EXAMENSARBETE INOM DATATEKNIK,
AVANCERAD NIVÅ, 30 HP
STOCKHOLM, SVERIGE 2020

Long-Term Exploration in Unknown Dynamic Environments

RODRIGUE BONNEVIE

**KTH Computer Science
and Communication**

Long-Term Exploration in Unknown Dynamic Environments

RODRIGUE BONNEVIE

Master's Thesis at RPL

Date: December 7, 2020

Supervisor: Daniel Duberg

Examiner: Patric Jensfelt

Swedish title: Långsiktig utforskning i okända dynamiska miljöer

School of Electrical Engineering and Computer Science

TRITA-EECS-EX-2020:877

Abstract

In order for autonomous robots to perform tasks and safely navigate environments they need to have a reliable and detailed map. These maps are generally created by the robot itself since maps with the required level of detail rarely exist beforehand. In order to create that map the robot has to explore an unknown environment. Such activity is referred to as autonomous exploration within the field of robotics. Most research done in autonomous exploration assumes a static environment. Since most environments in the real world often changes over time an exploration algorithm that is able to re-explore areas where changes may occur is of interest for autonomous long term missions.

This thesis presents a method to predict where changes may occur in the environment using Markov chains and an occupancy grid map. An exploration algorithm is also developed with the aim of keeping an updated map of a changing environment. The exploration algorithm is based on a static exploration algorithm that uses RRT* to sample poses and evaluates these poses based on the length of the path to get there and the information gain at and on the path to the sampled pose.

An evaluation of both the mapping and exploration is made respectively. The mapping is evaluated on its ability of suppressing noisy measurements whilst being able to accurately model the dynamics of the map. The exploration algorithm is evaluated in three different environments of increasing complexity. Its ability to seek out areas susceptible of change whilst providing data for the mapping is evaluated in each environment. The results show both a mapping and exploration algorithm who works well but are noise sensitive.

Referat

För att autonoma robotar ska kunna utföra handlingar och tillförlitligt kunna navigera i sin omvärld så behöver de en pålitlig och detaljerad karta. Dessa kartor är oftast generade av roboten själv då det sällan finns kartor som uppfyller dessa krav. För att skapa dessa kartor så behöver roboten kunna utforska okända miljöer och kartlägga dessa. Detta kallas autonom utforskning inom mobil robotik. Det mesta av forskningen som är gjord inom detta antar att miljön är statisk och inte förändrar sig men eftersom detta sällan är fallet i verkligheten så kan utforskning och återutforskning av dynamiska miljöer vara av intresse, särskilt för robotar som skall vara aktiva i samma miljö under en längre tid.

Denna rapport presenterar en metod att förutspå förändringar i en miljö med hjälp av Markovkedjor och occupancy grid kartor samt en utforskningsalgoritm vars mål är att hålla en uppdaterad version av en dynamisk miljö. Utforskningsalgoritmen är baserad på en sådan anpassad för en statisk miljö som använder RRT* för att välja ut positioner och orienteringar för roboten. Dessa evalueras baserat på sträckan för att komma dit och den nya informationen som kan observeras där och på vägen dit.

En evaluering av både kartläggning och utforskning är gjord. Kartläggningen är evaluerad på dess förmåga att hantera brus i mätningarna samtidigt som den behåller en bra representation av de dynamiska aspekterna i miljön. Utforskningsalgorithmen är testad i tre olika miljöer av olika komplexitet. Dess förmåga att upptäcka och utforska områden med ökad sannolikhet för förändring och samtidigt förse kartläggningen med data för att modellera miljön och dess dynamik är det som evalueras i experimenten. Resultaten visar att både kartläggning och utforskningsalgoritmen fungerar bra men båda är känsliga för mätbrus.

Acknowledgement

I would like to thank my supervisor Daniel Duberg for all the support that he has given me throughout this project. Thank you Fernando dos Santos Barbosa for acting as a second supervisor, and a big thanks to Viktor Kull for helping me learn C++ for and shortening my debug-time by a lot.

Contents

1	Introduction	1
1.1	Research Question	2
1.2	Objective and Scope	2
1.3	Contribution	2
1.4	Societal and Environmental Sustainability and Ethics	3
2	Background	5
2.1	Mapping	5
2.1.1	Occupancy Map	5
2.1.2	Octree Occupancy Map	6
2.1.3	UFOMap	6
2.1.4	Markov Chains and Dynamic Map Modeling	7
2.2	Path Planning	8
2.2.1	RRT	8
2.2.2	RRT*	8
2.3	Exploration	8
2.3.1	Autonomous Exploration Planner	9
2.4	Related Work	10
2.4.1	Long Term Mapping of Dynamic Environments	10
2.4.2	Long Term Robotics Projects	11
3	Method	13
3.1	Dynamical Map Representation	13
3.1.1	Dynamic Map Modeling	13
3.1.2	Sessions	14
3.2	Exploration	14
3.2.1	Information Gain	15
3.2.2	Cost Function	17
3.2.3	Score	17
4	Experiment	19
4.1	Mapping	19
4.1.1	Experimental Setup	19

4.1.2	Experiments	21
4.2	Exploration	26
4.2.1	Experimental Setup	26
4.2.2	Experiments	29
5	Discussion	35
5.1	Mapping	35
5.2	Exploration	35
5.3	Conclusion and Future Work	37
5.3.1	Conclusion	37
5.3.2	Future Work	38
	Bibliography	39

Chapter 1

Introduction

Mobile robotics is the field of studying robots that are capable of locomotion and is generally done with autonomy in mind. In order for a robot to be able to reliably operate and navigate in an environment it has to have a map to for example know its pose and plan its missions. Maps that are detailed enough are often hard to provide beforehand for the robot so ideally the robot should create it itself. This is called autonomous exploration. When it comes to mapping and exploration of unknown environments the majority of research done assumes a static environment. If a robot has to endure longer missions in real world environments the assumption of static surroundings will often not hold. In a static environment the mapping will be considered finished when the whole map has been visited once. Thus the exploration algorithm will only be interested in unvisited areas. With a changing environment the problem becomes harder as the robot has to explore continuously and revisit areas to see if and how they have changed.

It is generally not efficient to revisit all areas with the same frequency since changes in environments do not tend to be uniform over the map. Efficiency is often important since many robots are limited by their battery life and have other tasks to perform besides exploration and mapping. An algorithm that is able to estimate the likelihood of which areas in the map has recently changed could then be of great service when trying to keep an updated map of the environment.

Path prediction of dynamical objects are hard and sometimes unfeasible to do. The approach of this thesis is not to model these trajectories but rather model the probability of future change in different areas of the map. This information will then be used in an exploration algorithm whose aim will be to efficiently revisit areas in order to keep an accurate and updated static map representation of the environment. This thesis approach can therefore be said to be divided into two parts, mapping and exploration.

1.1 Research Question

How to model the dynamics of a changing environment and use that information in an exploration algorithm to efficiently and autonomously keep an updated map representation of an environment for a long period of time?

1.2 Objective and Scope

The main task in this project is to create a generic algorithm that will be able to learn what and how things in its environment is changing over time. This information will then be used in an exploration algorithm in order to efficiently explore and keep a good and updated map of the environment for an extended period of time. The project will be set in a moderately large environment for which a 3D map will be created. The robot will have sensors that measures the positions of obstacles within a certain range. The mission of the robot will be of a longer time period and the robot will be revisiting areas in its exploration. Since it is a long term mission modeling of fast dynamics is not the main focus in this project. The world will be assumed to be static when the robot explores it. It is not the behaviour of dynamic objects during one observation of it that is the scope of this project but rather changes in the environment between visits of a scene.

Due to the fact that the environment is dynamic the robot can not be sure which parts of the map are free or occupied, it is therefore necessary that the path planning part is done online and in real time in order to adapt to these unforeseen map changes. To increase the autonomy of the robot a strong focus will lie on the lightweightness of the system so that it could be run onboard the robot.

Since this thesis will focus on mapping and exploration the robot's location will be assumed to be known. The algorithm will not be given any other information other than the sensor's measurements and the pose. It will not try to recognize and segment the map into different objects. It is not within the scope of this report to discuss the hardware and surrounding software used, the focus will solely be the algorithms that define and describe the dynamic parts of the unknown environment and how to perform exploration in this environment.

1.3 Contribution

The underlying problem here is that it is extremely difficult and in some case impossible to predict the movements of dynamic objects, especially if they are humans or other agents. The idea in this thesis is to keep a good static representation of the environment and update it according to observed changes in different areas. This makes the results interesting for robots that want to rely on the well researched static methods but still need to be able to handle unforeseen changes in its environment.

1.4. SOCIETAL AND ENVIRONMENTAL SUSTAINABILITY AND ETHICS

The information from the modeling of dynamical areas could perhaps be used in robot path planning if it desires to avoid areas that are susceptible to change or wants an estimation of which doors are typically open for example. The same information provided in this report may be used in localization, since it provides some information about the probability of detecting dynamical objects.

1.4 Societal and Environmental Sustainability and Ethics

With increasing demand and research for unmanned autonomous vehicles (UAV) has become a rapidly developing technology that is becoming more and more advanced and finding more use-cases with the improvements on computing power and battery capacities.

Society

With autonomous agents having long term missions and getting better at it we can expect to have more robots in our future everyday life. As with every emerging technology that can have both good and bad impacts on our society. Since many UAVs are equipped with cameras privacy is becoming an issue. Long term robotics missions can typically be applied to surveillance or in person customer service that will then be able to collect large amount of data. Surveillance UAVs can arguably make many communities safer but it is important that all the information that will be gathered are used appropriately. Another issue that is facing our societies is what the automatization of work tasks and the replacement of humans to machines in the workforce will do in our social structures.

Ethics

Military uses of autonomous agents is a classic ethic's dilemma. UAVs have already been used in military applications and is hard to determine who bears the moral obligations of when UAVs are the ones to hurt humans or damaging property. It does not even have to be in such an extreme setting to find it hard to figure out the ethics that comes with robots in our society. Who is at fault when an autonomous agent does something wrong, is it the developers? The owner? No one? Autonomous agents can also do good things such as replacing humans that work in dangerous conditions such as in search and rescue missions and in mines.

Environment

When it comes to the environment unmanned vehicles have generally a smaller environmental impact both in production and use. Autonomous agent can also be used to gather information that can be used to make industries more resource efficient for example smarter watering and pesticides control in agriculture.

Chapter 2

Background

In this chapter the theory used for the methods described in this thesis are introduced and explained in short. This chapter is divided into two different parts, one that contains the theory for the mapping algorithm and one for the exploration.

2.1 Mapping

For agents operating in any environment it is important for them to have an idea of how the environment looks like. Without any spatial awareness any task in the environment will become difficult to perform since the agent is essentially blind to what it has seen before. Without a map localization becomes an impossible task and planning towards a goal a much more difficult one.

Maps can contain any information about the environment and describes how this information is spatially located relative to other objects in the map. The most basic and common map contains information about where obstacles are in a coordinate system.

In robotics the map required by the robot to function properly is often quite detailed in the sense of that all obstacles in the environment are placed in the map. Few such maps exist beforehand so the robot in this case has to create its own, this process is called mapping and requires that the agent is capable of perceiving its environment. Sensors used for mapping are often a depth sensor like for example a LIDAR or an RGB-D camera. This report will focus on how to use occupancy grid maps when modeling dynamic behaviour.

2.1.1 Occupancy Map

An occupancy grid is a type of map representation designed to be able to cope with sensor noise. The basic idea is to divide the map into volume elements called voxels. These voxels can take on three different states; unknown, occupied and free. The idea with an occupancy grid is to represent the probability that a voxel is in each state. For example, if a voxel is measured to be occupied the estimated probability

of the voxel being occupied should increase, and decrease for the other two states since they are complementary. One common way to implement this is to have a stochastic variable for each voxel in the map. This variable's value is then increased or decreased depending on if the voxel is measured to be either free or occupied. The voxel is then said to be free or occupied if this variable is above or below certain thresholds and unknown if none of these thresholds are met.

2.1.2 Octree Occupancy Map

When mapping large environments a naive representation of an occupancy grid can be quite memory consuming and costly to search in, especially in 3D. Octree occupancy map is a way to structure the data of a 3D map, a popular implementation done by Wurm et al. [1] called Octomap is often used in mobile robotics. Octomap uses a hierarchical data structure where every node in the tree has maximum 8 children, one for each octant. Each node in the octree represent a volume in 3D space and can be recursively sub-divided into 8 voxels of equal size until the desired voxel size is reached. If all children of a node are the same it is sufficient to only store the parent node and delete all of its children, this is called pruning. Pruning makes the map more memory efficient and faster to search in compared to a naive implementation of an occupancy grid since it reduces the number of nodes without losing any information about the environment. Duberg and Jensfelt [2] build on the work of Wurm et al. [1] and models unknown space explicitly which is suitable for exploration where unknown space often is accessed. Their implementation is also faster to manipulate than Octomap. This new map is called UFOMap and are the one used in this report.

Figure 2.1: Octree occupancy map

2.1.3 UFOMap

UFOMap, as well as Octomap, uses log-odds to model the probabilities of a voxel's state. Log-odds are a function that maps probabilities from $(0, 1)$ to $(-\infty, \infty)$. The reason that UFOMap uses log-odds instead of regular probabilities is that

2.1. MAPPING

it can then add the updates instead of performing a multiplication which makes the implementation faster. UFOMap allows the user to tune the amount that are added or subtracted if measured occupied or free in order to give the map certain characteristics. Small updates for example makes UFOMap behave as a low pass filter. In order to make the map agile for changes in the environment the log odds can be clamped. The clamping thresholds are parameters in UFOMap and can also be tuned.

2.1.4 Markov Chains and Dynamic Map Modeling

A Markov chain is a stochastic process where the next state of a stochastic variable is determined only from its previous state. It typically involves two or more states and the probabilities for the stochastic variable to either change its state to one of the other states or to remain in its current state.

Saarinen, Andreasson, and Lilienthal [3] introduce a way of modeling the dynamics of an environment. Here the dynamics of the environment is modeled as the probability that a voxel or cell changes its state from free to occupied or vice versa next time it is observed.

Figure 2.2: Markov model describing the dynamics of a voxel.

This probability of change are modeled with Markov chains. As the model only deals with space that have already been explored it will only consider the free and occupied cell states. A graphical representation of the model can be seen in figure 2.2. The model describes the probability for a voxel to change its state from free to occupied p_{fo} and occupied to free p_{of} . As the probability of remaining in a state is complementary to switching state only these two probabilities have to be estimated in order to model the behaviour of the voxel.

The probability that the voxel's state switches from occupied to free is estimated to be $p_{of} = \frac{\#occupied \rightarrow free}{\#occupied}$. The nominator here is the number of times the voxel has been observed to switch from free to occupied and the denominator the amount of times it has been observed as occupied. A similar approach is done when estimating that a voxel goes from free to occupied $p_{fo} = \frac{\#free \rightarrow occupied}{\#free}$.

2.2 Path Planning

When planning a path between a goal and a starting pose in an known environment a planning algorithm is needed. Two popular planning algorithms are RRT and RRT*. Both of these are sampling based, meaning that they sample intermediate goals and links them together for a complete path from start to finish.

2.2.1 RRT

RRT or Rapidly-exploring Random Trees is an algorithm often used in path planning. It incrementally expands a tree structure that fills the space in a stochastic or semi stochastic manner. In a path planning scenario RRT starts at a pose, typically the robots current pose, from that position it expands a node in the direction of either a random point, pose or a desired goal at a certain distance. As the tree grows RRT creates this extension from the node in the tree closest to the generated point. The process of expanding the tree is then done until one of the nodes is within a desired distance of the goal. As constraints can be added when creating a new node, such as the feasibility of navigating between the two nodes or proximity to obstacles the obtained path can be guaranteed to be safe and feasible. However, the path is not likely to be optimal due to the injected randomness and that the algorithm only connects the new nodes to its nearest neighbour.

2.2.2 RRT*

RRT* is an optimized version of RRT that if it has an infinite number of nodes finds the optimal path. Although this scenario is practically infeasible RRT* generates in general more optimal paths than RRT. RRT* introduces two new steps in the tree building. The first is when deciding where to attach the new node in the tree. Instead of just considering the closest node RRT* considers the closest path in the tree. This change will make the tree structure simpler since longer and more convoluted branches will be dismissed.

The other change is the reevaluation of the vertexes in the tree. Since RRT* consider the distance in the tree all nodes have a cost assigned them. If the tree can be rewired so that these cost decreases a more optimal route will be found. This makes the paths found by RRT* simpler and shorter since this improvement finds shortcuts in the tree.

2.3 Exploration

Exploration of unknown environments can be viewed as a optimization problem where the agent is to discover as much new information about the environment while minimizing for example the time or distance traveled to do so. In most cases in exploration information that the algorithms want to gain is whether an element of the map is either free or occupied. Exploration is often done with no a

2.3. EXPLORATION

priori information about the environment. There are two common approaches to this problem. One called frontier exploration, performed by González-Banos and Latombe [4] amongst others, plan their paths towards the border between free space and unknown space in order to explore and map the whole environment.

The other is called sampling based exploration. Next best view (NBV) in an exploration context is a sampling based planning algorithm that tries to solve the problem of deciding where it is best to go next when exploring. NBV samples a number of candidate poses and calculates how much new information the agent would gain at these points.

For example the NBV exploration algorithm presented by Bircher et al. [5] builds an RRT at the robots current pose to represent collision free paths for exploration. Each node in the tree is evaluated on the how much unknown space that can be observed there and the length of the path that it takes to reach the node. It is common for NBV planners to evaluate their sampled poses with a gain and cost function that together becomes a score for the pose.

2.3.1 Autonomous Exploration Planner

Although frontier based exploration are efficient they can have some problem when faced with larger environments due to their tendency to jump between unexplored areas. Sampling based algorithms can also have problems in larger environments since it can be costly to sample poses and find paths far away enough in large or complex environments.

Selin et al. [6] proposes an exploration algorithm that fuses both frontier and NBV planning. They use an NVB planner when exploring locally and a frontier exploration when larger distances has to be covered before new information is obtained. The exploration algorithm developed in this report is based on their work but does not make use of the frontier based part since it is not of great use in a re-exploration scenario because few or none of these borders exist.

In Selin et al. [6] points for the NBV planner are sampled by expanding an RRT* of fixed size from the agent's current pose. At each node in the tree the optimal orientation of the robot is calculated by selecting the yaw angle where the most unknown space is observable. This is then considered the information gained at that node. The cost of arriving at each node is also considered and added to the gain to create the information score. The cost function serves as a way of evaluating nodes with similar gain depending on how far away they are. It is more desirable that the robot seeks out gains closer to its location, since it most likely has to return there afterwards if it does not.

As proposed by Bircher et al. [5] information along paths are more interesting than just considering poses so the parent's score are added to the children's score in the tree. The node with the best score is then chosen and the agent navigates to the first node in that branch. The reason for not navigating all the way is that new information might have been obtained so that it not the best node anymore. When arriving at the first node in the branch containing the best node the tree is

reexpanded but the best branch is kept. The information gain for this branch is recalculated and the process is repeated.

The total information gain $g(x)$ at pose x can be viewed as the total volume of unmapped space that is within sensor range and not occluded by occupied space.

The cost of reaching a node in the tree is denoted as

$$c(d) = \exp(-\lambda d) \quad (2.1)$$

where d is the euclidian distance from the parent node to the current node in the tree and λ is a tuning parameter. As λ gets larger the more thoroughly the agent will explore the nearby area.

The score that all poses in the tree are evaluated against is calculated in the following fashion

$$s(x) = c(\|x_{parent} - x\|)g(x) + s(x_{parent}) \quad (2.2)$$

where x_{parent} is the pose of the nodes parent node in the tree.

2.4 Related Work

2.4.1 Long Term Mapping of Dynamic Environments

Many different approaches have been attempted when trying to describe dynamic environments. Hähnel et al. [7] filter out dynamic objects from the environment by trying to model the probability of measuring something dynamic given previous measurements. It is developed for the problem of performing localization in a dynamic environment. Dynamical parts are often hard to model and problem occur when trying to localize the agent's pose with respect to those objects. Their proposed solution is to remove the measurements of dynamical objects in order to localize using just the static object and thus being able to use a simpler localization algorithm.

Saarinen, Andreasson, and Lilienthal [3] estimates the probability of an independent cell in an occupancy grid to change from occupied to free and vice versa with independent Markov processes. A classification of a cell's dynamics is done based on the estimated probabilities of remaining and changing the state of a voxel. In order to be agile to changes in the environment's dynamics Saarinen, Andreasson, and Lilienthal [3] also uses recency weighting. Wang et al. [8] uses hidden Markov models but considers the states of the neighbouring cells in order to model the motion patterns of the dynamical objects. This method works on a smaller timescale and assumes that the robot is able to keep the objects in view and measure fast enough to reason about its movement. Rapp et al. [9] uses the same Markov model as Saarinen, Andreasson, and Lilienthal [3] to model the likelihood of a cell switching states. It however extends the Markov model and estimates the time that a cell or voxel remains in a static state.

2.4. RELATED WORK

Modeling dynamics with recency weighting is something that Biber, Duckett, et al. [10] and Arbuckle, Howard, and Mataric [11] both does. By taking the recency weighted average of previous measurements at different time scales Biber, Duckett, et al. [10] creates multiple occupancy grids that model changes in the map at different time scales. Stachniss and Burgard [12] model semi static objects, for example doors, that can move but are standing still the majority of the time by creating submaps that are segments of the map where those semi static changes have happened. The submaps contains information of the different configurations of these has been observed in so that the robot can choose the most suitable submap when it observes the location.

Krajnik et al. [13] addresses the possibility that certain changes in the environment are periodical. The authors model these periods using Fourier transforms, this require however that the measurements are taken at constant time intervals which may not always be practical for a mobile robot. Santos et al. [14] introduces a spectral analysis that is close to Fourier transforms that but allows for samples to be taken at a variable rate.

Another approach presented by Rosen, Mason, and Leonard [15] models the time that a certain feature in the map persist over time. This is done with survival analysis, a branch in statistics that estimates the time it will take before an event of interest takes place. Ambruş et al. [16] introduces the term meta-rooms. Meta-rooms are a cluttered representation of the static objects in a room and the authors of the paper develop an algorithm for segmenting out the static objects. This is done for a long term scenario when the robot revisit the same area multiple times. By reasoning about how similar previous measurements are compared to the latest and if the new or missing objects have been occluded by some other object these are then added as static objects in their meta-room. Ambruş et al. [17] extends the meta-room idea and clusters measurements together to dynamic objects and tries to identify and describe their behaviour over time. Both these papers assumes that the measurements are taken from the same viewpoint in the map.

2.4.2 Long Term Robotics Projects

A survey conducted by Kunze et al. [18] of long term autonomous robot missions characterizes their different approaches and methods and compares them in what the different projects set out to achieve. One long term robotics project is the CoBot project by Biswas and Veloso [19] which uses robots that collaborates with each other. The robots does not use an occupancy grid or a topological map as the previous mentioned methods uses, but instead a vector map that consists of line segments of different lengths. The robots utilizes the architectural plans in order to create its map hence it does not require any exploration. The vector map is used for localization and is considered to be static. For path planning the collaborative robots uses a topological map that is also based on the blueprints of the building and are static. The robots deal with the dynamic parts in the environment by having an obstacle avoidance algorithm that is employed when the robots are moving. The

robots does not try to model these dynamical parts of the map and instead deals with them when it encounters them.

The STRANDS project [20] is a joint project between multiple universities and researchers funded by the European Union. It is a long time autonomous robotics project for security and care in indoor environments. One of the deployments Hanheide, Hebesberger, and Krajník [21] takes place in a care home and provides an infoterminal for its inhabitants. This deployment project studies, amongst other things, where to be at what time in order to be of use for as many people as possible. This is done by trying to find the underlying periods in the probability of being useful at different areas in the care center. The method is explained in more detail in [13].

Chapter 3

Method

This chapter introduces the method presented in this thesis and the algorithms that it consists of. It will begin to describe mapping and representation of a dynamic environment and then move to exploration of dynamic environments.

3.1 Dynamical Map Representation

There are multiple use cases for dynamical maps. For example on an autonomous car it could be interesting to model the trajectories of other cars and pedestrians. But also to model the probability of obstacles ahead such as parked cars or road blockages. The trajectory modeling will only be useful on a shorter time scale compared to the obstacle modeling. The method presented in this report deals with the latter more long term problem.

A good example of a use case for this algorithm is the map for a robot that remaps a warehouse during the night so that this map can be used by other robots to plan their paths more effectively during the day.

The objective of this map is to have an accurate static representation of the environment while also model where the environment is likely to change in the near future. The algorithm's aim is to get a high level overview of the long term dynamics of the environment. The assumption that the environment is static when exploring but dynamic when not exploring is made as the proposed solution models the probability that a voxel changes its state between visits of a scene. The map used are based on an octree occupancy grid developed by Duberg and Jensfelt [2] as described in section 2.1.1 but contains further information on the dynamics of the environment based on the work of Saarinen, Andreasson, and Lilienthal [3].

3.1.1 Dynamic Map Modeling

The environment's dynamics are modeled as proposed by Saarinen, Andreasson, and Lilienthal [3] and described in Section 2.1.4.

The initialization of the probability that a voxel changes its state are initialized as $\frac{1}{2}$. This is done by initializing the number of observations of the voxel switching its state to 1 as well as the number of observations in each state. Then as the voxel is observed for its first time the probability becomes $\frac{1}{2}$. Thus the probability that the voxel changes its state from free to occupied, p_{fo} , is estimated to be $\frac{\#free \rightarrow occupied + 1}{\#free + 1}$ and from occupied to free, $p_{of} = \frac{\#occupied \rightarrow free + 1}{\#occupied + 1}$, as described in [3].

UFOMap [2] is also modified to accommodate the modeling of dynamics. Pruning, the merging of children in the tree based on their similarities, is turned off. This is because it is not straightforward how the dynamic parameters should be pruned since it is not that likely that voxels will have the exact same probabilities and averaging of these parameters might make the algorithm harder to evaluate. The absence of pruning will not have a big impact on this project since speed is not a limiting factor.

3.1.2 Sessions

The map is intended to model the change between visits of a scene. If a voxel is observed multiple times in a visit it is not intended for it to update the dynamics of that voxel more than one time per session. This follows from the assumption that the world is static when exploring. The concept of sessions is therefore introduced. Sessions are an exploration loop limited in time or space where the static voxel parameters (occupied, free and unknown) are updated continuously but not the dynamic ones (p_{of} and p_{fo}). Each voxel observed in a session is placed in a container where its state before the session is stored beside its current state. At the end of each session these voxels are updated as a batch where these two states are compared to determine if the voxel has changed between the two sessions.

3.2 Exploration

When exploring static environments the main goal of the exploration is to discover as much unknown space as possible. However in the case of exploring changing environments revisitation of previously explored areas are a central part. The goal of the exploration here is to explore areas that are likely to change but also making sure that it knows where change is happening. In order to do so it has to balance visiting dynamic parts of the map and revisiting static areas to check if they have become dynamic. If the algorithm only seeks out the dynamical parts it is possible that it will neglect areas that are initially static. With few measurements the estimation of the dynamics have large variances and to make sure that the agent has a map that is as good as possible the exploration has to make sure that all areas are revisited on a regular basis.

The path planning and pose sampling for the proposed exploration algorithm in this report is based on the work of Selin et al. [6] which is described in Section 2.3.1. In this section the modifications of this algorithm will be described.

3.2. EXPLORATION

3.2.1 Information Gain

Information gain is the amount of relevant knowledge that is obtained at a certain pose or path. The types of information that is considered in the proposed exploration algorithm can be divided into three different categories and are presented below in order of importance.

1. New information
2. Dynamic information
3. Overall revisitation

New information includes all previously unknown voxels, if the agent has the possibility of discover something new. For example, if a door to a previously closed room has suddenly opened the agent should be highly encouraged to explore it. This part of the information gain is kept as in the original algorithm.

Dynamic information are areas where changes are likely to happen. In order to have an as good as possible map of the current state of the environment the agent should seek out these areas.

Overall revisitation is important in order to have a good and unbiased dynamic map of the whole environment. Here the idea is to encourage visiting areas that have not been visited for a long time. If this metric is not included in the information gain it is likely that certain areas will be forgotten as they have previously been observed as static and thus be of low interest. The main problem with this is that the estimation of map dynamics is dependent on the quantity of measurements, if few measurements are taken the uncertainty in estimation will be high. Without this metric the exploration will also not be as agile to changes in dynamic behaviour.

The total information gain is a combination of all these three types. This total information gain $g(x)$ for a point x can be viewed as a concentration of information throughout space, and when integrated over an observable area or volume it gives the total information gain for that pose. To avoid revisitation of areas during the same session, space that have previously been observed in the current session will not be considered in the information gain until next session.

The gain function can be written as

$$g(x) = \alpha(x_u) + \beta(x_f, x_o) + \gamma(x_{lastseen}) \quad (3.1)$$

where α , β and γ are functions. x_u, x_f and x_o are booleans for which state this point is in; unknown, free and occupied respectively and will thus be 1 if true and 0 otherwise. $x_{lastseen}$ represent how long ago the point was last observed. For simplicity the information concentration for unknown space is set to 1 and the other functions weighted relatively accordingly to their importance.

$$\alpha(x_u) = x_u \quad (3.2)$$

The dynamic information is based on the map's estimation of the likelihood that the map would change in that point next time it will be observed. The map (described

in section 3.1) estimates these probabilities on a voxel level, p_{of} if the voxel is occupied and p_{fo} if the voxel's state is free.

In order to obtain the information gained from a pose the information concentration needs to be integrated over a volume. For occupied space it is ambiguous of how large the size of that volume is. In this case the map is modeled by voxels and depth into occupied space was chosen to be a voxel width. Since the observable volume will in most cases contain more free voxels than occupied ones it is necessary to weigh them accordingly. A straightforward solution to this is to normalize the part of the information gain concerning dynamic free space. This is done so that a ray cast from the agent to maximum sensor range in free space has the same information gain as a hit on occupied space. The dynamic gain is given by

$$\beta(x_f, x_o) = \varphi x_f * p_{fo} + x_o * p_{of} \quad (3.3)$$

where $\varphi = \frac{map_{res}}{r_{max}}$ and map_{res} is the voxel width and r_{max} is the maximum sensor range. For overall coverage it is important that no area is left unseen for a long period of time, therefor a part of the information gain should be dependent on how long ago this point was seen. It is also important that areas that are hard to reach do not have disproportionately large gains. Therefore a sigmoid function is chosen in order to assure that the gain stays between reasonable limits and that gain is low for recently visited areas.

$$\gamma(x_{lastseen}) = \frac{1}{\exp^{-a(x_{lastseen}+b)} + 1} \quad (3.4)$$

where a decides the slope and b the offset of the sigmoid.

Scalar weights w_α , w_β w_γ are then multiplied with the different sub-functions in order to make the behaviour of the algorithm easier to tune

$$g(x) = w_\alpha \alpha(x_u) + w_\beta \beta(x_f, x_o) + w_\gamma \gamma(x_{lastseen}) \quad (3.5)$$

If there is noise in the agent's localisation or sensor measurements space near occupied space will tend to be modeled as dynamic. These areas are not particularly interesting to explore and should thus not be included in the information gain. So when calculating the information gain the dynamic properties of free voxels that are close to occupied space are not taken into account but the dynamic proprieties of occupied space is. Space behind occupied space will not contribute to any information gain since it will be occluded from the sensors and is thus also removed from the information gain.

3.2. EXPLORATION

Figure 3.1: Space that is considered to contain information for a 2D case. Here an object with black borders are being observed by an agent with a field of view, the circle segment. The area considered containing information gain are shown in yellow and the area that is not, orange.

In figure (3.1) an example of this is shown. The figure represent a 2D case where the circle segment is the field of view of the agent, yellow space is the area that the information gain is considered and orange space is the part of the field of view that is excluded either because it is occluded or because it is free space close to occupied space. This exclusion of nearby free space can be seen as an inflation of all obstacles when calculating information gain from probabilities that free space have become occupied.

3.2.2 Cost Function

It is important that the sampled poses are not just evaluated on the information gained at them but also the cost to arrive at them. Otherwise the algorithm will likely be very greedy and tend to jump between different areas of interest resulting in a inefficient exploration. The cost c is the same as described by Selin et al. [6],

$$c(d) = \exp(-\lambda d) \quad (3.6)$$

where d is the euclidian distance from the parent node to the current node in the tree and λ is a tuning parameter. As λ gets larger the more thoroughly the agent will explore the nearby area.

3.2.3 Score

The score function is also unchanged from the algorithm presented by Selin et al. [6].

$$s(p) = c(||x_{parent} - x||) \int_{fov} g(x) dv + s(p_{parent}) \quad (3.7)$$

CHAPTER 3. METHOD

where p is the pose where the score is evaluated at, p_{parent} is the pose of the parent node, x denotes the position of the poses and fov is the field of view at the yaw that generates the largest gain for the agent at pose p .

Chapter 4

Experiment

In this chapter the experiments and experimental setup will be presented and explained. There are two different algorithms that have been developed in this report, one mapping algorithm and one exploration algorithm. The mapping algorithm works standalone but the exploration is built on the mapping. Hence the mapping algorithm is evaluated independently and the exploration algorithm together with the mapping algorithm.

All exploration experiments are made in the simulation environment Gazebo in order to have more control of the noise and to facilitate the refurbishment of the environment.

4.1 Mapping

4.1.1 Experimental Setup

Figure 4.1: Sensor setup for mapping experiments.

The mapping algorithm is evaluating by having a simulated depth sensor looking at a wall that have a certain probability of appearing or disappearing in front of a background. The information from the simulated sensor is relayed as point clouds

to the mapping algorithm. The sensor origin is where the coordinate axes are in the figure 4.1.

The mapping algorithm is evaluated based on how well it is able to handle measurement noise. Noise can be added to these sensor measurements both in the form of sensor noise where all points in the point cloud have their own independent noise and localization errors where all points share the same offset. The noise added to the virtual sensor is gaussian and its magnitude is expressed by the ratio between its standard deviation and the voxel size of the map.

Figure 4.2: Occupied voxels when the wall is present.

The reason for the larger wall behind the evaluated wall seen in figure 4.2 is in order to set the voxels around the smaller dynamic wall to free. UFOMap casts rays from sensor's origin to the points in the point cloud that represent occupied space. This background wall is then needed in order to set the voxels surrounding the smaller wall to free. The smaller wall is 1 by 10 by 10 voxels large. The estimated parameters of all voxels at the location of the dynamic wall are averaged for all voxels in the wall and recorded for each session along with the ground truth value.

The map will be evaluated on how UFOMap parameters related to noise suppression affects the modeling of dynamic parameters. This will be done on both static and dynamic objects. The voxel layers in front and behind the wall are evaluated in order to capture the impact of noise on nearby voxels and at the object.

The magnitude of the noise added to the measurements are describes as the ratio between the smallest voxel size in the map and the standard deviation of the gaussian noise. A noise ratio of 0.5 means that the voxel size is twice as large as the standard deviation.

As described in section 2.1.1 the state of a voxel are described by the value of a variable stored in the voxel. Each time an observation is made of a voxel's state this variable is updated accordingly, the value describing the state of the voxel is also clamped so that the map is agile to changes in the environment. UFOMap allows us to tune how this variable should be updated upon measurements, if the measurements have low fidelity for example the updates are then small so that a change of a voxel's state requires multiple consistent measurements.

4.1. MAPPING

	Fast	Default
Occupied measurement	100%	70%
Free measurement	30%	40%
Occupied threshold	87%	97%
Free threshold	32%	12%

Table 4.1: Table of parameter settings in UFOMap.

A comparison between two different UFOMap settings is made, UFOMap’s default parameters and a setting with more fidelity in its measurements that is called the fast setting. The update steps for free or occupied measurements and the clamping threshold are displayed in table 4.1. The fast setting will change a voxel’s state instantly if it is measured to be occupied while previously being free however some noise suppression characteristics is added to the event of voxels going from occupied to free. This makes the fast map quick to respond to changes in the environment but is also more susceptible to noise. The default setting also models measurements of occupied space to have higher fidelity than free space. The default map settings also have measurements modeled with lower fidelity than the fast map making the map react a bit slower to change while also suppressing noise, like a low pass filter.

As the purpose of the map might not only be to model the dynamics of the environment but also model free and occupied space the default map settings might be advantageous for the way it filter out high frequent noise and favours voxels being occupied.

4.1.2 Experiments

The first experiment of the mapping algorithm is an evaluation of its dynamic parameters when looking at a static wall with 100% chance of being present. Sensor noise is added to the measurements. The dynamic parameters of the voxels inside (wall), the layer in front (1F) and two voxels in front (2F) of the wall are evaluated.

(a) Free to occupied

(b) Occupied to free

Fast map settings

(c) Free to occupied

(d) Occupied to free

Default map settings

Figure 4.3: A static wall with noise to voxel size ratio of 0.5. Evaluated at the wall, one (1F) and two (2F) layers in front of the wall. The dashed lines are the ground truth of the voxels inside and surrounding the wall. The two plots to the left display the probabilities that voxels changes their state from free to occupied and in the right plots from occupied to free.

In figure 4.3 the average dynamic value of the voxels inside and in front a static wall are displayed. With no sensor noise the probability of that something enters the voxel will be 1 for wall voxels and go towards 0 for those in front of the wall. The probability of going from occupied to free will go towards 0 for wall voxels and be unmeasured and stay in their initial states (0.5) for the voxels in front since those voxels will never be in an occupied state. As demonstrated by the dashed lines.

In these figures it can be seen that the map with the default parameters are much better at suppressing noise in front of the wall. This is also true for the estimated probability of switching form occupied to free however these are not equally important to model since the voxels in front of the wall rarely are occupied as can be seen by the slow convergence. For the wall voxels the default map has

4.1. MAPPING

a higher accuracy for modeling the switching from occupied to free but a lower for free to occupied here. Again, occupied to free is more important since these voxels are much more likely to be occupied.

(a) Free to occupied

(b) Occupied to free

Fast map settings

(c) Free to occupied

(d) Occupied to free

Default map settings

Figure 4.4: Dynamic wall with 50% chance of appearing or disappearing with different ratios between noise and voxel size. Only the voxels in the wall are evaluated. The dashed line is the ground truth of the voxels. The two plots to the left display the probabilities that voxels changes their state from free to occupied and in the right plots from occupied to free.

In figure 4.4 a dynamic wall with 50% probability of being present is observed and evaluated for the voxels inside the wall. Different magnitudes of sensor noise is added to the measurements and displayed as an unique plot in the figure. As can be seen in the figures the probability of that a voxel's state goes from occupied to free is modeled pretty equally with quite low errors but the estimation of free to occupied are significantly worse for the map with default parameters. The estimation of occupied to free are less noise sensitive to the low pass characteristics of the map with default parameters. This is because of when the object is not present the

area of where the object appears will all be measured as free since the depth sensor measurements are further back. However when the object is present sensor noise will be impacting the voxels around the object. As the map with default parameters acts like a low pass filter it will remove some measurements of voxels being occupied thus lowering the modeled probability of switching from free to occupied.

Fast map settings

Default map settings

Figure 4.5: Dynamic wall with 50% chance of appearing or disappearing. Here the ratio between noise and voxel size is 0.1 and only the surrounding voxels are evaluated. The letter denotes if the voxels are in front or behind the wall and the number how many layers from the wall they are. The dashed line is the ground truth of the voxels. The top two plots are with the fast map settings and the bottom two with default. The two plots to the left display the probabilities that voxels changes their state from free to occupied and in the right plots from occupied to free.

In figure 4.5 the surrounding voxel layers in front and behind the same dynamic wall discussed above are evaluated. The ratio between noise and voxel size for these measurements are 0.5. As can be seen here the map with fast parameters describes the dynamics of the object better but also captures the noise in the surrounding

4.1. MAPPING

voxels thus making the dynamic object appear larger and more dynamic than it is. With default parameters the noise in the voxel layers in front of the object are suppressed especially for the free to occupied parameters. However as mentioned above the probability of switching from free to occupied in the wall are quite heavily underestimated. In the layers two voxels away from the wall the parameters are converging slowly the reason for that is that these layers have a sparse amount of measurements. The voxels behind the wall have larger errors than the voxels in front. This is because of sensor occlusion. This means that these layers are of less importance of modeling accurately because of the rarity of voxels being in these states.

map		Probability of being present			
		15%	50%	85%	100%
fast	free to occ	2.6%	6.5%	15%	14%
	occ to free	2.1%	7.6%	12%	15%
default	free to occ	9.5%	23%	41%	37%
	occ to free	0.49%	4%	5.2%	5.7%

(a) Voxels in the dynamical object

map		Probability of being present			
		15%	50%	85%	100%
fast	free to occ	5.1%	15%	25%	30%
	occ to free	45%	34%	24%	19%
default	free to occ	0.047%	0.18%	0.33%	0.46%
	occ to free	3.3%	9.3%	11%	13%

(b) Voxels one layer in front

Noise to voxel ratio is 0.5

map		Probability of being present			
		15%	50%	85%	100%
fast	free to occ	6.5%	17%	35%	37%
	occ to free	4.8%	19%	30%	37%
default	free to occ	15%	39%	73%	74%
	occ to free	8.6%	17%	34%	43%

(c) Voxels in the dynamical object

map		Probability of being present			
		15%	50%	85%	100%
fast	free to occ	6.9%	20%	34%	41%
	occ to free	45%	29%	18%	10%
default	free to occ	0.024%	0.094%	2.0%	2.6%
	occ to free	14%	17%	15%	11%

(d) Voxels one layer in front

Noise to voxel ratio is 1

Table 4.2: Estimation errors of dynamic map parameters. The experimental setup is the same as previous experiments. Each column represent an experiment with where the probability of the dynamic wall being present varies. The states with the most measurements are highlighted in green. The two tables at the top have noise with a ratio of 0.5 added to the measurements and at the bottom two tables the noise ratio is 1.

Table 4.2 displays the errors of the dynamic parameters for dynamic objects with varying probabilities of being present. Each object has been observed for 200 sessions and the error displayed in the table are the mean value of sessions 150-200. In 4.2a and the voxels of the dynamical object are evaluated and in 4.2b the voxels one layer in front. The noise added is 0.5 times the voxel size and the parameters that have most data are highlighted in green. The non highlighted cells are of less importance since for example it is less likely that a voxel inside a static wall is free and thus it is more important that the algorithm has a lower error for the more

likely states.

Here as has been seen before the default map are significantly worse at modeling the probability of going from free to occupied and significantly better at modeling voxels near dynamical objects. Hence default map parameters would be better for environments without elements with high probability of switching from free to occupied but with instead semi static elements that are mostly present and static elements. For example modeling how likely doors are open without having lots of noise in front of static objects. Default parameters also models occupied to free probabilities for static and semi static objects better than fast parameters.

In table 4.2c and 4.2d the errors are quite large and the results of the different maps are quite similar except that default again has larger errors for the estimation of voxels going from free to occupied. However default still manages to model free voxels in front of all objects with high accuracy. Overall the modeling of dynamical parameters are quite bad with higher noise ratio which might require in some use cases that a dynamic map with coarser resolution is run in parallel with the map used for mapping free and occupied space.

4.2 Exploration

4.2.1 Experimental Setup

The exploration algorithm aims to explore multiple different aspects of the map at the same time. As mentioned in section 3.2.1 it has three main focuses. Exploration of unknown space, re-exploration of areas that are likely to have changed since its last observation and overall re-exploration of the entire environment. The exploration is evaluated with the default map settings described in section 4.1.1. The reason for this is because this setting is the more realistic to use in a real world scenario of the two different settings evaluated in the mapping experiments.

Figure 4.6: Environments where the exploration algorithm is evaluated.

Three different environments, shown in figure 4.6, have been chosen to evaluate the exploration algorithm in. The agent begins and ends each of its exploration sessions

4.2. EXPLORATION

at its home position, the red dot in the map as seen in figures 4.6.

For each environment three different setups are tested. One where the environment is empty as in figure 4.6 and no additional obstacles have been added. This is to show how well the exploration is able to re-explore the environment and will serve as a comparison to the other two scenarios. The second and third setup will be with dynamical objects placed in the environment and with the dynamical part of the gain turned off in the first run and then turned on in the last run. This is to show the difference between how the exploration decides to re-explore the environment depending on its objective.

The first environment is just a big empty room without any inner walls or other static objects, see figure 4.6. The objective here is to show how the algorithm behaves in a simple environment.

The second environment in figure 4.6 is in a more complex setting. The aim is to show the algorithm's behaviour when walls and rooms are present. The environment consists of four rooms placed in a symmetrical manner around a small room in the center. The idea here is that the session length will be just long enough for it to explore one of the four room thoroughly each session. This will make the UAV at the start of each session choose which room to visit and thus leading to results that are more easily interpreted.

The third environment is an empty apartment to show how it behaves in a nonsymmetrical and more complex environment that is closer to a real world environment. Here again the three different setups of objects in the environment will be evaluated.

Figure 4.7: The three dynamical objects used in the experiments

The dynamical objects added to the environment are boxes of various sizes shown in figure 4.7, each box has a certain probability of appearing or disappearing at the beginning of each session. The reason for the object being boxes is that they are simple objects that have a large surface area that are easily observed from any point of view. All objects have a 50% chance of appearing or disappearing to create the maximum level of dynamicality. This allows the algorithm to focus just on detecting and re-exploring highly dynamical areas instead of weighing the importance of smaller or less dynamical objects which will be harder to evaluate.

(a) One empty room (b) Four symmetrical rooms (c) Apartment

Figure 4.8: Placements of dynamical object in each environment

The dynamical objects are placed so that it will be clear if the algorithm prefer to explore certain areas depending on their dynamicality.

For all exploration experiments the agent’s localization is set to be perfect. Since the mapping algorithm is noise sensitive as little noise as possible is desirable. However noise-free simulations were not achievable in this setup due to interpolation and time errors in the transformations between different coordinate frames.

Figure 4.9: Color representation of voxel visitation ratio.

The exploration algorithm is evaluated on how often it visits different areas in the environment. It is therefore recorded in how many sessions each voxel has been observed during all sessions. This number is divided by the total number of sessions that has elapsed which gives a ratio of how often a voxel has been visited. In order to visualize this visitation ratio they are segmented and represented as different colors shown in fig 4.9 depending on how many times they have been observed. A value of 0% means that the voxel has not been seen in any of the sessions whereas 100% means it has been observed in every session. In order to visualize this 3D volume of visitation rates a horizontal slice of the map is shown which gives the impression of a visitation heatmap over the environment.

Each experiment was two hours of simulated time which result in about 100-150 sessions. The reason for results not being averaged between different runs is the ambiguity of doing it in a map where voxels values might be conflicting and the realization that most experiments converges to the same result after the selected experiment length. It is better and easier to run longer once than shorter multiple times since the algorithm is dependent on data to accurately model the dynamics of the environment. And as can be seen in section 4.1.2 about 50 observations are enough for the dynamical parameters in the map to converge thus 100-150 sessions

4.2. EXPLORATION

should be enough for the exploration algorithm to converge.

4.2.2 Experiments

Figure 4.10: Visitation rates for exploration in an empty room without any objects.

The first experiment is set in an empty square room without any walls or objects in it. As can be seen in figure 4.10 no particular area of the room is preferred by the algorithm. The reason why the corners of the room are observed less is that they contain less volume of non occluded voxels and are therefore not as interesting. This is wanted since it is not really desirable that the robot looks inside corners, it is better for the agent to look elsewhere where there are more volume to explore.

For the second experiment in this open room environment three objects have been added in the top left corner. In the first part of the second experiment these objects are kept static to make a fair comparison between the case when the dynamic gain is enabled and not. This is because the corner with objects will be harder to observe than the other corners and the algorithm will have to make more effort in exploring it resulting in a slight different pattern seen in figure 4.11a compared to figure 4.10. In the second part of this experiment the objects are instead highly dynamical with each a 50% chance of either appearing or disappearing. In figure 4.11b it can clearly be seen that the upper left corner is much more frequently visited than the other corners meanwhile the rest of the environment is not neglected and evenly revisited.

The first experiment in the larger symmetrical environment, displayed in figure 4.6b, is without any objects and without any dynamic information gain just to show that the algorithm is able to create an unbiased exploration pattern in the environment. The result of this experiment can be seen in figure 4.12 and shows that the space is re-explored evenly and the majority of the rooms are visited every fourth session.

Figure 4.11: Visitation rates for exploration in a room with objects being static or dynamic.

For the second part of the experiments in this environment objects have been added to the leftmost room. In the first run the dynamic information gain is turned off in order to show how the algorithm re-explore the space with obstacles in one room and in the second run the dynamic gain is turned on in order for the agent to seek out dynamic parts of the environment. As can be seen in figure 4.13b the leftmost room is not visited that much more, especially the back of the room, though it contains dynamic objects. The overall exploration has also decreased. The reason for that is believed to be localization noise that makes the walls appear dynamic and thus spreading out the dynamic gain all over the map.

It can be seen in figure 4.13b in how the robot in almost every session has explored the room in the center thoroughly. This room does not contain any dynamical objects but is a room that will get much coverage by the fact that the drone starts and ends every session in the center of this room. However it here gets more coverage compared to the experiment with static objects in figure 4.13a which is a bit curious. Figure 4.14 displays the estimated probabilities of change of the map at the end of the of the experiment with dynamic objects. In these figures it can be hard to see how often each voxel is measured in each state. This needs to be taken into account to determine how dynamic the voxels really are modeled to be. One example of this is the free voxels in figure 4.14b which all can appear to be modeled as highly dynamical but really are not since these voxels seldom or never have been measured in an occupied state, which means that they will still have their initial

4.2. EXPLORATION

Figure 4.12: Visitation rates for exploration in the larger symmetrical environment without any objects present.

probabilities. This can be confirmed by looking at these voxels estimated probabilities for switching from free to occupied which are very low indicating that they are mostly, or always, modeled to be free.

With this in mind when looking at the walls in the environment most wall voxels appear to be mostly in occupied state and modeled as static occupied voxels except for the walls between the center room and its adjacent rooms. Some of these voxels lowest probability of change are over 25% for both states which will be considered dynamical by the exploration algorithm despite the fact that they really are static. The reason for that is thought to be due to small localization errors in yaw when the robot is rotating and explains the exploration behaviour in figure 4.13b.

The third set of experiments is set in a more complex environment seen in figure 4.15. This environment is an apartment and is trickier to navigate in than the previous environments. Similarly to the previous two experiments it is first shown that the agent is capable of keeping an even revisitation rate when the objects in the environment are both static and dynamic. Again it can be seen that the algorithm prefers the dynamic room when the dynamic gain is turned on.

Figure 4.13: Visitation rates for exploration in a symmetrical but more complex environment with objects placed in the leftmost room being static in the left image and dynamic in the right.

4.2. EXPLORATION

Figure 4.14: Probabilities for a voxel to change its state at the end of the of an exploration experiment with dynamic objects present in the leftmost room. This is the same experiment that is displayed in 4.13b.

Figure 4.15: Visitation rates for exploration in a more complex environment with static and dynamic objects. The objects are present in the leftmost room.

Chapter 5

Discussion

5.1 Mapping

The results of the mapping algorithm shows that it is able to accurately describe the dynamics of the environment but it is sensitive to measurement noise. The noise suppression used is quite simple being only a low-pass filter. Since localization errors can typically grow quite large compared to voxel sizes, noise ratios in most real world use-cases will be too high to work properly if the same map resolution is used for both static and dynamic mapping. A solution for this could be to use a coarser map when modeling change in the map, since changes in the map is typically hard to predict with precision this map with higher fidelity might actually be more desirable.

To really generate some good result and make it useful for real world applications the mapping have to become less noise sensitive. Since it was hard to achieve good results with moderate noise levels, when only working on voxel level. It is believed that some higher level filtering or clustering have to be applied.

5.2 Exploration

From the experimental result of the exploration algorithm it can be seen that the algorithm is able to detect and reexplore dynamical parts of the map. It also manages to have an even rate of re-exploration over the whole environment during every experiment. However the exploration is noise sensitive when it comes to detecting and revisiting dynamical objects. Some measures to reduce the impact of noise are described in section 3.2.1 to ignore dynamic parameters in front of occupied space. However noise in occupied space is still present in the information gain since it is not segmented out. Even if the noise ratio in the map is small the much larger surface area of static objects such as walls floors and ceilings will make the impact bigger on the information gain. This can most clearly be seen in the exploration of the cross shaped room in figures 4.13 and 4.14. Both the mapping and exploration approach have good results in low noise setting but seem to be a

little bit too simple to work well with noise. Modifications need to be added in order for it to achieve better results that can work in a real-world scenario. The simplest of which could be just to use a coarser map for the dynamic modeling of the environment to reduce the ratio between noise and voxel size.

A challenge for the exploration problem is that the voxels containing interesting dynamical properties are very few compared to the total amount of voxels. This can be seen in figure 4.14 where the voxels inside dynamic objects will not be modeled as dynamic because of sensor occlusion. This then requires that the gain for these dynamical voxels have a high gain to make it worth the travel distance and then makes the algorithm sensitive to errors and noise when evaluating these aspects of the environment.

The results from the experiments shows the visitation rate for about 100 sessions but the algorithm changes its exploration pattern during sessions since it takes some time for the dynamic parameters in the map to converge. It might have been better to show the visitation pattern for only the last 50 sessions in the exploration experiments. If the robot operates in an environment where the dynamics might change over time a recency weighting of previous measurements could easily be implemented thus making the dynamic estimation more agile to changes.

The poses sampled for the exploration in this project were sampled using RRT* which works well for smaller environments but has a harder time to scale in bigger. Environments larger than those used in this report would have been challenging since large RRT* have to be expanded every time the robot reaches a sampled pose. A good sampling of poses that is even throughout the environment is needed to ensure good result from this algorithm. If certain areas are sampled less then those will probably be visited less frequently and it might take a long time before the agent has detected dynamical objects there.

An alternative to the sampling of poses could be to instead of just extending the RRT* in the direction of the pose the algorithm could plan all the way to that point. In the current setup it is costly to expand the RRT* in a larger environment since most tree nodes are close to the initial pose of the robot. The evaluation of poses and gain calculation along the path could still be the same but this approach might be better at sample poses in a larger environment. Although this is just an improvement it will probably not solve the problems with pose sampling. It might be better not to sample poses at all and instead use roadmap methods for path planning since most of the environment is already known. This roadmap could be kept between sessions.

A drawback with the planner is that it reevaluates at every node in its tree often. This makes the time horizon that it optimizes its path on really short, something that is necessary when dealing with unknown environments but less appealing in mostly known. It was not uncommon for the drone to move back and fourth between areas or planning a much less than optimal exploration loop during one session. The RRT* planner works well in unknown environment where new information is gained at most poses that the robot reaches. This is however not the case when re-exploration and a better planner could be implemented that are better at using

5.3. CONCLUSION AND FUTURE WORK

the information at hand.

To make the algorithm more agile with respect to changes in dynamicality and maybe allowing faster convergence of the dynamic parameters a moving average of the previous measurements could be implemented. Because as the algorithm stands now it could have problem in a long time scenario to swiftly adapt its dynamic model if the environment changes, in other words the algorithm will after some time tend to develop habits in its exploration patterns that might not be optimal if the dynamics of the environment is not time invariant.

Exploration in dynamic environments is a complex problem and the approach presented in this report is quite simple. Despite the simple approach it still resulted in a complex algorithm that was difficult to tune, hard to test in its developing stage and to evaluate when it was finished. The algorithm was difficult to evaluate because there were not any clear metric to describe how well it re-explore the environment and no other similar algorithms exists to compare it against. The lack of good evaluation metrics also made it really hard to test and improve the algorithm since it was difficult to predict and determine the quality of the outcome of a change.

Tuning of the parameters in the exploration algorithm was a time consuming and hard task that had the same difficulties as the testing of the algorithm during the developing stage. It often felt ambiguous and difficult to predict how the behaviour of the exploration would change when tuning parameters. A better set of parameters could certainly be found if there were more time.

The method presented in this report is a simple approach that has been proven to work in low noise settings. It has been developed as a first step in exploration of dynamic environments. As is common with simple approaches they might not have a high performance. However with some improvements it is believed that an exploration algorithm based on this algorithm will be able to work well in noisy environments such as the real world.

5.3 Conclusion and Future Work

5.3.1 Conclusion

Both the mapping and exploration algorithm works in low noise settings however it is determined that both are very noise sensitive. When the standard deviation of the noise is around the voxel size results are poor and often unusable. This applies especially to the exploration since spurious dynamic voxels around static objects have a big impact on its behaviour. In order for the exploration to work in a real world scenario fundamental improvements has to be made in how to filter out measurement noise in dynamic modeling.

The approach in this report is to take an exploration algorithm developed for static environments and modify it to operate in a dynamic environment. This has proven to work. However, it is clear that it is not optimal and results could be improved if the planning algorithm for example utilizes that it navigates in a known environment.

5.3.2 Future Work

Suppression of noise is now done with low pass filters on voxel level in the map. This approach is limited in what result it can achieve. Noise suppression on a higher level for example static objects or the whole map could give much better result and make it easier for the exploration to figure out which areas that are interesting to explore. Clustering or segmentation of dynamical voxels could also be something that both the mapping and exploration algorithm could make use of. This would make it easier to reduce the impact of noise.

One improvement on the noise sensitivity of the exploration could be to "see through" occupied space when calculating information gain at the sampled poses. The opacity of occupied space could depend on the probability of the voxel being free next observation, p_{of} . This will maybe make static objects with noise around them less attractive and also including space that have previous been occluded by the occupied voxels. This will perhaps encourage the exploration to seek out doors that are have a probability to be open upon next visit. Another improvement could be to threshold dynamic gain based on the occupancy values. However it is believed that fundamental changes in the representation of dynamic models has to be made in order to have an algorithm that works in a real world scenario.

As mentioned before a better planner that takes in more information about the environment and perhaps considers the gain of the whole exploration loop instead of just the next best view, a planner that optimizes on a broader time horizon and utilizes the knowledge already obtained in previous sessions.

Bibliography

- [1] Wurm, Kai M et al. “OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems”. In: *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*. Vol. 2. 2010.
- [2] Duberg, Daniel and Jensfelt, Patric. “UFOMap: An Efficient Probabilistic 3D Mapping Framework That Embraces the Unknown”. In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 6411–6418.
- [3] Saarinen, Jari, Andreasson, Henrik, and Lilienthal, Achim J. “Independent markov chain occupancy grid maps for representation of dynamic environment”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 3489–3495.
- [4] González-Banos, Héctor H and Latombe, Jean-Claude. “Navigation strategies for exploring indoor environments”. In: *The International Journal of Robotics Research* 21.10-11 (2002), pp. 829–848.
- [5] Bircher, Andreas et al. “Receding horizon" next-best-view" planner for 3d exploration”. In: *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2016, pp. 1462–1468.
- [6] Selin, Magnus et al. “Efficient Autonomous Exploration Planning of Large-Scale 3-D Environments”. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1699–1706.
- [7] Hähnel, Dirk et al. “Map building with mobile robots in dynamic environments”. In: *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*. Vol. 2. IEEE. 2003, pp. 1557–1563.
- [8] Wang, Zhan et al. “Modeling motion patterns of dynamic objects by IOHMM”. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 1832–1838.
- [9] Rapp, Matthias et al. “Semi-markov process based localization using radar in dynamic environments”. In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE. 2015, pp. 423–429.
- [10] Biber, Peter, Duckett, Tom, et al. “Dynamic maps for long-term operation of mobile service robots”. In: *Robotics: science and systems*. 2005, pp. 17–24.

BIBLIOGRAPHY

- [11] Arbuckle, Daniel, Howard, Andrew, and Mataric, Maja. “Temporal occupancy grids: a method for classifying the spatio-temporal properties of the environment”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 1. IEEE. 2002, pp. 409–414.
- [12] Stachniss, Cyrill and Burgard, Wolfram. “Mobile Robot Mapping and Localization in Non-Static Environments”. In: (2005).
- [13] Krajník, Tomas et al. “Spectral analysis for long-term robotic mapping”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 3706–3711.
- [14] Santos, Joao Machado et al. “Lifelong information-driven exploration to complete and refine 4-D spatio-temporal maps”. In: *IEEE Robotics and Automation Letters* 1.2 (2016), pp. 684–691.
- [15] Rosen, David M, Mason, Julian, and Leonard, John J. “Towards lifelong feature-based mapping in semi-static environments”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 1063–1070.
- [16] Ambruş, Rareş et al. “Meta-rooms: Building and maintaining long term spatial models in a dynamic world”. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 1854–1861.
- [17] Ambruş, Rareş et al. “Unsupervised learning of spatial-temporal models of objects in a long-term autonomy scenario”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 5678–5685.
- [18] Kunze, Lars et al. “Artificial intelligence for long-term robot autonomy: a survey”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 4023–4030.
- [19] Biswas, Joydeep and Veloso, Manuela M. “Localization and navigation of the cobots over long-term deployments”. In: *The International Journal of Robotics Research* 32.14 (2013), pp. 1679–1694.
- [20] Hawes, Nick et al. “The strands project: Long-term autonomy in everyday environments”. In: *IEEE Robotics & Automation Magazine* 24.3 (2017), pp. 146–156.
- [21] Hanheide, Marc, Hebesberger, Denise, and Krajník, Tomáš. “The when, where, and how: an adaptive robotic info-terminal for care home residents”. In: *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*. ACM. 2017, pp. 341–349.

TRITA -EECS-EX-2020:877