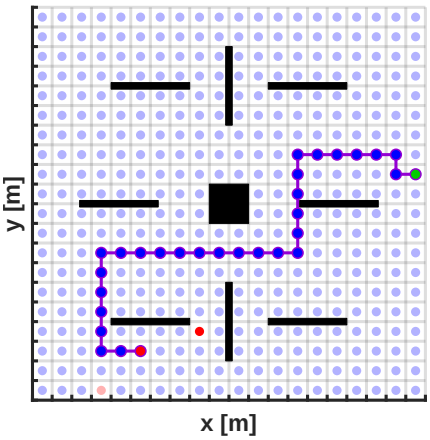


(a) $t = 0$ ***(b)*** $t = 5$

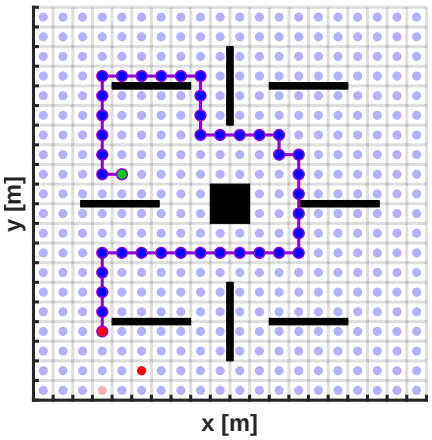
(c) $t = 10$ ***(d)*** $t = 15$

Figure 4.11: *Example of extracted time instances from a simulation using the ExpPlanner with the perimeter alarm system* (3.7), $\varepsilon = 0.01$, *and* $m = 6$. *The coordinates of the* $\mathbf{v}_{max} = [x, y]_{p_{max}}$ *is marked. The red crosses around the UGV are the observable regions, i.e.,* $\mathbb{S}_{x(t)}$. *The target is found at* $t = 39$.

The path traversed by the UGV in both simulation examples, Figure 4.10 and Figure 4.11, are shown in Figure 4.12a and Figure 4.12b, respectively.



(a) The traversed path of the confident alarm system example.

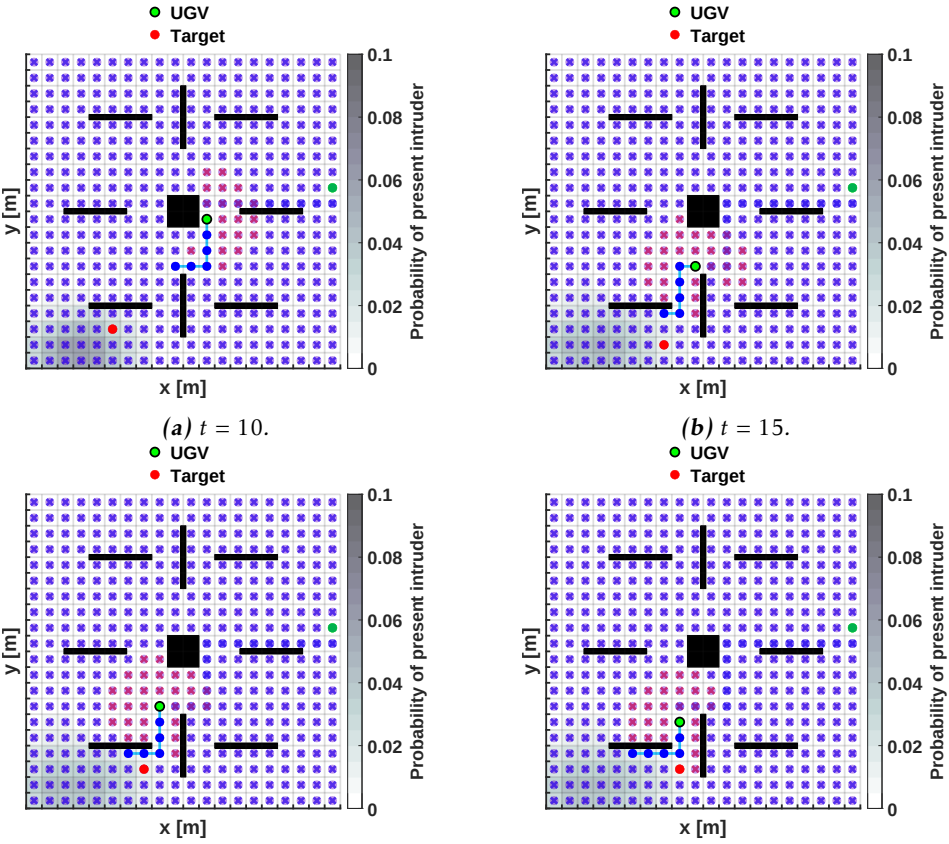


(b) The traversed path of the perimeter alarm system example.

Figure 4.12: *The traversed paths of* ***(a)*** *the confident alarm system example in Figure 4.10, and* ***(b)*** *the perimeter alarm system in Figure 4.11, generated using the ExpPlanner.*

4.3.2 Markov planner, simulation

As previously mentioned in Section 3.2.3, the Markov planner propagates the probability of a present target between neighboring regions. Figure 4.13 shows an example of how the simulation could look when the Markov planner is used together with the confident alarm system (3.5), $\varepsilon = 0.1$, and $m = 6$.



(a) $t = 10$.

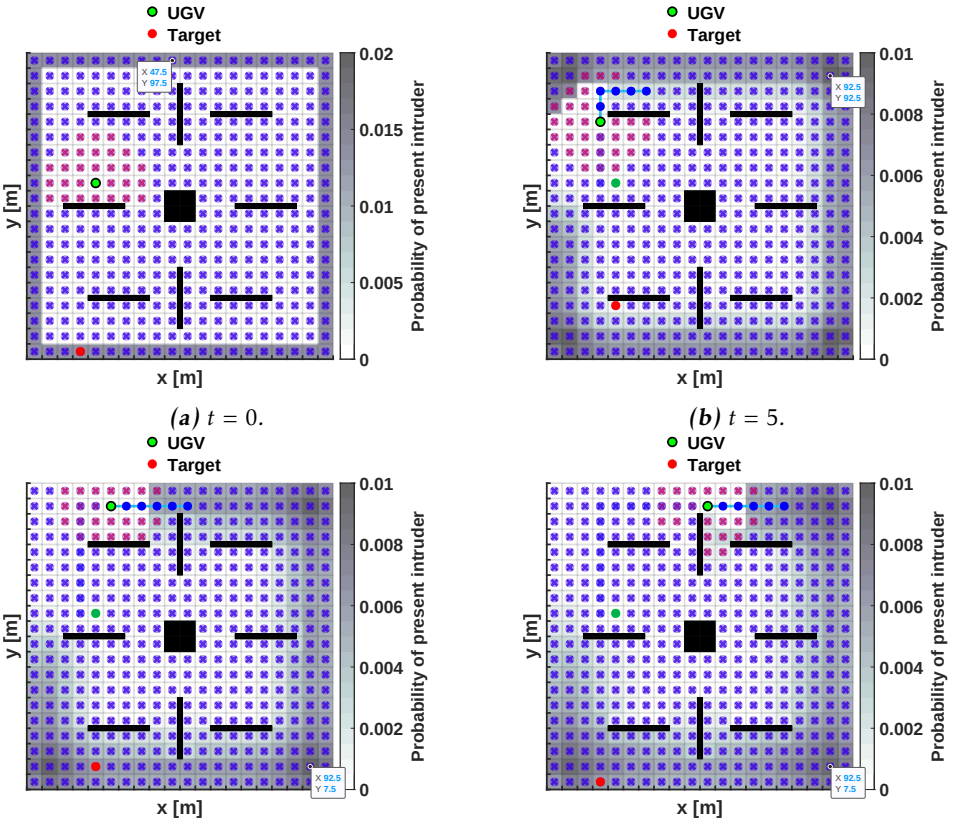
(b) $t = 15$.

(c) $t = 16$.

(d) $t = 17$, *the target is found.*

Figure 4.13: *Example of extracted time instances from a simulation using the Markov planner with the confident alarm system* (3.5), $\varepsilon = 0.1$, *and* $m = 6$. *The red crosses around the* UGV *are the observable regions, i.e.,* $\mathbb{S}_{x(t)}$. *The target is found at* $t = 17$.

Figure 4.14 illustrates an example of a simulation using the Markov planner together with the perimeter alarm system (3.7), $\varepsilon = 0.01$, and $m = 6$.



(a) $t = 0$.

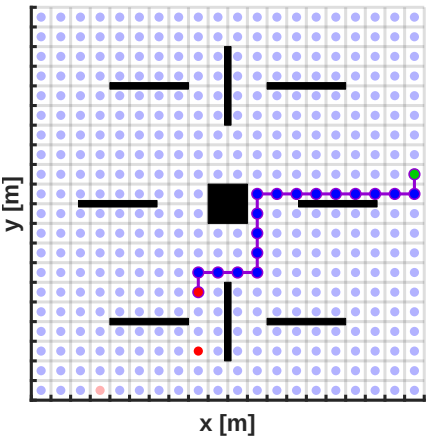
(b) $t = 5$.

(c) $t = 10$.

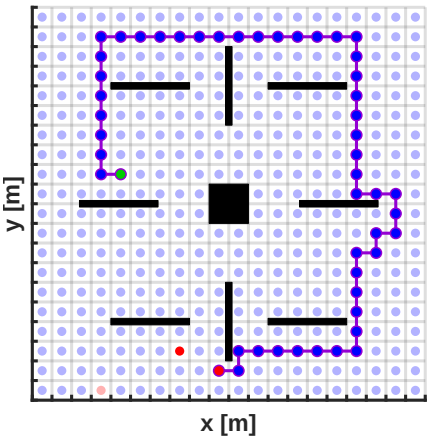
(d) $t = 15$.

Figure 4.14: *Example of extracted time instances from a simulation using the Markov planner with the perimeter alarm system* (3.7), $\varepsilon = 0.01$, *and* $m = 6$. *The coordinates of the* $\mathbf{v}_{max} = [x, y]_{p_{max}}$ *are marked. The red crosses around the* UGV *are the observable regions, i.e.,* $\mathbb{S}_{x(t)}$. *The target is found at* $t = 49$.

The path traversed by the UGV in both simulation examples, Figure 4.13 and Figure 4.14, are shown in Figure 4.15a and Figure 4.15b, respectively.



(a) The traversed path of the confident alarm system example.



(b) The traversed path of the perimeter alarm system example.

Figure 4.15: *Traversed paths of* ***(a)*** *the confident alarm system example in Figure 4.13 and* ***(b)*** *the perimeter alarm system in Figure 4.14. They are generated using the Markov planner.*

5
Discussion

The results presented in Chapter 4 are discussed in Section 5.1. The used method in Chapter 3 is discussed and evaluated in Section 5.2.

5.1 Results

The obtained results are discussed below in the order it is presented in Chapter 4.

5.1.1 The time constant $\mathcal{T}$

It should be noted that after $\mathcal{T} = 250$, the changes in $\bar{t}$ and σ are relatively small between the different examined $\mathcal{T}$. Hence, any $250 \leq \mathcal{T} \leq 500$ would probably suffice as a suitable *"stress parameter"* for a UGV in a scene with similar decomposition, and sparse obstacle placement.

5.1.2 Confident alarm system

Scaling factor $\varepsilon = 0.01$

The increased values of both $\bar{t}$ and σ for the ExpPlanner with $m = 5$ compared with the rest of the horizons are perplexing. One hypothesis is that with a horizon of $m = 4$, some initial search trees $\mathbb{H}_{x(t)}$ are not large enough to reach α, and their paths' failure probabilities $\mathcal{F}(h)$, $\forall h \in \mathbb{H}_{x(t)}$, are small enough for Q to direct the UGV towards the $p_{max} = \alpha$. However, for $m = 5$, some of the initial search trees $\mathbb{H}_{x(t)}$ are large enough to make the contribution of Q in (3.25) negligible compared to that from $\mathcal{F}(h)$, but still not large enough to reach α. For some of these initial search trees, the initial state could be located away from α and the target; thus, prioritizing paths that deviate from the shortest path to α from

$h(t_0)$. The ExpPlanner's performance is practically the same for $m = 4$ and $m = 6$, with the exception of the expected increased t_{CPU} with $m = 6$. This suggest that the ExpPlanner does not gain anything with the introduction of Q with $\varepsilon = 0.01$ when $m \lesssim 7$.

Scaling factor $\varepsilon = 0.1$

The fluctuation of the performance of the ExpPlanner between the horizons, is probably a consequence of ExpPlanner's characteristic of allocating the observed probabilities $p_i(t)$, $\forall i \in \mathbb{S}_{x(t)}$ to *all* free regions. Consequently, if the initial alarm region α is observed, the allocation of p_α will give rise to the case where an arbitrary region is designated as the region with the maximal probability p_{max}. This is reinforced by the uniform and random selection of p_{max} from $\mathbb{S}_{max}$ in (3.26); this can be seen in Figure 4.10d. With this possible arbitrary p_{max}, a more randomized behavior is obtained when using the ExpPlanner if the correction factor Q interferes a lot with the failure probability $\mathcal{F}(h)$ in (3.25), allowing the directional prioritization too much influence. Furthermore, the ExpPlanner generally prioritizes the number of observed regions $|\mathbb{S}_{x(t)}|$ as shown in Figure 4.12a. Nevertheless, for the examined case, the performance of the ExpPlanner improved with $\varepsilon = 0.1$ in Table 4.4 compared to $\varepsilon = 0.01$ in Table 4.3.

The path generated by the Markov planner in Figure 4.15a is *almost* the shortest path to the target from the UGV's starting position; depending on the target's previous position, it could even be *the* shortest path. Further analysis of the Markov planner's optimality is needed to draw definite conclusions.

A comparison of Figure 4.4 and Figure 4.5 also suggests that the Markov planner is more robust (more consistent) than the ExpPlanner when using the *confident alarm system* with $\varepsilon = 0.1$.

The *confident alarm system* overall

The Markov planner benefits from the introduction of Q when using the *confident alarm system*. The consistency of the Markov planner's performance suggests that a directional prioritizing renders the usage of a long prediction horizon obsolete for the Markov planner. The Markov planner performs just as well with a horizon of $m = 4$ as with $m = 8$.

The ExpPlanner demonstrates general monitoring that prioritizes coverage and complete exploration, whereas planning utilizing the Markov planner exhibits a behavior similar to an aggressive guard dog, going straight for the intruder.

5.1.3 Uncertain alarm system

Scaling factor $\varepsilon = 0$

The slight improvement of the Markov planner's performance compared to the *confident alarm system* with $\varepsilon = 0$ in Table 4.2 is to be expected as the purpose of introducing the *uncertain alarm system* in (3.6) was to handle the possible cases

of the initial search trees $\mathbb{H}_{x(t_0)}$ not reaching a node where the UGV can observe non-zero probabilities, as previously mentioned in Section 3.4.1. A comparison of the *confident alarm system* with $\varepsilon = 0.1$ and the *uncertain alarm system* with $\varepsilon = 0$ is given in Section 5.1.5.

The *uncertain alarm system* overall

There is essentially no difference between the confident and the *uncertain alarm system* with the introduction of the correction factor Q.

5.1.4 Perimeter alarm system

Scaling factor $\varepsilon = 0$

The consistent performance of both planners in Table 4.5 is because of the uniform property of the initial target distribution. The more uniform the target probabilities are distributed between the regions, the more random the UGV will behave for either planner.

Comparing Figure 4.8 and Figure 4.9 suggests that the performance of the Markov planner is more robust (more consistent) than that of the ExpPlanner when the *perimeter alarm system* is used with $\varepsilon = 0$. Although, both planners struggle with planning when they use the *perimeter alarm system*.

Scaling factor $\varepsilon = 0.01$

That the introduction of Q does not affect the performance of both planners is not surprising. Comparing Table 4.6 and Table 4.5, a correction factor Q with $\varepsilon = 0.01$ does not affect $V(h)$ in (3.25) enough to randomize further the already random behavior of the UGV when Q is omitted.

The uniform distribution between the perimeter regions initially compels the UGV to plan its path along the scene's perimeter. It is only a question of whether the direction along the perimeter closes the distance to the target or extends it. Figure 4.14 illustrates an example of this when the Markov Planner is used. Here, the size of $\mathbb{S}_{max}$ is small, but its elements are scattered between the different corners of the scene.

As previously mentioned, the ExpPlanner demonstrates more general monitoring than the Markov planner. Figure 4.11 illustrates an example of a simulation where this behavior occurs using the *perimeter alarm system*. The deviation towards the scene's center in Figure 4.11d is most likely due to the higher information gained because of the increased number of regions observed, $|\mathbb{S}_{x(t)}|$. The location of p_{max} and its effect through the correction factor also contribute to this deviation from the perimeter.

Scaling factor $\varepsilon = 0.1$

The deterioration in performance for the ExpPlanner with $\varepsilon = 0.1$ in Table 4.7 compared to $\varepsilon = 0.01$ in Table 4.6 is most likely due to the uniform property of the *perimeter alarm system*'s target distribution, in tandem with the invasive Q. The uniform property of the target distribution results in multiple instances where a large number of regions obtain the maximal probability of a present target at the initial search time t_0, *i.e.*, $\mathbb{S}_{max}$ is large.

A probable reason for the consistent performance of the Markov planner is that the uniform property of both the initial target distribution and the Markov planner's probability propagation does not necessarily lead to a large $\mathbb{S}_{max}$. However, the elements in $\mathbb{S}_{max}$ are scattered around the UGV, illustrated in Figure 4.14. This combination of the uniform property of the probability propagation and an invasive Q induces a random behavior.

The deterioration of the performance of the ExpPlanner with an increasing horizon is probably due to it facilitating a larger distance between the starting node $\mathbf{v}_{h(t_0)}$ and the end node $\mathbf{v}_{h(t_0+mT_s)}$ of each branch $h \in \mathbb{H}_{x(t_0)}$; thus, increasing the invasive nature of Q. This increase of Q's invasive nature does not affect the Markov planner, probably because the uniform initialization of the *perimeter alarm system* propagates the target distribution to more regions than what the UGV can observe and neutralize. As previously mentioned, the correction term Q is not compatible with the uniform initialization of the *perimeter alarm system*.

The *perimeter alarm system* overall

A large $\mathbb{S}_{max}$ in combination with p_{max} being selected uniformly and randomly from $\mathbb{S}_{max}$ may induce a random behavior of the UGV through the invasive Q if the elements of $\mathbb{S}_{max}$ are scattered in different directions from the UGV. It could be detrimental even if $\mathbb{S}_{max}$ only has a few elements, as long as there are multiple elements and they are scattered around the UGV. Both planners struggle with planning when they use the perimeter alarm system.

5.1.5 Comparison of alarm systems

By comparing the results of the Markov planner using the *confident alarm system* with the correction factor Q with $\varepsilon = 0.1$ in Table 4.4 with the case of only using the *uncertain alarm system* in Table A.1, it is concluded that the former performs better than the latter. Furthermore, the Markov planner's performance is enhanced by introducing the directional prioritizing with the *confident alarm system*, see Section 4.2.1.

The results from Figure 4.5 and Figure 4.9 suggest that the robustness of the Markov planner's performance depends on the confidence of the initial alarm system and if a directional prioritizing is utilized. This dependence on the initial alarm system's confidence probably originates from the used Markov chain being ergodic, see Definition 2.13, and the transition matrix $\mathcal{P}$ being uniform, see (3.15).

The more the (initial) target distribution is (uniformly) spread out amongst multiple regions, the closer the target distribution $\mathbf{p}(t)$ is to the stationary distribution π. The closer the initial $\mathbf{p}$ is to π, the faster the Markov chain approaches stationarity. As previously mentioned, the closer the Markov chain is to stationarity, the less accurate, and thus the less useful, the Markov planner becomes.

5.1.6 Comparison of planners

The Markov planner outperforms the ExpPlanner in all examined scenarios. Furthermore, a comparison of the number of outliers of the Markov planner in Figure 4.5 and Figure 4.9 with the ExpPlanner in Figure 4.4 and Figure 4.8 hints that the Markov planner is more robust than the ExpPlanner. These results also indicate that the more uniform the target probabilities are distributed between the regions, the more random the UGV will behave, leading to more outliers. Thus, the performance is improved by introducing knowledge about the target's movement.

5.2 Method

Some of the more prominent shortcomings of the used method are:

- The number of features that were discretized.
- The absence of a controller and, thus, the absence of motion modeling between the discrete nodes.
- The lack of variation in the studied scene, *i.e.*, the layout of the scene, thus, damaging the study's robustness and reliability.

As mentioned, the mean detection time $\bar{t}$ is not a true time value; instead, it is the mean number of iterations. Another aspect that is heavily simplified through discretization is the complete observation of a region when its centroid is observed. Although, a more accurate description of the scene will be obtained when the size of the regions tends to zero, leading to more valid results. The different discretized aspects do contribute to more reliable results. Furthermore, the absence of a sensor model excludes any consideration of sensor error or possible environmental influence.

No controller was developed due to no consideration of the motion between the discrete nodes. As such, the validity of the results could be questioned, but the purpose of the study is to act as a preliminary evaluation of how multi-sensor systems should be designed, as stated in Section 1.2.

Due to time limitations, only one obstacle layout is examined, as mentioned in Section 4.2. Multiple simulations of various obstacle layouts are necessary for more reliable results.

5.2.1 Source criticism

Sources' credibility was constantly considered during the literature study. Sources by authors with multiple works done in the same field were deemed credible. Course literature from related courses at Linköping University and articles from well-established journals were also chosen as credible sources. Multiple sources were analyzed when searching for sources related to Markov chains due to the incoherence related to the definition of and difference between Markov chains and Markov processes as described in Section 2.5. The multiple sources related to the Markov chain and its properties were cross-examined to validate the material.

6
Conclusions

Presented in this chapter are all the conclusions drawn in this study. Section 6.1 and Section 6.2 shortly present conclusions related to the ExpPlanner and the Markov planner. Conclusions related to the objective function are presented in Section 6.3, and general conclusions related to the problem formulation in Section 1.3 are presented in Section 6.4. Finally, in Section 6.5 are suggested continuations of this thesis presented.

6.1 ExpPlanner

The ExpPlanner's purpose was to work as a baseline for the Markov planner. Still, some conclusions for the ExpPlanner are drawn. The planning done by the ExpPlanner exhibits general monitoring that prioritizes the number of observed regions $|\mathbb{S}_{x(t)}|$, and a larger covered area. The ExpPlanner benefits from a confident initial alarm system.

6.2 Markov planner

One of the main conclusions is that the Markov planner seems to need a confident initial knowledge of the target, *i.e.*, the initial target distribution must be confined to a few, relatively close alarm regions, preferably only one initial alarm region, α, as the *confident alarm system* in (3.5) does. A Markov planner should be implemented so that the number of regions with non-zero probability $|\mathbb{S}_{+}|$ is kept small.

The introduction of the correction factor Q in (3.25) further improves the Markov planner's performance. The scaling factor ε should be chosen low enough for Q

not to intrude on the failure probability $\mathcal{F}(h)$, (3.23), which is the fundamental objective value. A scaling factor of $0.01 \leq \varepsilon \leq 0.1$ would suffice for desirable performance. The introduction of directional prioritizing through Q also shortens the length of the needed prediction horizon m.

6.3 Objective function

With proper conditions, a probability-based objective function is considered suitable for path planning a UGV for monitoring an area. These conditions are primarily a confident initial alarm system and some directional prioritizing incorporated in the objective function. One possible flaw with the correction factor is that if multiple elements in $\mathbb{S}_{max}$ are scattered around the UGV, then an invasive correction factor could induce a random behavior through the uniform and random selection of p_{max} in (3.26).

6.4 General conclusions

One of the main conclusions is that both path planners benefit from a confident initial alarm system, and both planners are not suitable for a uniformly *uncertain alarm system*, such as the *perimeter alarm system* (3.7). The knowledge of the target's movement that the direct modeling of the Markov planner provides improves the performance distinctly compared to the performance of the more general ExpPlanner. From comparing Figure 4.4 and Figure 4.5, along with the comparison of Figure 4.8 and Figure 4.9, it is concluded that the Markov planner is more robust and more likely to detect an intruder than the ExpPlanner. The Markov planner has a smaller number of failures, *i.e.*, outliers, compared to the ExpPlanner.

It is suggested that a monitoring UGV plan its path utilizing an ExpPlanner when in a general monitoring mode, *e.g.*, a sort of passive surveillance mode, and when an alarm is registered, the path planning switches to the Markov planner, enabling the "aggressive guard dog" mode. The Markov planner is suitable for time-critical cases where urgent path planning is essential, *e.g.*, during an intrusion into vital installations. A *confident alarm system* that pinpoints the location of the intrusion is necessary.

Partitioning the surveillance area into cells is a well-examined simplification in autonomous control. Nevertheless, modeling the target distribution through Markov chains is a suitable approach with spatial partitioning. Albeit a tried simplification, spatial partitioning is deemed a still relevant tool in autonomous control; the density can be increased by decreasing the size of each cell and increasing the number of cells to converge to a continuous representation of the surveillance area.

Further investigations of the collaboration between stationary sensors and mobile sensor platforms such as the UGV are needed. An initial suggestion is a

probability-based representation of the target's position, possibly with Markov chains as a foundation to propagate the target distribution when the target moves outside the collaborative sensing area. In a collaboration between stationary sensors and mobile platforms, a stationary sensor is best utilized to indicate the target's position by directly manipulating the probability values for the monitored cells. The stationary sensors can also keep track of the number of targets in the surveillance area. These usages make it necessary to implement a centralized sensor network.

6.5 Future work

A more thorough analysis of the objective function with different configurations is desired to get a complete picture of the performance of the Markov planner, *e.g.*, changing the detection radius r, the decomposition, the step length of the UGV, the obstacle placement, and so forth. A more realistic scenario with a target with a purpose, *i.e.*, the target no longer moves according to a uniform random walk and instead moves towards a specific coordinate or goal, would be especially interesting. Rarely do intrusions into vital installations occur where the intruder randomly walks around. It would be interesting to study different transition matrices $\mathcal{P}$ that either; coincide or differ with the true transition matrix of the target's movement. To examine the robustness related to model error, a transition matrix that differs from the target's true transition matrix can be applied. The reuse of the search tree is highly recommended in future work for a more efficient implementation.

The structure of the Markov planner enables the modeling of multiple targets simultaneously. By representing each target with an individual target distribution (Markov chain) $\mathbf{p}_T(t)$, where T is the target index. Subsequently, by concatenating these target distributions to a matrix $\mathbf{P}(t) = [\mathbf{p}_1^\intercal(t), \mathbf{p}_2^\intercal(t), ..., \mathbf{p}_{T_{max}}^\intercal(t)]^\intercal$ is the new representation obtained. T_{max} denotes the number of present targets. Each region i will then be assigned the sum of the i:th column of $\mathbf{P}(t)$; consequently, the value in each region will be the expected value of the number of targets within that region as per

$$E_i\left\{\sum_{j=1}^{T_{max}} X_j\right\} = \sum_{j=1}^{T_{max}} E_i\{X_j\}. \tag{6.1}$$

Where X_j is the stochastic variable that target j is present. Furthermore,

$$E_i\{X_j\} = \sum_k k * Pr_{X_j}(k) = Pr_{X_j} \tag{6.2}$$

where k is the number of present targets from the Markov chain represented by X_j, which is either 0 or 1. This results in the expected number of intruders in

each region i being just the sum of the probability that each "active" intruder is present in region i. The only possible complication is that the transition matrix $\mathcal{P}$ becomes a diagonal block matrix of the $\mathcal{P}$ used for one intruder, most likely resulting in unreasonable computation time. This could be avoided by reusing the past search tree, as long as no new intruders are introduced; if so, a new search tree must be generated.

The final proposed continuation is to change the correction factor Q in (3.25) from only considering one region with the maximal probability to considering kernel-based prioritizing. Instead of prioritizing paths in the direction of the region with the single highest probability, Q will prioritize paths in the direction of the area or kernel (collection of regions) with the highest total probability. One suggested approach is to combine regions within an area with a diameter equal to the UGV's detection radius r, see Figure 6.1.

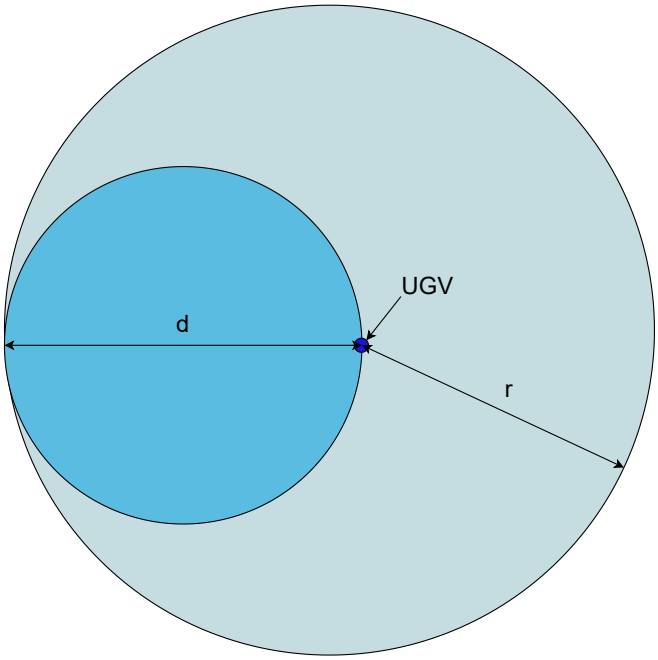


Figure 6.1: *The proposed kernel size for future work using kernel-based correction factor Q. d is the proposed diameter of the kernel, which is equal to the UGV's detection radius r.*

Appendix

A

Tables

The tables containing the less interesting results from the evaluation of the developed methods to model the target distribution.

A.1 Uncertain alarm system

The results related to the uncertain alarm system are presented here.

Table A.1: *The results for the* uncertain alarm system (3.6) *with incremental* m. t_{CPU} *is the computation time for* 1000 *simulations in seconds. The scaling factor used was* $\varepsilon = 0$.

			$m = 4$			
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	42.307	33.614	185.50	18.524	15.097	63.56
2	42.187	34.248	184.28	17.401	13.749	60.24
3	42.712	34.902	180.68	18.429	14.369	63.56
			$m = 5$			
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	37.314	32.560	412.84	15.924	12.313	141.64
2	39.938	34.787	450.17	15.092	10.730	135.53
3	37.968	33.177	413.82	15.819	11.015	141.60
			$m = 6$			
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	32.425	29.335	943.41	14.584	11.932	347.63
2	35.063	32.629	1019.36	14.232	9.250	333.61
3	32.889	29.661	936.35	14.824	11.617	348.64
			$m = 8$			
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	28.221	26.907	6030.48	14.146	9.787	2484.45
2	29.787	28.555	6376.55	13.931	9.238	2424.75
3	27.746	26.818	6000.34	14.298	9.766	2526.16

Table A.2: *The results for the* uncertain alarm system (3.6) *with incremental* m. t_{CPU} *is the computation time for* 1000 *simulations in seconds. The scaling factor used was* $\varepsilon = 0.01$.

			$m = 4$			
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	27.348	27.202	116.91	13.922	9.788	47.69
2	28.472	27.004	118.14	13.746	10.780	48.24
3	26.670	26.094	113.43	14.188	11.088	49.11
			$m = 5$			
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	33.710	33.206	352.22	14.057	10.188	126.16
2	35.565	34.183	370.57	13.490	8.420	120.19
3	33.352	31.996	351.10	14.116	9.600	126.91
			$m = 6$			
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	26.963	27.560	751.16	14.059	10.309	328.46
2	30.414	29.284	839.46	13.714	9.170	321.43
3	28.146	26.983	792.42	14.186	10.967	335.51
			$m = 8$			
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	22.027	22.797	4840.55	13.982	9.243	2518.62
2	23.862	26.375	5218.62	13.792	8.647	2457.65
3	22.129	24.786	4883.69	14.193	9.614	2558.62

Table A.3: *The results for the* uncertain alarm system (3.6) *with incremental* m. t_{CPU} *is the computation time for* 1000 *simulations in seconds. The scaling factor used was* $\varepsilon = 0.1$.

	$m = 4$					
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	17.408	18.832	75.50	13.529	8.802	45.83
2	18.939	23.297	83.12	13.142	7.838	44.45
3	17.967	23.182	77.65	13.611	8.512	46.15
	$m = 5$					
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	20.347	25.002	214.95	13.486	8.634	116.71
2	23.146	30.987	243.74	13.234	7.780	115.24
3	20.120	24.236	215.44	13.763	8.695	119.82
	$m = 6$					
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	20.280	24.845	568.14	13.707	9.644	310.58
2	22.229	28.963	627.37	13.388	7.950	303.15
3	19.969	23.291	562.95	13.968	9.437	320.97
	$m = 8$					
	ExpPlanner			Markov planner		
starting array	$\bar{t}$	σ	t_{CPU} [s]	$\bar{t}$	σ	t_{CPU} [s]
1	21.477	25.586	4653.44	13.830	8.867	2403.62
2	23.672	30.281	5121.46	13.900	9.283	2399.74
3	20.737	24.749	4553.85	14.322	10.013	2512.60

Bibliography

[1] Per Boström-Rost, Daniel Axehill, and Gustaf Hendeby. On global optimization for informative path planning. *IEEE Control Systems Letters*, 2(4):833–838, 2018. doi: 10.1109/LCSYS.2018.2849559.

[2] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, 2016. doi: 10.1109/TIV.2016.2578706. URL `https://ieeexplore.ieee.org/document/7490340`.

[3] Jouni Rantakokko, Kristofer Bengtsson, Jonas Nygårds, Fredrik Näsström, and Rogier Woltjer. Tekniköversikt autonoma och obemannade system - del 2: Markstriden, Mars 2020. ISSN 1650-1942. Rpt. No. FOI-R–4901–SE.

[4] S. Nilsson, M. Andersson, T. Andersson, E. Bilock, V. Deleskog, F. Hemström, D. Lindgren, S. Molin, J. Nygårds, E. Relfsson, and J. Rydell. Autonom övervakning med samverkande sensorer, December 2020. ISSN 1650-1942. Rpt. No. FOI-R–5065–SE.

[5] Per Skoglar. *Tracking and Planning for Surveillance Applications*. PhD thesis, Linköping UniversityLinköping University, Automatic Control, The Institute of Technology, 2012.

[6] Washburn Alan R. Search for a moving target: The fab algorithm. *Operations Research*, 31(4):739 – 751, 1983. ISSN 0030364X.

[7] Steven Michael. LaValle. *Planning algorithms*. Cambridge University Press, Cambridge, 2006. ISBN 0521862051. URL `http://lavalle.pl/planning/`.

[8] W.H. Kwon and S.H. Han. *Receding Horizon Control: Model Predictive Control for State Models*. Advanced Textbooks in Control and Signal Processing. Springer London, 2006. ISBN 9781846280177. URL `https://books.google.se/books?id=ITSKhlKy2u8C`.

[9] Eric W. Weisstein. Grid graph, Mars 2021. URL `https://mathworld.`

wolfram.com/GridGraph.html. From MathWorld – A Wolfram Web Resource.

[10] Per Boström-Rost. On informative path planning for tracking and surveillance, 2019. ISSN 0280-7971.

[11] Bruno Siciliano and Oussama Khatib. *Springer Handbook of Robotics*. Springer Handbooks Ser. Springer, 2 edition, 2016. ISBN 9783319325521.

[12] Amarjeet Singh, Andreas Krause, Carlos Guestrin, William Kaiser, and Maxim Batalin. Efficient planning of informative paths for multiple robots. Number IJCAI 2007 - 20th International Joint Conference on Artificial Intelligence, pages 2204–2211, 2007.

[13] Dutta Ayan, Bhattacharya Amitabh, Kreidl O Patrick, Ghosh Anirban, and Dasgupta Prithviraj. Multi-robot informative path planning in unknown environments through continuous region partitioning. *International Journal of Advanced Robotic Systems*, 17(6), 2020. doi: 10.1177/1729881420970461. URL https://doi.org/10.1177/1729881420970461.

[14] Frank Lingelbach. Path planning using probabilistic cell decomposition, 2005. ISSN 1404-2150. QC 20101209.

[15] Mathew, Neil. Discrete path planing strategies for coverage and multi-robot rendezvous. Master's thesis, 2014. URL http://hdl.handle.net/10012/8169.

[16] Erik G. Larsson. TSKS11 networks: Models, algorithms and applications – Supplementary notes. [unpublished], 2019.

[17] Mukwende Placide and Yu Lasheng. Information decay in building predictive models using temporal data. In *2010 Third International Symposium on Information Science and Engineering*, pages 458–462, 2010. doi: 10.1109/ISISE.2010.108.

[18] Paul A. Gagniuc. *Markov chains : From theory to implementation and experimentation*. John Wiley & Sons, Incorporated, 2017. ISBN 9781119387572 (epub). URL https://ebookcentral.proquest.com.

[19] Vlad Stefan Barbu and Nikolaos Limnios. *Semi-Markov Chains and Hidden Semi-Markov Models Toward Applications*, volume 191 of *Lecture Notes in Statistics*. Springer New York, August (printed ver.) 2008. ISBN 9780387731735 (epub). URL https://ebookcentral.proquest.com.

[20] S. Karlin and H.E. Taylor. *A First Course in Stochastic Processes*. Elsevier Science, 2012. ISBN 9780080570419. URL https://books.google.se/books?id=dSDxjX9nmmMC.

[21] J. T. Schoof and S. C. Pryor. On the proper order of markov chain model for daily precipitation occurrence in the contiguous united states. *Journal of Applied Meteorology and Climatology*, 47(9):2477 – 2486, 2008.

doi: 10.1175/2008JAMC1840.1. URL https://journals.ametsoc.org/view/journals/apme/47/9/2008jamc1840.1.xml.

[22] Bruno Sericola. *Markov Chains: Theory and Application*. John Wiley & Sons, Incorporated, July 2013. ISBN 9781118731444 (epub). URL https://ebookcentral.proquest.com.

[23] Y. Dodge, D. Cox, International Statistical Institute, and D. Commenges. *The Oxford Dictionary of Statistical Terms*. Oxford University Press, 2006. ISBN 9780199206131. URL https://books.google.se/books?id=_OnjBgpuhWcC.

[24] S. Asmussen and S. Asmussen. *Applied Probability and Queues*. Applications of mathematics : stochastic modelling and applied probability. Springer, 2003. ISBN 9780387002118. URL https://books.google.se/books?id=BeYaTxesKy0C.

[25] David Roxbee Cox and Hilton David Miller. *The theory of stochastic processes*. Methuen & Co. Ltd., 1965. ISBN 0416237606.

[26] Daniel W. Stroock. *An introduction to Markov processes*. Graduate texts in mathematics: 230. Springer, 2013. ISBN 9783642405228.

[27] Oxford University Press. Definition of ergodic, 2021. URL https://www.lexico.com/definition/ergodic. [Accessed: 2021-11-27].

[28] Nicky Case. Sight & light. URL https://ncase.me/sight-and-light/. [Accessed: 2021-04-20].

[29] Gunnar Blom, Jan Enger, Gunnar Englund, Jan Grandell, and Lars Holst. *Sannolikhetsteori och statistikteori med tillämpningar*. Studentlitteratur, Lund, 5 edition, 2005. ISBN 9144024428.

[30] Linköping University. Git repository for TSFS12 - autonomous vehicles: Planning, control and learning systems. URL https://gitlab.liu.se/vehsys/tsfs12. [Accessed: 2021-03-01].

[31] Torkel Glad and Lennart Ljung. *Reglerteori : flervariabla och olinjära metoder*. Studentlitteratur, Lund, 2 edition, 2003. ISBN 978-91-44-03003-6.