Kevin Meyer

Multi-robot informative path planning under communication constraints

Master's thesis in Cybernetics and Robotics
Supervisor: Anastasios Lekkas
February 2021

Kevin Meyer

Multi-robot informative path planning under communication constraints

Master's thesis in Cybernetics and Robotics
Supervisor: Anastasios Lekkas
February 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Acknowledgements

This thesis is the culmination of several months of work, and many years of study. There are countless individuals who have helped me come this far. I would like to thank my supervisor, Associate Professor Anastasios Lekkas, for the guidance he has given me throughout the last year. My gratitude is extended to those who have helped me formulate my thoughts and proof-read sections of my thesis, in particular Marvin Reza, Katharina Zürbes and Kiet Hoang. I want to thank my friends, family and those close to me for their motivating words and general positivity. Finally, once again, shoutout to the bois.

Kevin Meyer
February 2021

Abstract

Collaborative multi-robot systems show great promise in extending the capabilities of a robotic system beyond the abilities of any single robot. This is particularly evident in the case of heterogeneous robot teams, where the strengths of individual robotic platforms are combined to mitigate their corresponding weaknesses. This thesis presents a system that coordinates a team of sensor-equipped quadrotor Unmanned Aerial Vehicles (UAVs) acting as mobile range sensors for an Unmanned Ground Vehicle (UGV). The robot team operates in an unmapped, static environment populated by obstacles. Communication constraints between the UGV and the individual UAVs are imposed to ensure an adequate throughput of sensor data from the UAVs. Several path planning algorithms are presented which enable collaborative and communication constrained planning. A collaborative exploration planner is presented which combines the results from individual view-based exploration planners. Two variants of formation-based planners are also presented; one which uses a set of predefined formation poses and another which continuously reshapes the formation through sampling. Elements of the aforementioned planners are also merged to create two types of combined planners. The systems are implemented within the Robot Operating System (ROS) framework and tested in the Gazebo simulation environment. Results from testing the planning systems in several scenarios show that each planning algorithm has unique strengths and weaknesses. While the collaborative exploration planner excels in total information gain, the formation-based planners are observed to be more robust and efficient in gathering information.

Sammendrag

Samarbeidende multirobotsystemer utnyttes i større grad fordi de kan utvide mulighetene til et robotsystem utover evnene som hver enkelt robot innehar. Dette er spesielt tydelig når det gjelder heterogene robotlag, der stryken til individuelle robotplattformer kan kombineres for å dempe svakhetene hos de enkelte robotene. Denne oppgaven presenterer et system som koordinerer et lag med sensorutstyrte «quadrotor» ubemannede luftfartøy (UAVs) som skal fungere som mobile sensorer for et ubemannet bakkekjøretøy (UGV). Robotlaget opererer i et ukjent, stillestående miljø som er fylt med hindringer. Kommunikasjons-relaterte begrensninger mellom UGV og de enkelte UAV-ene pålegges systemet for å sikre tilstrekkelig gjennomstrømning av sensordata fra UAV-ene. Flere algoritmer presenteres for å muliggjøre samarbeidende og kommunikasjonsbegrenset planlegging. En algoritme presenteres som kombinerer resultatene fra individuelle utsiktsbaserte planleggingsalgoritmer. To typer formasjonsbaserte algoritmer presenteres: en som baserer seg på et forhåndsdefinert sett med formasjoner og en om kontinuerlig endrer formasjonen gjennom prøvetaking. Elementer av disse planleggingsalgoritmene blir også slått sammen for å skape to typer kombinerte planleggere. Systemene er implementert innenfor ROS-rammeverket (Robot Operating System) og testet i Gazebo-simuleringsmiljøet. Resultater fra testing av planleggingssystemene i simulerte scenarier viser at hver planleggingsalgoritme har unike styrker og svakheter. Mens den algoritmen som kombinerer individuelle planleggingsalgoritmer utmerker seg i total informasjonsgevinst, observeres det at de formasjonsbaserte algoritmene er mer robuste og effektive i å samle informasjon.

Preface

This project was conducted during the fall semester of 2020 at the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway. The project and thesis work are the author's contribution to the course "TTK4900 - Engineering Cybernetics, Master's Thesis". The author's chosen specialization is within "Roboter og fartøystyring - Autonome systemer".

The implemented systems were developed and tested on a computer equipped with an AMD Ryzen™ 5 3600 CPU clocked at 3.6/4.2GHz, an Nvidia GeForce®GTX 1060 GPU clocked at 1506/1708 MHz, 32 GB of RAM clocked at 3200 MHz and running Ubuntu 18.04 (Bionic).

Several open source libraries and frameworks were used, which are presented in detail in Chapter 3. A brief summary of these follows:

- **C++** was the main programming language used for the implemented systems.
- **Robot operating system (ROS)**[1] Melodic Morenia provided the framework for the robotics systems.
- **Gazebo**[2] v9.0.0 was the simulation environment used for the implemented systems.
- **OctoMap**[3] v1.9.1 was used to create and interact with a 3D voxel representation of the robots' environment.
- **PX4 autopilot software**[4] v1.10 provides the software architecture for the aerial vehicle simulator. This includes the `mavlink`[5] messaging protocol.
- Several **ROS-packages** were used, outside of the packages available in the ROS core components[6] and in the desktop installation of ROS[7]:

[1]https://www.ros.org/
[2]http://gazebosim.org/
[3]https://octomap.github.io/
[4]https://docs.px4.io/v1.10/en/index.html
[5]https://mavlink.io/en/
[6]https://www.ros.org/core-components/
[7]http://wiki.ros.org/melodic/Installation/Ubuntu#Installation-1

- The common packages for the Husky UGV system[8].
- The common packages for the Navigation stack[9].
- The `teb_local_planner`[10] package plugin.
- The `mavros`[11] package.
- The common packages for interfacing ROS with OctoMap[12], specifically the `octomap_ros`[13], `octomap_msgs`[14] and `octomap_server`[15] packages.

An overview of the contributions in this thesis are presented in Section 1.3, with the design and implementation of the systems presented in Chapter 4 and an analysis of the results presented in Chapter 5. The contributions stemming from previous work performed by the author in the Specialization project [1] are presented in Chapter 3.

All graphics and images featured in this thesis were made by the author, except when explicitly stated otherwise.

The code used in this project is available on the author's personal github in the following repository: `https://github.com/kevinmey/mipp_project`.

Additionally, a video presenting some of the main results from this project is available on the following web address: `https://youtu.be/XLopOdeN_Ks`. It can also be found by scanning the QR code:

Fig. 1 Scannable QR code for the project video.

[8]http://wiki.ros.org/Robots/Husky
[9]http://wiki.ros.org/navigation
[10]http://wiki.ros.org/teb_local_planner
[11]http://wiki.ros.org/mavros
[12]http://wiki.ros.org/octomap
[13]http://wiki.ros.org/octomap_ros
[14]http://wiki.ros.org/octomap_msgs
[15]http://wiki.ros.org/octomap_server

Table of contents

List of figures

Glossary

agent	An entity which can perceive its environment through sensors and act upon it through actuators.
anytime algorithm	Algorithms with the two properties of quickly finding a feasible solution and improving the solution with further processing.
configuration	A complete specification of the location of every point on the robot geometry.
configuration space	The space of all possible robot configurations.
percept	An agent's perceptual inputs at any given instant.
robot	The physical manifestation of an agent, comprising of both software and hardware.
task environment	Specification of the performance measure, environment and agent actuators and sensors.
voxel	A three-dimensional cell in a regular grid encoded with a value.
workspace	The physical space the robots operate in (usually $\mathbb{R}^2$ or $\mathbb{R}^3$).

Acronyms

BS	Base Station.
CCB	Communication Constraint Breakage.
CCMIPP	Communication Constrained Multi-robot Informative Path Planning.
CVAC	Collaborative Visual Area Coverage.
EKF	Extended Kalman Filter.
FCU	Flight Controller Unit.
FOV	Field of View.
FSM	Finite-State Machine.
GNSS	Global Navigation Satellite System.
GNT	Gap Navigation Tree.
GUI	Graphical User Interface.
IGE	Information Gain Efficiency.
IMU	Inertial Measurement Unit.
IPP	Informative Path Planning.
LiDAR	Light Detection And Ranging.
LoS	Line-of-Sight.
MIPP	Multi-robot Informative Path Planning.
MPC	Model Predictive Control.
OA	Obstacle Avoidance.
PID	Proportional–Integral–Derivative.
PRM	Probabilistic Roadmap Method.
ROS	Robot Operating System.
RRT	Rapidly-exploring Random Tree.
RRT*	Rapidly-exploring Random Tree Star.
SFR	Sampling-based Formation Reshaping.
SLAM	Simultaneous Localization And Mapping.

SPA	Sense-Plan-Act.
SRG	Sensor-based Random Graph.
SRT	Sensor-based Random Tree.
TEB	Timed-Elastic-Band.
TIG	Total Information Gain.
ToF	Time of Flight.
UAV	Unmanned Aerial Vehicle.
UGV	Unmanned Ground Vehicle.

Symbols

$\alpha(i)$	Sampled configuration.
$\mathcal{C}$	Configuration space.
c_{ang}	Angular distance cost parameter.
c_{euc}	Euclidean distance cost parameter.
c_{info}	Information gain cost parameter.
l_{com}	Communication range.
t_{com}	Communication timeout.
f_{uav_sfr}	Formation reshaper loop frequency.
n_{form_bank}	Amount of formations in formation bank.
n_{sfr_bank}	Amount of sampled formations in SFR bank.
n_{sfr_keep}	Amount of sampled formations kept in SFR bank.
ψ_{sfr_samp}	Max. sample yaw deviation of SFR planner.
r_{sfr_samp}	Sampling radius of SFR planner.
t_{sfr_samp}	SFR planner sampling timeout.
f_{uav_pl}	UAV planner loop frequency.
f_{ugv_pl}	UGV planner loop frequency.
l_{hybrid}	Hybrid planner switch distance.
$f_{octomap}$	OctoMap update frequency.
$res_{octomap}$	OctoMap voxel resolution.
q	Configuration.
c_{mult}	Max. multiplier value for waypoint multiplier.
d_{mult_max}	Max. distance threshold for waypoint multiplier.
d_{mult_min}	Min. distance threshold for waypoint multiplier.
h_{uav_int}	UAV operation altitude interval.
h_{uav_min}	Min. UAV operation altitude.
$h_{uav,i}$	UAV *i* operation altitude.
ψ_u	Standard angular unit.

r_{uav_cov}	UAV circular coverage radius.
l_{uav_cov}	UAV circular coverage distance.
l_{uav_rrt}	Max. edge distance for UAV RRT* planning.
$p_{rrt\_samp,i}$	Sampling center for UAV *i* RRT* planning.
r_{rrt_samp}	Sampling radius for UAV RRT* planning.
rnk_{rrt_v}	Max. vertex rank for UAV RRT* planning.
t_{rrt_samp}	Sampling time for UAV RRT* planning.
v_{uav}	Max. UAV velocity.
l_{ugv_wp}	UGV local waypoint spacing.
n_{ugv_wp}	UGV local waypoint amount.
t_{ugv_wp}	Min. UGV local waypoint time.
v_{ugv}	Max. UGV velocity.
$\mathcal{W}$	Workspace.

Chapter 1

Introduction

Artificial intelligence and robot systems are gaining a greater presence in the lives of people and society. While artificial intelligence now has a permanent presence in the worlds of finance, data analysis and media, robot systems have only recently begun transitioning from research labs and factories to ordinary life. The mental image of robots as precise but relatively "dumb" robotic manipulators which populate manufacturing lines is giving way to complex systems which interact with us more directly. This includes vacuum cleaning robots in peoples homes, self driving cars on the roads and unmanned drones patrolling the skies.

Robot systems show great promise in situations where human operation is difficult, dangerous or otherwise undesirable. Ground based robots, also known as Unmanned Ground Vehicles (UGVs), have been applied to the fields of emergency response in disaster scenes [2], structural health inspection [3] and for extraterrestrial exploration [4]. The aerial counterpart to these robots, Unmanned Aerial Vehicles (UAVs), have been used extensively for environmental monitoring [5, 6] and industrial inspection [7, 8]. Systems have also been implemented which leverage both UGVs and UAVs in cooperative teams. Such systems have been applied to the issues of precision agriculture [9], humanitarian demining operations [10], wildfire detection and fighting [11] and mapping disaster areas [12].

The goal of this project is to investigate and implement solutions for coordinating sensor-equipped UAVs around a UGV in operation. The UAVs will be tasked with investigating and mapping areas around the UGV while maintaining certain communication constraints to the UGV. Several methods will be implemented, presented and have their performance compared in multiple simulated scenarios.

In the rest of this chapter, the motivation for implementing such a system will first be presented. This motivation is accompanied by a review of related literature to present the current state of the field. Following this, a definite problem formulation is given to illustrate the objectives of this project work. Finally, an overview of the contributions made in this project is presented, along with an outline of the project thesis.

1.1 Motivation and related work

This project is a continuation of the Specialization Project [1] completed prior to the commencement of the master thesis work, in which a simulation framework for Multi-robot Informative Path Planning (MIPP) was implemented. The work centered on a robotic team of one UGV and multiple UAVs which were equipped with sensors and which could plan and move in an unknown environment populated by obstacles. The systems showcased in the specialization project centered on exploration of the unknown environment by the individual robots.

In this thesis, the focus has been shifted to the coordination of the UAVs around the UGV such that they aid the UGV continuously as mobile sensors. This is more in line with the scenario that was originally presented and discussed in [1]. A central theme of this scenario is the UGV-platforms superior payload and computing capabilities compared to the UAV-platforms superior mobility. Offloading the data processing aspect of the UAV to the UGV would place requirements on the communication between individual vehicles. Therefore it was desirable to investigate solutions which could take into account communication constraints when coordinating the UAVs as mobile sensors around the UGV.

This problem can be categorized as a Communication Constrained Multi-robot Informative Path Planning (CCMIPP) problem applied to a UGV-UAV system. With this initial, broad problem formulation in mind, a literature review was conducted to investigate similar problems and potential solutions within the space of UGV-UAV cooperative systems. The review of these systems is presented here, while a more general review of related fields is presented in Chapter 2.

UGV systems have been extensively researched in robotics, as ground based systems are able to carry heavier computing and sensor payloads. While sufficient computing and sensing power to enable autonomous operation is now available in lighter payloads, the ability for UGV systems to carry heavier payloads still translates to the ability to analyze more data. Systems which enable autonomous operation of UGVs also have the added benefit of abstracting planning from the three-dimensional world to two-dimensional representations.

UAV systems are often characterized by their ability to operate in all three spatial dimensions, and to do so at higher speeds and maneuverability than their ground-based counterparts. Different classes of UAV systems exist, which influences the type of problems the UAV system can be applied to. Fixed-wing UAVs mainly operate outdoors and over long distances. Research into multirotor UAVs has advanced in recent years as they are able to operate both indoors and outdoors and can easily switch between maintaining a stable hover and performing acrobatic maneuvers. This thesis will mainly be concerned with the latter of the two, and the term UAV will generally refer to multirotor systems unless stated otherwise.

UGV-UAV cooperative systems which exploit the strengths and weaknesses of each vehicle platform have been applied to many challenges. In [13], application areas and current challenges for such systems are presented. Included is the use of UAVs as mobile sensors to provide relative tracking for state estimation augmentation [14, 15], aid in UGV formation keeping [16, 17], provide terrain classification data for UGV navigation [18, 19] and to detect targets [20]. Most relevant for this thesis is the use of UAVs as range sensors to sense the immediate vicinity of the UGV, and literature from this field has therefore been given more attention.

Early examples of UGV-UAV cooperation for mobile range sensing usually revolve around the coordination of fixed-wing UAVs around groups of UGVs. In [21], a system for coordinating fixed-wing UAV orbits around a UGV group and its estimated centroid is demonstrated for a surveillance scenario. Though the advantages of having aerial sensors above the UGV formation are implied in the scenario, they are not explicitly considered. Such considerations are taken in the systems presented in [22] for a similar scenario. Here, an area is investigated to localize targets using decentralized control of the vehicles. The vehicles are driven by information gain gradients on a shared probabilistic map, with the UAVs providing an initial map that the UGVs later refine with better sensors.

Usage of multirotor UAVs as mobile sensors to aid a UGV system have also been demonstrated. In [23], a UGV equipped with a helipad and UAV is used in a periodic indoor surveillance scenario. The UGV drives along a predefined path, stopping at inspection points to allow the UAV to take off and survey an area before landing again. Similarly, [24] demonstrates a group of UAVs following a group of UGVs in a three dimensional exploration scenario. Exploration tasks in the form of frontiers are assigned using Integer Programming methods, with a distinction made between frontiers which are explorable by UGVs and UAVs. The UAVs passively follow the UGVs until an area is detected which the UGVs cannot map (such as the top of an obstacle), following which the method will assign a UAV to map the area. A visibility constraint for the UAV frontiers is used to ensure the UAVs are always visible to at least one UGV, while the general communication constraint problem is tackled by using a distributed system approach.

Systems have also been demonstrated where UAV surveillance precedes UGV operation. In [25], a UAV is used to map a mock-up disaster area populated with fixed and movable obstacles before a UGV can plan a path to a spotted victim. To provide the initial map, a UAV equipped with a downward-facing camera flies in a *Boustrophedon* path [26], also known as a lawn mower pattern, to cover the entire predefined area. Alternatively, [27] demonstrates a system which more intelligently chooses areas for the UAV to explore prior to UGV operation in a search and rescue mission. The UAV exploration algorithm, which is a modification of the D* algorithm, performs a search over candidate paths to minimize the *total response time* of both vehicles. Candidate paths are investigated considering both explored and unexplored space (and points on the boundary between them), the total time it would take for the UAV to explore these paths and the time saved should a better path for the UGV be found in the unexplored area.

Systems exist which coordinate the UAVs to aid the UGV system in a more continuous manner. The method presented in [28] utilizes the advantages of each platform in a continuous exploration scenario. Similar to the method used in [24], two different frontier regions are detected which are assigned to either the UGV or the UAV. The vehicles are however allowed to operate independently of each other, meaning their primary collaborative feature comes in the form of building a combined map and exploring different areas. A similar strategy is proposed in [29], where a UAV either provides fine-grained data to complement a coarse-grained map built by a UGV, or augments the UGV sensor data in real time. While the former resembles the two-phased strategies presented earlier, the latter can be extended to the continuous collaborative case.

Research has also been done on systems where the UAV continuously aids a UGV with sensor data while being constrained by the UGV location or planned path. A simple example of this is presented in [30] where a UAV hovers over the UGV in operation, providing an aerial image of the UGV and it's surroundings while also performing relative tracking for localization. The images can either be used by a teleoperator driving the UGV, or by image processing algorithms to help plan trajectories in rough terrain. A similar approach for indoor search and rescue is demonstrated in [31]. Here, a UAV provides a birds-eye view for the UGV and localizes itself by using a visual tag on the UGV that is visible at all times. Likewise, the UAV is localized using a visual tag visible from the UGV camera, and relative localization is achieved.

In [32], a UAV assists a UGV by performing dynamic area coverage of the area around a moving UGV. The UAV motion control is constrained by a requirement to maintain visibility to the UGV, expressed as a range constraint. Two methods are analyzed, one utilizing a naive orbital trajectory for the UAV, and another optimization-based coverage technique. Both techniques are analyzed and evaluated using an information gain represented by the area that is covered by the UAV and an *effective coverage* radius around it.

A system for the surveillance of the path ahead of the UGV has also been presented in [33]. Here, a UAV is tasked with exploring the area around the immediate future path of the UGV before the UGV is able to come within a certain distance of unmapped area. This is solved by having the UAV utilize a *Conformal Lawn Mower* coverage plan of the area ahead of the UGV, which is a *Boustrophedon* path that conforms to the contours of the UGV path.

Physical constraints in UGV-UAV systems have also been investigated, with the primary example being tethered systems. Modelling and design of such systems is presented in [34], without considerations for the planning implications of the tether. Usage of the tether as a tool to scale difficult terrain along with having the UAV as a mobile sensor for the UGV was presented in [35], though tether constrained planning is likewise not considered.

In [36] the complementary sensors of the UGV and UAV in a tethered system are considered in a navigation scenario. Collaborative control is achieved by having the UGV first construct a plan using a Rapidly-exploring Random Tree Star (RRT*) planner and having the UAV scout ahead along future waypoints of the UGV path. A fixed maximum tether length to constrain the scout ahead distance and a fixed UAV altitude simplify the problem. A tethered UAV acting as a visual assistant for UGV is presented in [37], where the goal is to place the UAV to maximize the quality of its viewpoint for a teleoperator while minimizing risk. The risk is calculated taking into account the path leading to the viewpoint and its effects on the tether length, tortuosity, azimuth and contact points.

Having reviewed the literature, the CCMIPP problem presented earlier can be compared to existing solutions. Early solutions using fixed-wing UAVs have been applied to similar problems [21, 22], but are not directly applicable to the multirotor case. Systems exist where the usage of each vehicle type is split into phases, such as UAV deployment from a UGV-mounted helipad [23] or UAV mapping prior to UGV operation [25, 27]. For situations where more continuous collaboration is desired, systems either focus on independent operation of both vehicles without constraints between them [28, 29], simplify the problem by having the UAV maintain a stable hover over the UGV [30, 31] or apply constraints in simplified scenarios with point-mass vehicle models and obstacle free environments [32, 33].

Two promising directions for further investigation in UGV-UAV collaborative systems have been identified by the author. The system presented in [24] applies a UAV swarm to aid UGVs in sensing areas that are unreachable to the UGVs while constraining the UAV motion through a visibility constraint. Allowing the UAVs to investigate areas independently under constraints to the UGVs would be an interesting direction for investigation, but no such works have been identified by the author. Tethered UGV-UAV systems are also interesting to analyze, as the constraints of the physical tether can in some cases emulate communication constraints. The work in [37] in particular is interesting, as the risk calculation related to the tether includes elements that are relevant to communication constraints (which will be presented in Section 2.3.4) and comparing this risk to the utility of certain viewpoints can potentially be adapted to the CCMIPP scenario.

Outside of the examples presented here, the author has investigated concepts from other related fields. Relevant literature for communication constrained exploration was a main focus for the review done in the specialization project, and is presented in Section 2.3.2 and Section 2.3.4. The former section also includes a review of methods within the related field of Collaborative Visual Area Coverage (CVAC). For the case of multiple UAVs acting as mobile sensors over the UGV, theory and solutions found in formation control were also investigated and are presented in Section 2.3.3.

From the review of the literature presented here, as well as the related literature which is presented in Chapter 2, the author has been unable to identify existing solutions which are applicable to the general CCMIPP problem presented earlier in this section. The exact contributions in regard to filling that gap are presented in Section 1.3. Following the literature review, the problem to be solved can be presented in greater detail.

1.2 Problem formulation and objectives

The primary goal of this project is to implement a multi-robot planning system which can coordinate a set of sensor-equipped UAVs around a UGV. The individual UAVs are to maintain communication constraints with the UGV while they operate in an unknown three-dimensional environment occupied by obstacles. The effectiveness of the developed system(s) will be determined by information-based metrics, with the basic goal of maximizing information gained about the unknown environment over time. A more detailed description of the performance metrics can be found in Section 5.1. The developed system(s) will be tested in multiple realistic simulation scenarios that resemble the intended use-case of outdoor operation in relatively large environments.

The problem is in large part formulated as a continuation of the specialization project work presented in [1], with the primary motivation stemming from a proposed cooperation that was discussed in the formulation of the aforementioned project. Though a formal cooperation never came to fruition, the problem to be solved is still rooted in a problem that was identified by entities in the industry. As such, the author still sees relevance in investigating the problem.

With both the problem, the initial findings from the specialization project and the review of related works in mind, two directions for investigation are identified which can potentially solve the problem. These two directions can be formulated as two research questions:

- Can the Informative Path Planning (IPP) systems made for the UAV vehicle platforms be extended to solve the UGV-UAV CCMIPP problem in an effective manner?
- How would the aforementioned multi-robot system compare with systems developed explicitly for the purpose of CCMIPP in the same UGV-UAV scenario?

The first task highlights the continuation of the work done in [1], where the individual UAV systems were observed to be effective when applied to the unconstrained (in terms of communication constraints with the UGV) single-robot case. Though the single-robot systems were not developed explicitly with the cooperative and communication constrained multi-robot use-case in mind, it is interesting to analyze its effectiveness in such a scenario and what emergent behaviours the system displays as a result.

The second task is motivated by the intention of solving the underlying goal in the most effective manner, rather than being wed to the idea of extending the already implemented systems and making them as efficient as possible. As presented in Section 1.1, the author has been unable to identify existing and accessible systems which are applicable to this exact problem. Therefore, one or multiple systems must be created as a part of this thesis work which can be compared to the extension of the work done in [1].

As the focus has been shifted to coordinating the UAVs around a UGV in operation, a relevant performance metric must be defined. As the exact UGV operation is not a part of the problem formulation, an exploration-focused performance metric such as "time to explore an environment" can not be used outside of the case where the UGV operation involves exploration of the environment. Though this case is presented in Section 5.6, cases of pure UGV path planning operations are also presented.

Since the systems will be building on and tested in the framework developed in [1], the systems will have to interface with multiple robotics subsystems implemented using the Robot Operating System (ROS) framework. This includes sensor-, localization and control-systems, which in turn interface with the realistic simulation environment Gazebo. Therefore, the primary goal is complemented by the goal of making the system run in real time in the existing robotic system, and to make modifications on the existing system to accommodate the new system.

The author acknowledges that testing the developed systems in a realistic robot architecture requires the use of a multitude of other robot systems. These systems include perception systems to localize the vehicles and process the sensor data into maps, as well as low-level control systems to guide vehicles towards high level navigation goals. As implementing these systems would constitute large works on their own, the focus is placed on implementing the high-level planning systems which give navigational goals to the UAVs. The systems which make up the robot architecture and framework are presented in Chapter 3, while the contributions of the author will be presented in the next section.

1.3 Contributions

This work expands on the work done during the specialization project, where a simulation and robotics framework for testing CCMIPP systems was developed. This included the development of single-robot exploration systems for both the UGV and the UAVs. These contributions are elaborated in the specialization project thesis [1]. This work has also relied on the work done by countless other parties in developing robotics frameworks and systems, and the author has made an effort to give credit to these parties where possible.

The main contributions of this work will be presented in Chapter 4 and Chapter 5, while some minor contributions that expand more directly on the work done in the specialization project are presented in Chapter 3. While the contributions include specific methods and systems which implement these methods, the creation of the system as a whole is also a contribution of this work.

Several systems have been developed which, to the authors best knowledge, are novel in some way when compared to the surveyed literature. These novel contributions include:

- Systems that enables the coordination of a team of UAVs around a UGV operating in a unknown obstacle-filled environment in a high fidelity simulator, including recovery behaviour for UAVs that break range or Line-of-Sight (LoS) communication constraints to the UGV.
- A method for coordinating UAVs in an asynchronous, real-time manner using individual sampling-based exploration algorithms such that individual UAVs maintain range and LoS communication constraints to a UGV operating in the aforementioned environment.
- A method for sampling UAV-formations above a UGV operating in the aforementioned environment such that individual UAVs maintain range and LoS communication constraints to the UGV while also reshaping the formation to increase information gain.

A simplified overview of the system is presented in Figure 1.1. The contributions of this work are largely centered on the systems presented in blue. Systems presented in red were primarily implemented by the author in the Specialization project, but were heavily modified for use in this project. The other systems for both the UGV and UAV platforms are presented in greater detail in Chapter 3, along with the general robotics framework they are implemented in.

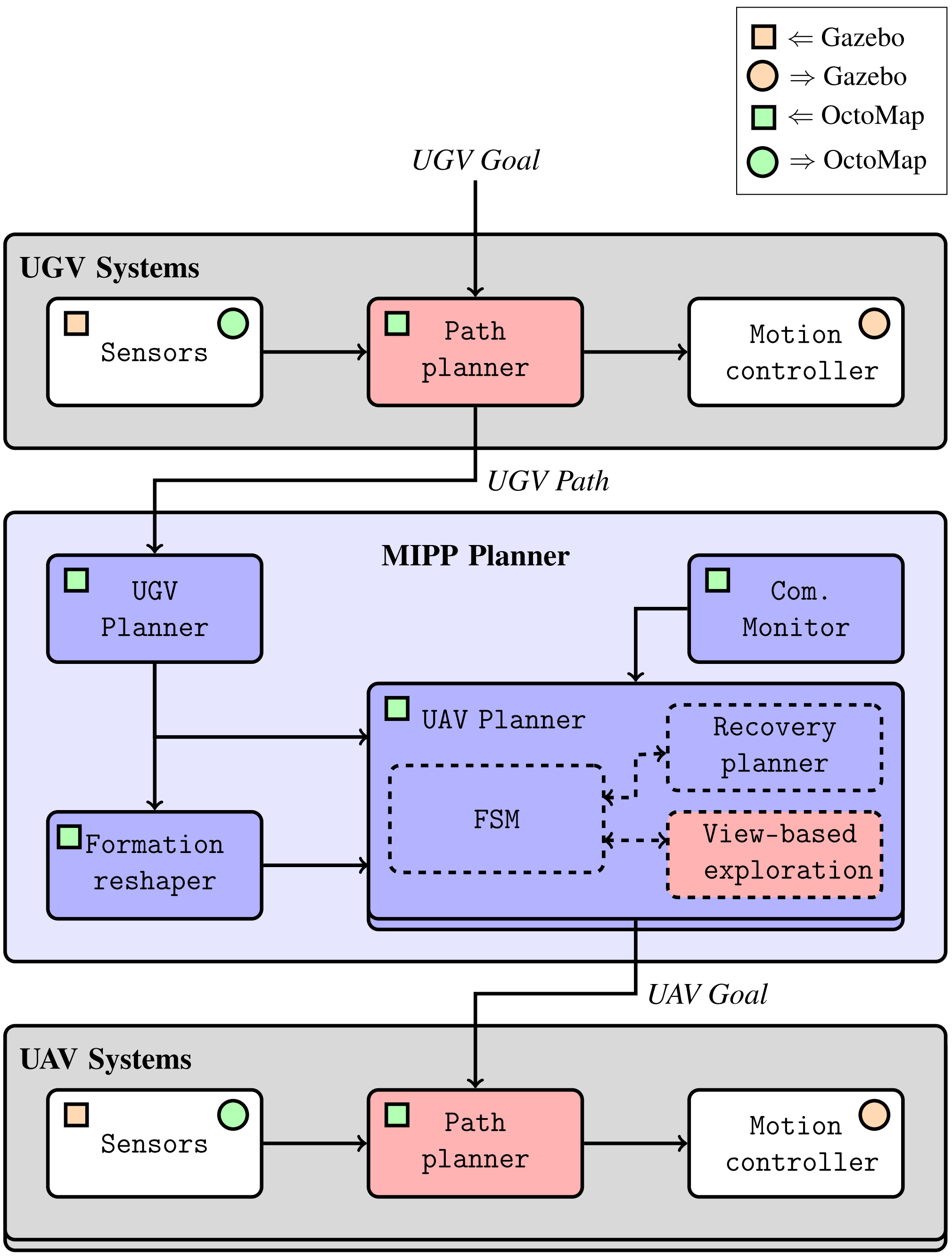


Fig. 1.1 Simplified overview of the systems making up the main system architecture of the CCMIPP system. Blue systems were primarily implemented for this project work . Red systems were made by the author in the Specialization project, but were heavily modified for use in this project. Systems which may have multiple instances running, such as the UAV Systems, are indicated as a stack of system blocks. Inputs from and outputs to the mapping (OctoMap) and simulation (Gazebo) systems are also visualized.

1.4 Report structure

This project thesis is divided into six main chapters:

Chapter 1 introduced the general problem formulation that is to be tackled in this thesis. A review of existing systems and solutions for the general problem formulation was presented, and a gap in the research was identified that this project work could address. A more defined problem formulation was then presented, along with the objectives of this project work. The contributions made by this project work were also presented, along with an overview of the implemented system structure.

Chapter 2 presents background material for the systems that are implemented in this project. This includes both theory and related literature work. The chapter begins by illustrating some general concepts within robotic sensing, planning and motion control. Following this, several concepts within robot path planning will be introduced, including the central path planning algorithms used in this project. These concepts are then extended to the multi-robot case, with some additional theory on formation control and communication constrained planning.

Chapter 3 covers the software frameworks and systems used in this project. First, both the robotic software framework and the simulation software are covered. Following this, the sensor- and control-systems for the simulated vehicle platforms are presented. The systems developed during the Specialization project are also covered in this chapter, including the implemented single-robot exploration planners.

Chapter 4 presents the design and implementation of the main systems created for this project. The overall system architecture which enables collaborative robot planning is presented first. The chapter then presents the invidiual planning methods and algorithms implemented for this project work.

Chapter 5 presents the results from testing the implemented systems in multiple simulated scenarios. First, the performance metrics used to assess the systems' performance are defined, followed by a presentation of the simulated environments that the systems will be tested in. Four different scenarios are then presented, along with the results of running the implemented systems in these environments. These results are briefly commented on.

Chapter 6 concludes this thesis. The chapter includes a summary of the material presented in the preceding chapters, as well as a more general discussion of the thesis results. Potential improvements and future work are presented at the end of the chapter to illustrate what remains to be done and what can be achieved with the foundation created in this project work.

Chapter 2

Background

This chapter presents background information and theory that was considered and utilized when solving the problem presented in Section 1.2, as well as some literature from the relevant fields. Since the systems which had to be implemented cover a broad spectrum of robotics fields, the chapter will have to cover a range of topics in a concise manner. Concepts from this chapter will be referenced in Chapter 4 when motivating the choices that were made when designing the system.

This chapter contains a substantial amount of information that was gathered during the Specialization project [1], which has been condensed in this work. In general, the author has chosen to summarize information on single-robot concepts and expand on multi-robot concepts, as these concepts are more relevant to the work done in this project. A detailed list of the information taken from the Specialization project is presented below.

- Section 2.1.1 summarizes Section 2.2 of [1].
- Section 2.1.3 summarizes Section 2.3 of [1].
- Section 2.2.1 summarizes Section 2.1 of [1].
- Section 2.2.2 summarizes Section 2.4 of [1].
- Section 2.2.3 summarizes Section 2.5 of [1].
- Section 2.2.4 expands on Sections 2.7.1-3 of [1].
- Section 2.3.1 summarizes on Section 2.6 of [1].
- Section 2.3.2 expands on Section 2.7.4 of [1].

- Section 2.3.4 expands on Section 2.8 of [1].

This chapter begins by presenting information that is necessary to understand the framework of modern robotics systems, and which influences the choices made when designing planning systems. Path planning concepts for single-robot systems are then presented, both for the pure path planning case and for information based variants. These concepts are then expanded to the multi-robot case, with an emphasis on multi-robot planning for information gathering. Other concepts related to multi-robot planning, like formation control and communication constraints, are briefly discussed.

2.1 Modern robotics

Modern robotics systems often use the Sense-Plan-Act (SPA) methodology when operating autonomously. Robotic sensing is the act of taking measurements and creating a representation of the environment which can be used for higher level robot planning. Included in this is the act of localizing the robot itself in this environment. With the environmental measurements, also known as percepts, a robot can make plans given the current state of the world and with knowledge about how its actions affect the achievement of a goal. Given an environment and a plan, robots use lower level motion controllers to actuate the system and follow the given plan.

As a contribution of this thesis includes the assembly of a robotics system which encompasses all of these areas, some fundamental theory will be presented in this section. The systems that make use of the theory covered in this section are largely not the contribution of the author, and enable the assumptions made about the problem to be solved in Section 1.2. Theory and background relevant to the implementation of the planning systems making up the main contributions of this thesis are addressed in later sections of this chapter.

2.1.1 Sensing and representing the environment

Before any plans or actions can be made and enacted by a robot, it must have a representation of the environment and its place in it. Range-based sensors mounted on the robots can provide information about the robot's surroundings. In the case of Light Detection And Ranging (LiDAR)-sensors, this sensing is done by measuring the Time of Flight (ToF) of emitted and reflected laser pulses to get a range to the reflective surface [38, §31.2.2]. Cameras are also a form of range sensor which provide rich visual information to autonomous system processes.

Cameras in particular setups can also provide range information, either through triangulation in the case of stereo cameras [38, §31.2.4] or through detecting a structured light pattern that is emitted onto objects in front of the camera [38, §31.2.5].

Localizing the robot can also be done using robot-mounted sensors. Robotic systems applied to outdoor use-cases often make use of Global Navigation Satellite System (GNSS) receivers to provide a global position. Measurements from robot-mounted proprioceptive sensors such as Inertial Measurement Units (IMUs) or wheel odometers provide data about the robots immediate acceleration and velocity. Fusing position, velocity and acceleration measurements can be done by using a Kalman Filter [39], together with a model describing the robot. Data from the the aforementioned range-based sensors can also be used to localize the robot by comparing measurements over time, such as through visual odometry and Simultaneous Localization And Mapping (SLAM) strategies [40].

A map of the environment can be built using a collection of range points and localization data relating these range points to three-dimensional coordinates in the environment. Such point-cloud representations of the environment preserve the original distribution of the data and place no restrictions on the world geometry. but are in general difficult to use for higher level planning [38, §31.4.1]. Common abstractions include discretization of the data into cells of a specific spatial resolution representing areas of the environment. Occupancy grid maps are a common way of abstracting 3D point clouds into a 2D grid cells, each with a probability of being free or occupied or being unmapped [41, 42, §45.2.1]. Other methods which preserve more of the 3D data include occupancy grid maps which assign an average elevation to each cell (also referred to as 2.5D maps) [42, §45.3.1] or 3D occupancy maps where 3D space is divided into voxels with occupancy probabilities [42, §45.3.2].

2.1.2 Planning robot behaviour

Once a robot has a representation of the world, along with information on how its actions will affect the world, robotic systems can plan their actions in order to achieve a given goal. When discussing planning models one can refer to robots as *agents*, ie. entities that perceive their environment and can act upon the environment with actuators. Agents range in complexity from simple reflex agents to complex utility-based or learning agents. One way to describe the mapping from a set of sensed data to its appropriate robot action is through the agent function, with the implementation of this function being the agent program [43, §2.1].

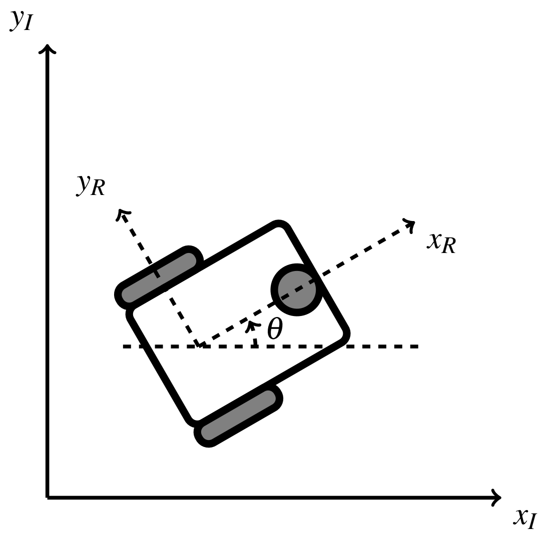
The design of the agent-program reflects what information is taken into account during the planning and decision process. In a simple reflex agent, only the current or latest percept is taken into account and mapped directly to an action. This can for example be an autonomous vehicle that has the instruction to drive forward at all times, but stop if it detects something in front of it. Model-based reflex agents take this further by maintaining an internal world state which has information that is not directly evident in the current percept. Goal- and utility-based agents also consider how the robot action will affect given goal or utility functions which are to be achieved or maximized. The most complex of these are the learning-based agents, which analyze the performance of the agents through time and learn behaviours which achieve the goal or maximize the utility more effectively [43, §2.4].

The states and behaviours making up the agent-program can be organized using several models, one of which is the Finite-State Machine (FSM). An FSM is a model of computation containing a set amount of states and state transitions, where the system can only be in one state at any time. In robotics, a state can describe a set of behaviours that the robot exhibits which is unique to that state. For example, a robot may be in a planning state after having received some percepts, and once a plan has been computed it will transition to an enacting state. If no such plan can be made with the current percepts, the robot may transition to a information gathering state. As such, complex behaviour can arise from simple definitions of robot states and transitions between the states [44].

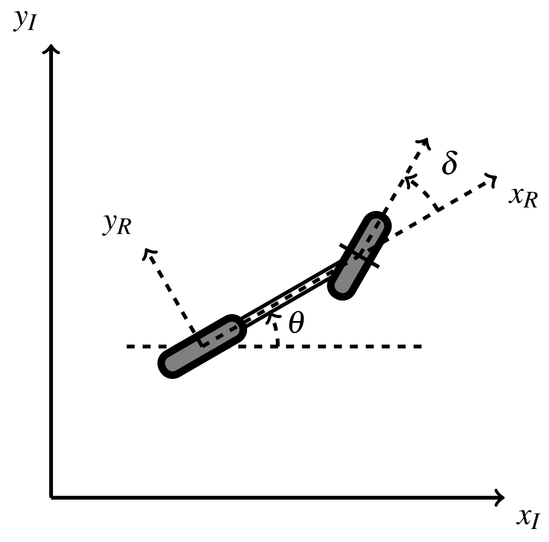
2.1.3 Vehicle models and motion control

Motion control addresses the issue of controlling the robot actuators such that the robot follows the given plan. Constraints from the robot kinematics, dynamics and actuators must be considered to determine which motions are feasible for the robot, and are usually also taken into account during the higher level planning covered earlier. Kinematic constraints in robotics are often referred to as being holonomic or non-holonomic. Holonomic constraints depend only on system coordinates and potentially time, meaning that a system with only holonomic constraints can be accelerated in any direction [45, §1.3]. Non-holonomic constraints may depend on derived system states, such as speed or acceleration. An example of a non-holonomic constraint is the no-slip constraint on wheels, leading to cars not being able to instantaneously move sideways. Such constraints complicate motion control and must be accounted for.

Various vehicle models exist to represent these constraints on vehicle systems. For wheeled ground vehicles, the kinematic properties and constraints of the system usually revolve around the wheel configuration and actuation. Popular models include the differential-drive model and the bicycle model. In the former, two actuated wheels placed with a shared axis can provide forwards and rotational motion, with additional non-actuated wheels to stabilize the platform. The latter relies on one wheel for actuation and the other for steering, prohibiting rotations of the body in place. These two models are illustrated in Figure 2.1.



(a) Differential-drive model.

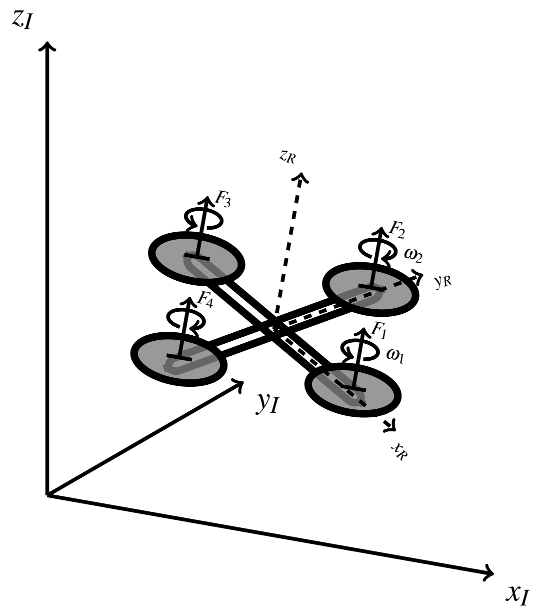


(b) Bicycle model.

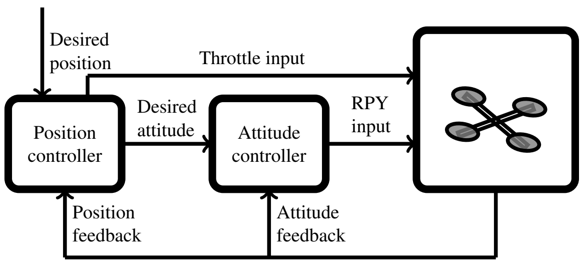
Fig. 2.1 Two wheeled vehicle models. (a) The differential drive model has two actuated wheels with a common wheel axle provide forwards/backwards and rotational movement. A third spherical wheel is added for stability. Motion control achieved through simple PID controller on position and orientation (discussed below). (b) The bicycle model has an actuated back-wheel and steerable front-wheel, closely resembling key characteristics of car-like robots (no instantaneous lateral movement, no turning in place). Motion control and planning can be achieved by parameterizing the path into lines and arcs[46, 47].

Aerial systems differ from their ground-based counterparts in that they must more actively consider all three dimensions, whereas ground systems can usually abstract away the z-dimension and only consider movement on the ground plane. A typical quadrotor aerial model is shown in fig. 2.2a. The quadrotor consists of four individual rotors fixed to a rigid cross-shaped frame, in pairs of clockwise- and anticlockwise-rotating rotors. Motion is achieved by controlling the thrust provided by each rotor. How thrust (upwards/downwards motion), roll and pitch are controlled is easily conceptualized. Yaw motion is achieved by adjusting the average speed of the clockwise and anticlockwise rotating rotors [48, §52.2.2].

Control strategies include feedback control of measurements to the vehicle model to minimize errors between reference and measured system states. In some systems, for example in quadrotor systems, feedback controllers may be layered such that an outer layer controlling position gives reference states to inner attitude controllers. An example of such a multi-layered control strategy is shown in fig. 2.2b. These feedback controllers often come in the form of Proportional–Integral–Derivative (PID)-controllers. Another control methodology includes the use of Model Predictive Control (MPC), where the motion control problem formulated as an optimization problem. Kinematic constraints on the body can then be directly accounted for by including them in the optimization problem.



(a) Quadrotor model.



(b) Hierarchical quadrotor control.

Fig. 2.2 The quadrotor UAV model: (a) Body model with propeller forces F, angular velocity ω and direction of spin denoted with arrows. Coordinate frames are denoted I for the inertial frame and R for the robot/body frame. (b) Illustration of hierarchical control of the quadrotor position using two controllers. Figure adapted and simplified from Figure 52.6 in [48].

2.2 Path planning

Robot path planning represents a large portion of problems within robotics. The general path planning problem can be formulated as the issue of planning collision-free paths that allow a complex body to move from one initial state to a goal state. Within this formulation it is understood that the environment can be populated by obstacles, and that an intersection between the complex body and these obstacles constitutes a collision which is to be avoided [49, §7.1].

Many formulations and additions to this problem exist which modify the path planning problem to cover other fields in robotics, such as exploration, formation holding, etc. These topics will be covered in later sections. Path planning is made distinct from the related problems of motion- or trajectory planning. The former results in a list of waypoints and/or geometric paths, while the latter two include the translations and rotations necessary to move a complex body along this path and how to respect the mechanical limitations of the body while doing so respectively[50, §1.1].

2.2.1 The path planning problem

First, a formal definition of the path planning problem is presented. A classical example which can be used in this definition is the *mover's problem*[1] [51]. Here, a polyhedron is to be moved through Euclidean space while avoiding polyhedral obstacles. The basic definition of the problem follows [49, §7.2.2].

Given:

1. A workspace $\mathcal{W}$, where either $\mathcal{W} = \mathbb{R}^2$ or $\mathcal{W} = \mathbb{R}^3$:
   The physical space the robots operate in, which is usually abstracted to two dimensions for ground robots.

2. An *obstacle region* $\mathcal{O} \subset \mathcal{W}$:
   The parts of the workspace occupied by obstacles which the robots must avoid.

3. One or several *robots* $\mathcal{A}_1$, $\mathcal{A}_2$, ..., $\mathcal{A}_m$ defined in $\mathcal{W}$:
   Robots, or agents, which can occupy space in the workspace and which move under certain constraints.

4. A configuration space $\mathcal{C} = \mathcal{C}_{free} \cup \mathcal{C}_{obs}$ ($\mathcal{C}_{obs}$ and $\mathcal{C}_{free}$ are defined later):
   The space of all possible robot configurations, with a configuration q being a complete specification of the location of every point on the robot geometry.

5. An *initial configuration* $q_I \in \mathcal{C}_{free}$.

[1]Also referred to as the *Piano mover's problem*

6. A *goal configuration* $q_G \in \mathcal{C}_{free}$. The initial and goal configurations are often called a *query* ($\mathcal{C}_{obs}$, $\mathcal{C}_{free}$).

Problem: *Compute a (continuous) path,* $\tau : [0,1] \to \mathcal{C}_{free}$*, such that* $\tau(0) = q_I$ *and* $\tau(1) = q_G$. The *obstacle region* $\mathcal{O} \subset \mathcal{W}$, which can be a collection of polyhedra, three-dimensional triangles, or piecewise-algebraic surfaces, is used to define the *C-space obstacle region* $\mathcal{C}_{obs}$. By letting $\mathcal{A}(q) \subset \mathcal{W}$ represent the set of points occupied by a robot $\mathcal{A}$ when it is in configuration q, the *C-space obstacle region* $\mathcal{C}_{obs}$ is defined as:

$$\mathcal{C}_{obs} = \{q \in \mathcal{C} \,|\, \mathcal{A}(q) \cap \mathcal{O} \neq \emptyset\}.$$

From this definition, the*free space* $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}$ is easily defined [49, §7.2.1].

2.2.2 Path planning algorithms

There exist a multitude of path planning algorithms to solve the aforementioned path planning problem, the choice of which will depend on the task environment and other characteristics such as complexity, processing time, etc. Algorithms can be divided into several broad categories, depending on how they formulate and solve the problem.

Optimization-based planning has already been mentioned in the context of motion planning covered in 2.1. Optimization methodologies such as MPC can be applied to the general path planning problem in continuous space by formulating the problem as an optimization problem, with the option of including obstacles and robot kinematics as constraints. However, in most formulations of planning, computing the optimal solution is given little or no importance. Factors that contribute to this are that the optimal path may potentially not be desirable, may be excessively long or might have a computational complexity which is far higher than their (not necessarily optimal) counterparts [49, §7.6.7].

Graph-based planning is an alternative to optimization-based planning which first transforms the environment into an appropriate graph-based representation, then plans a path through the graph. The graph G will consist of vertices $V \in \mathcal{C}_{free}$, representing discrete configurations, and edges $E \in \mathcal{C}_{free}$ connecting the vertices. Such a graph-based representation of the environment can be created by partitioning the free configuration space $\mathcal{C}_{free}$ into cells through cell decomposition. Each cell is then represented by a graph vertex, and neighboring cells are connected with graph edges. One example is the uniform cell decomposition created by discretizing $\mathcal{C}_{free}$ into equally sized cells (in either 2D or 3D) [50, §6.3].

Once a graph has been constructed, there exist many classical algorithms which can traverse the graph from a given initial- and goal-vertex to find the shortest path. Examples include Dijkstra's algorithm [52] and the A* algorithm [53]. Though these algorithms are efficient on the constructed graphs, the construction of the graphs require that $\mathcal{C}_{free}$ be known a priori. Not only is this not always the case, especially with dynamically changing environments, it is in some scenarios a central issue of the problem, such as for exploration scenarios.

Sampling-based planning algorithms are similar to graph-based planning algorithms in that a graph over $\mathcal{C}_{free}$ is constructed and used to find a collision-free route. However, sampling-based planners iteratively construct the graph by sampling individual configurations and potentially adding them to the graph. The standard goal for sampling-based planners is to provide a weaker, yet interesting, form of completeness: *if a solution path exists, the planner will eventually find it* [49, §7.3].

The Probabilistic Roadmap Method (PRM) is one such algorithm, in which a graph is constructed by sampling configurations in $\mathcal{C}_{free}$, adding them as vertices and connecting them to other nodes in graph that are in the neighborhood[2] and that have an edge that is entirely within $\mathcal{C}_{free}$. Though this provides a computationally efficient way to construct a graph that can avoid some of the downfalls of cell decomposition, it is still mainly a preprocessing step that runs before a graph-based planner finds the shortest path through it [49, §7.3.1].

Another family of sampling-based algorithms solves both the graph-construction and shortest-path problem simultaneously by constructing trees rather than graphs. A tree T is, like a graph, comprised of vertices and edges $T = \{V, E\} \in \mathcal{C}_{free}$. However, a tree can not be disconnected and must be acyclic. Each tree-vertex also has a unique parent-vertex, but may have any number of child-vertices which it is the parent of. The exception is the initial vertex in the tree, which is referred to as the root-vertex, which does not have another vertex as a parent. This results in a natural hierarchy arising from the distance (in nr. of vertices) to the root-vertex, and only one unique path from a given vertex to the root-vertex.

[2]Vertices that are under a certain distance threshold, in some metric defined in configuration space, are said to be in the same neighborhood.

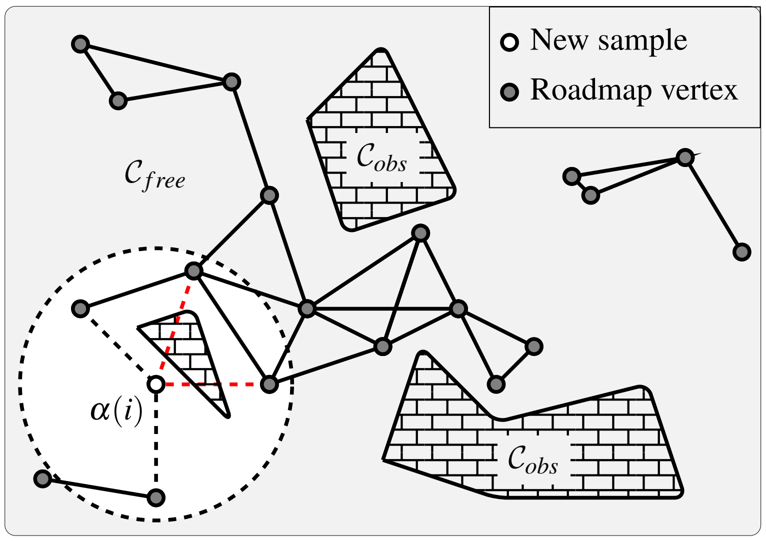


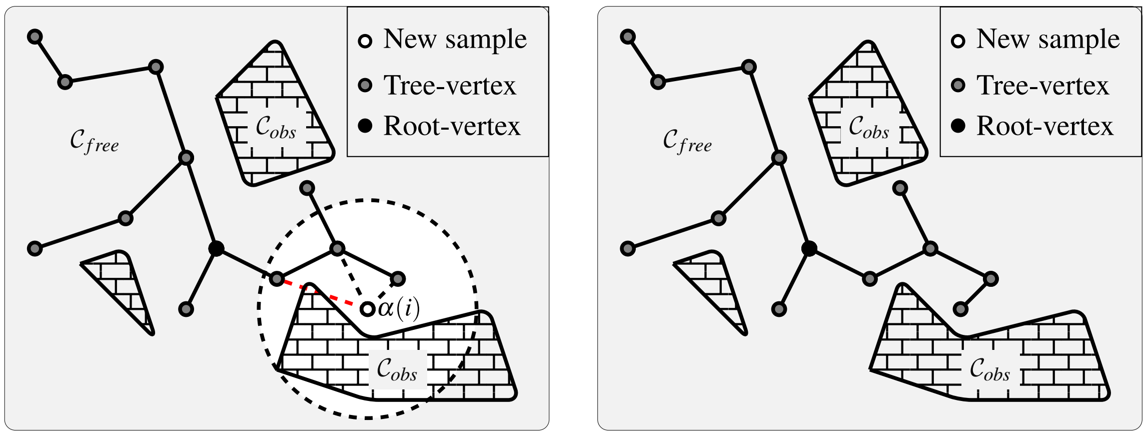
Fig. 2.3 Illustration of a possible roadmap created with the PRM. A sample $\alpha(i)$ is taken. Vertices in the neighborhood of $\alpha(i)$ are checked for collision free edges.

2.2.3 Rapidly-exploring Random Trees

The Rapidly-exploring Random Tree (RRT) algorithm constructs a tree from a given root-vertex by sampling new configurations $\alpha(i)$ and only connecting resulting vertices to one tree-vertex [54]. In the case of path planning, the initial configuration q_{init} is usually chosen as the root-vertex, and a tree is grown that attempts to have a tree-vertex end up within an acceptable neighborhood of the goal configuration q_{goal}[3]. Planners using RRT usually sample q_{goal} at set intervals or with a given probability.

In the classical RRT algorithm, the tree-vertex $q_{nearest}$ that is closest to the new sampled configuration $\alpha(i)$ is chosen for expansion. CLoseness can be defined in multiple ways, but is often related to euclidean distance. A state transition equation of the form $\dot{q} = f(q,u)$, which can express nonholonomic constraints for the robot given a configuration q and input u, is integrated over a given time to steer the robot from $q_{nearest}$ towards $\alpha(i)$. The integration results in a configuration q_{new}, which may or may not be the same as $\alpha(i)$, which is added to the tree with an edge to $q_{nearest}$. The RRT algorithm is shown in Algorithm 2 and Algorithm 3, and a graphic illustrating the algorithm is shown in Figure 2.4.
The original formulation of the RRT algorithm has several properties which make it ideally suited for path planning problems: It is relatively simple to implement, can include non-

[3]RRT is a single-query planning method, as opposed to the multi-query methods presented earlier that can answer the path planning problem for multiple combinations of initial- and goal-vertices [49, §7.3.2].



(a) RRT sampling. (b) RRT tree connection.

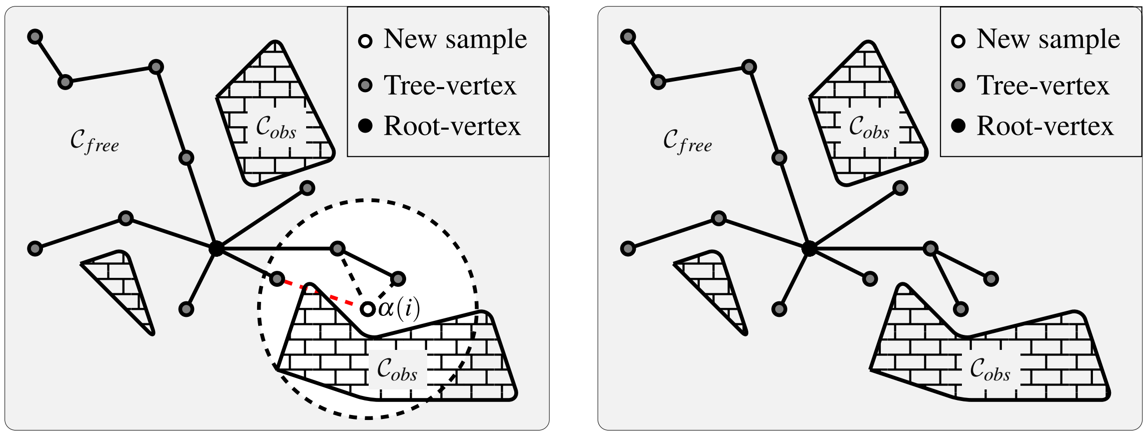
Fig. 2.4 The RRT algorithm grows a tree in the free configuration space by sampling new configurations and adding them to the closest tree-vertex. The tree is initialized with only one vertex, known as the root-vertex. (a) A sample configuration $\alpha(i)$ is taken. Tree-vertices in the neighborhood of $\alpha(i)$ are checked for collision free connection. (b) $\alpha(i)$ is connected to the closest tree-vertex in its neighborhood with which it can make a collision free edge.

holomic constraints in the planning through $\dot{q} = f(q,u)$[4], and the algorithm is able to find a feasible solution relatively quickly. Whether or not the algorithm could improve the initial solution was a subject of debate, and it was proven that the probability of the algorithm improving the initial solution to an optimal one was actually zero [55]. This lead to the development of RRT variants which could improve the initial result.

The RRT* variant of the RRT algorithm modifies the way new configurations are connected to the tree to make it an anytime algorithm, meaning it is able to quickly find a solution and to improve upon the solution with further processing [55]. The key modification is the inclusion of costs for each tree-vertex and the connection of q_{new} to the tree-vertex within its neighborhood that would result in the lowest cost for the new vertex. This cost is classically chosen as the distance within the tree from the tree-vertex to the root-vertex. An additional rewiring step is then performed by checking tree-vertices in the neighborhood of q_{new} and assigning q_{new} as the parent-vertex of these vertices if their resulting cost is reduced.

In algorithmic terms the RRT* algorithm modifies the EXTEND procedure of Algorithm 2, shown in Algorithm 4. A graphical illustration of the modification is shown in Figure 2.5. The RRT* algorithm is proven to be asymptotically optimal, meaning that is is able to converge to the optimal path between a given initial- and goal-configuration given that such a

[4]If the robot is considered holonomic the function returns the collision-free endpoint of a ray within $\mathcal{C}_{free}$.



(a) RRT* sampling. (b) RRT* tree connection.

Fig. 2.5 The RRT* algorithm grows a tree in the free configuration space by sampling new configurations and adding them to the tree-vertex with the lowest cost. The tree is initialized with only one vertex, known as the root-vertex. (a) A sample configuration $\alpha(i)$ is taken. Tree-vertices in the neighborhood of $\alpha(i)$ are checked for collision free connection. (b) $\alpha(i)$ is connected to the tree-vertex in its neighborhood that has the lowest cost (distance to root-vertex) and with which it has a collision free edge.

path exists. The additional steps required by the RRT* algorithm also do not incur substantial computational overhead when compared to the original RRT, meaning that asymptotic optimality can be ensured without losing the benefits of finding an initial feasible solution quickly.

Other parts of the algorithm can also be modified to improve convergence. The Informed-RRT* algorithm applies a heuristic to the way new configurations $\alpha(i)$ are sampled to improve convergence-time to the optimal path [56]. It does this by constraining the sampling-region after an initial path is found, such that new samples are taken within a prolate hyperspheroid including the initial-and goal-configurations. The dimensions of the hyperspheroid depend on the current best path length, such that new samples will only be able to improve on the current best path. The modified sampling function is shown in Algorithm 5.

Variants of the basic RRT method exist outside of the pure robot path planning case. In Informative Path Planning (IPP) problems, the RRT algorithm can be used to detect interesting regions to explore. The algorithm is also easily extended to multi-robot scenarios since each robot can maintain its own tree. However, ensuring effective collaboration and avoiding collision in such scenarios is a much broader topic that will be covered in Section 2.3.

2.2.4 Informative path planning

Informative Path Planning (IPP) tackles the issue of maximizing the information collected by a sensor-equipped robot, subject to budget constraints on the planned paths [57]. Like the issue of path planning, solving IPP problems involves searching over large and complex spaces of possible paths, with particular formulations being NP-hard [58] or even PSPACE-hard [51]. In fact, the IPP problem is often compared to and formulated as a variant of the *travelling salesman problem*, where a robot must visit all unexplored places while minimizing the total travelled distance. This has lead to several methods which use heuristics to solve the problem in non-complete ways.

Frontier-based exploration is one such method, which can be summarised as: *To gain the most new information about the world, move to the boundary between open space and uncharted territory* [59]. Here, the IPP problem is considered in the exploration scenario, where the goal is to cover unmapped areas of the environment until the environment is sufficiently mapped[5]. The boundary regions between mapped and unmapped space, known as frontiers, are chosen as navigation goals for the underlying robot path planner, with the assumption being that these regions will be explored once a robot is moved in the vicinity of it. This results in the identification of frontier regions being the main IPP problem to be solved.

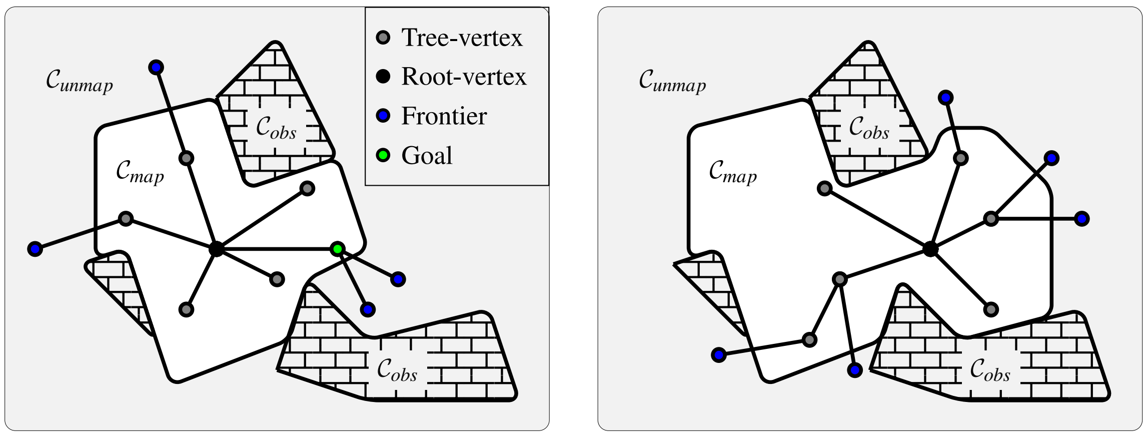
Another method which tackles the IPP problem more directly is to consider the information that can be gathered along a path. If one considers a robot equipped with directional sensors, such as cameras, the path can be seen as a sequence of views related to the direction the sensors are facing. The goal then becomes to determine the sequence of views which adequately cover the scene [61]. This method is similar to the frontier region method in that views with high levels of information gain will be views that cover the frontier regions.

Solutions to the issue of estimating information gain from each configuration/view will depend on the sensor characteristics and environmental representation. Omni-directional sensors allow the estimation of information gain as the size of unmapped, visible space within a sensor radius of the robot. In the case of an occupancy grid representation of the environment, this translates to the amount of unmapped, visible grid cells within the radius of the robot sensors. For directional depth sensors one must consider the Field of View (FOV) and maximum depth. Information gain can then be estimated through ray-casting within the

[5]Robotic exploration is closely related to the field of robotic mapping, and the terms are sometimes used interchangeably [60].

FOV and counting intersections with unmapped polygons [62, 63].

The RRT algorithm can be applied to make systems which utilize both frontier-based and view-based methods. Classically, image processing edge detection methods have been used on occupancy grids representing the robot environment to detect frontier regions. However, frontiers can also be detected in the tree produced by the RRT algorithm by detecting if edges cross from mapped to unmapped space [64]. This method also scales beyond the two dimensional case of occupancy grids. A visualization of frontier detection and exploration using the RRT algorithm is presented in fig. 2.6 for a robot equipped with an omni-directional sensor.

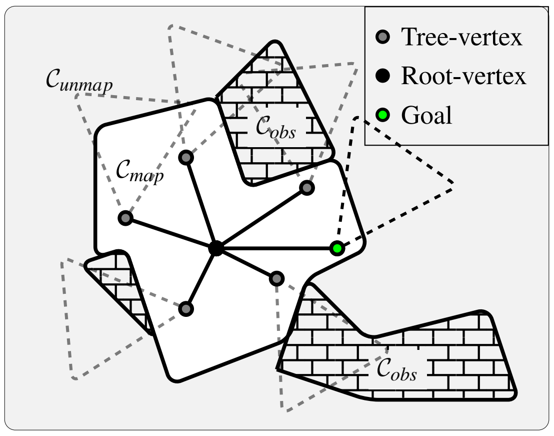


(a) A tree is grown using the RRT algorithm and frontiers are detected where edges cross from mapped to unmapped space. A navigation goal for the robot is chosen when the RRT algorithm ends.

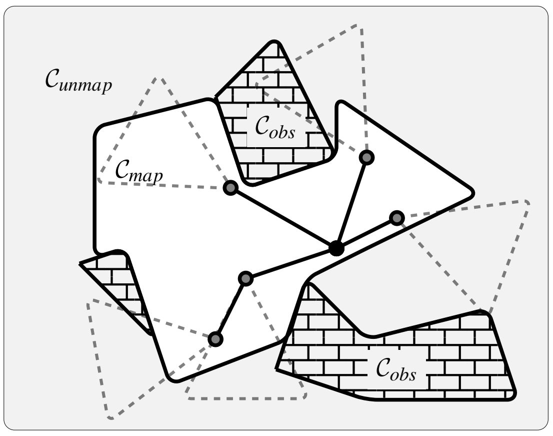
(b) The robot moves to the navigation goal, mapping unmapped space as it approaches the frontier. When the navigation goal is reached, the RRT frontier detection algorithm is repeated. Exploration continues until no new frontiers are found.

Fig. 2.6 The RRT can be used in frontier exploration to detect frontier points. Here, a tree is grown from the robot location represented by the root-vertex. Mapped space is represented by white areas, and unmapped space in gray. Space is mapped by moving the robot close to unmapped space, assuming that the robot sensors map the area directly around it.

The RRT algorithm can also be applied for view-based exploration by considering the orientation of the robot at each tree-vertex configuration. The information gain of the tree-vertices can then be estimated through the aforementioned methods, and the vertex with the highest amount of information gain can be chosen as the navigation goal. After navigating the robot to the first view in the sequence, the tree can be regrown from the new location. Resampling the previously selected path leads to a receding horizon behaviour for IPP [65]. This variant of RRT-based exploration is visualized in fig. 2.7.



(a) A tree is grown using the RRT algorithm and robot FOVs and information gain are calculated at each vertex. A navigation goal for the robot is chosen when the RRT algorithm ends.



(b) The robot moves to the navigation goal, assuming the orientation that is associated with the vertex. When the navigation goal is reached, the RRT view-based algorithm is repeated. Exploration continues until no FOVs covering unmapped space are found.

Fig. 2.7 The RRT can be used in view-based exploration to find configurations with a high information gain. Here, a tree is grown from the robot location represented by the root-vertex. Mapped space is represented by white areas, and unmapped space in gray. Space is mapped by moving the robot close to unmapped space, such that the robot FOV covers unmapped space. The robot FOV is here represented by triangles, as if seen from above.

Other IPP methods which utilize tree-like datastructures similar to that used in the RRT algorithm have also been demonstrated. The Sensor-based Random Tree (SRT) method applies the RRT concept to robots with omnidirectional ball- or ray-based sensors, such that a robot can be moved along an incrementally growing tree in $\mathcal{C}_{free}$ and backtrack on the tree once no local frontiers are available [66]. The Gap Navigation Tree (GNT) method attempts to map an area using a minimal representation based on sensor measurement events, utilizing a tree data structure made up of these events. Central in the method is the use of discontinuities in the depth measurements with respect to the heading of the robot, known as *gaps*, which in some senses resembles the frontiers presented earlier [67]. Other methods based on sampling roadmaps, trees and graphs for IPP have also been developed for environments where the information quality can be extracted as a function of the environment [57].

2.3 Multi-robot planning

The motivations for multi-robot systems largely depend on the task environment the system is applied to (whether they are inherently distributed or depend on multiple robots). Their usage can also be motivated by the added robustness of having multiple robots. Multi-robot systems and the problems they are applied to can be characterized along several attributes [50, §7.2]:

Centralized or decentralized, relating to whether decisions are made by one centralized controller or by each individual robot using global or local data respectively.

Homogeneous or heterogeneous, relating to the differences in robot capabilities within the group, such as motion constraints or sensing abilities.

Unlabeled, k-color or labeled, relating to whether tasks can be done by any arbitrary robot in the group, if there is a number (k) of different tasks and robot types, or if each task must be accomplished by a particular robot.

Systems which are highly decentralized and homogeneous can be characterized as collective swarm systems. In such systems, each robot has a high degree of independence and operates without explicit communication with other robots, often merely treating other robots as dynamic obstacles in the environment. Such systems can achieve an emergent global behaviour, such as flocking or coverage, using relatively simple control laws for the individual robots. However, some tasks require or benefit from explicit coordination and communication between robots.

2.3.1 Intentionally cooperative systems

Intentionally cooperative systems can solve a broader range of problems at the expense of requiring communication and explicit coordination between the robots. Strongly cooperative systems tackle tasks that are not trivially distributed and where coordination is essential, while weakly coordinated systems allow robots to operate with periods of independence following coordination phases.

Planning and coordination can be done in a centralized manner given that the planner has all the relevant information from all the robots. The concept of the configuration space presented in Section 2.2.1 can be extended to multi-robot systems by concatenating the individual robot configuration spaces and planning in the joint configuration space [49, §7.6.4]. This method quickly suffers from the *curse of dimensionality* as more robots are considered, with the number of vertices required to cover each dimension with a certain density growing exponentially with the number of robots.

Another centralized planning methodology involves the assignment of priorities to each robot [68]. Each robot can then make its plans individually in a sequence following the priorities, having to only consider the plans of the robots with higher priority (for example by treating them as dynamic obstacles in the path planning case). Though the curse of dimensionality is avoided since the planning algorithm now scales linearly with the number of robots, the scheme used to assign priorities can still cause issues. This is evident in the example presented in Figure 2.8. Some schemes include random priorities or priorities which arise naturally based on the importance of individual robots or their tasks. Other priority schemes can be generated using genetic algorithms, like hill climbing [69], or graph based heuristics [70].

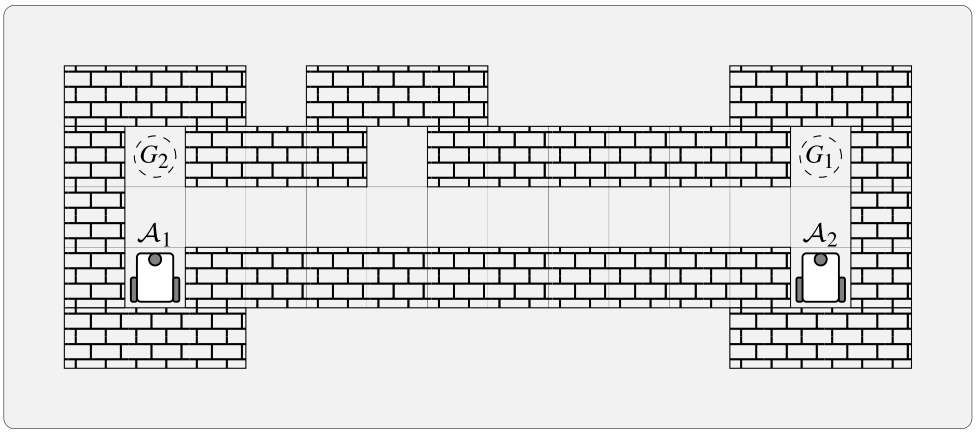


Fig. 2.8 An example scenario where prioritization significantly influences the result. If robot $\mathcal{A}_1$ has the highest priority when planning its path, robot $\mathcal{A}_2$ can only start moving after $\mathcal{A}_1$ has reached its goal configuration q_{g1}. If robot $\mathcal{A}_2$ has priority, $\mathcal{A}_1$ can use the cavity to let $\mathcal{A}_2$ pass, thereby giving a better (less time used) result. Example adapted from [70].

Decentralizing the coordination in an intentionally cooperative system places emphasis on the networked communication between robots, as each robot is able to make more informed local decisions by knowing the global situation communicated by other robots. The field of networked robotics tackles this problem by analyzing the propagation of information through a robotic network and its effects on control, planning and perception [71, §53.4]. Game-theoretic methods, such as voting, can be used in such systems to communicate and come to agreement on a global solution from each robot's local solution [72]. By formulating the multi-robot system communication topology as a graph, graph-based algorithms like consensus can be utilized to reach a common solution despite communication delays and noise

[73]. Planning robot motions taking communication constraints explicitly into consideration is another field of robotics which has been studied extensively, and which will be discussed in section 2.3.4.

2.3.2 Multi-robot Informative Path Planning

Multi-robot Informative Path Planning (MIPP) concerns the extension of IPP to the multi-robot case, which can be achieved with various levels of cooperation and coordination. Simply running individual IPP systems on each robot and ensuring collision avoidance between robots is likely to improve the speed at which information is gathered. For example, running frontier-based exploration on multiple robots sharing a common map which they can localize themselves in improves on the performance done by a single robot running the same system [74]. However, IPP and robotic exploration present a scenario where coordination can intuitively improve performance, simply by considering that the same region cannot be explored twice.

Coordinating exploration frontiers through task assignment is a simple way to add a collaborative element to the system. Assigning a priority to each robot and considering them sequentially allows each robot to choose its best exploration frontier, and then weight the information gained on that frontier negatively for other robots who have yet to choose a frontier [75]. Frontiers can also be assigned more intelligently to avoid the downsides of fixed priority planning, such as through bidding in market economies [76] or through assignment algorithms such as the Hungarian Algorithm [77].

Expanding beyond the basic frontier exploration paradigm, methods have also been developed to divide the remaining unexplored space between the robots. Unsupervised clustering algorithms, such as the K-Means algorithm [78], can be used to partition the unmapped cells of an occupancy grid into as many clusters as there are robots, outperforming the greedy-like frontier approaches [79]. Other partitioning methods have been presented where the structure of the environment is taken into account, such as by using Voronoi Graphs to detect rooms and hallways for exploration in indoor scenarios [80].

Sampling-based techniques have also been applied to the MIPP field. The SRT algorithm for single-robots has been expanded to include multiple robots [81]. SRTs are built in parallel by each robot, with the ability for robots to expand on other robot trees once their own trees are fully expanded. This concept is extended to include bridge-connections between trees such that the trees are connected into a Sensor-based Random Graph (SRG) [82].

Concepts from the related field of Collaborative Visual Area Coverage (CVAC) can also be utilized for MIPP systems, particularly in the domain of collaborative IPP where sensor coverage and overlap is to be considered. The CVAC problem concerns placing sensor-equipped robots such that the sensors cover a predefined area adequately, and can usually be split into two categories: blanket- or sweep-coverage [83]. In the former, the area is statically defined and the result is a static placement of the robots. The latter, also known as dynamic coverage, tasks the robots with covering a changing objective, potentially resulting in constant movement for the robots [84, 85].

Many methods within CVAC abstract sensor coverage over the environment into simple shapes, such as circles, rectangles or trapezoids [86]. Calculating the region of overlap between different robot sensors can then be simplified to a geometric problem. Once a region of overlap has been found, some systems systems choose to only account for information gain in that region from the sensor with the best coverage quality [87]. Systems which utilize camera sensors can also consider overlap through an area pr. pixel metric, allowing (and in some cases encouraging) sensor overlap [88].

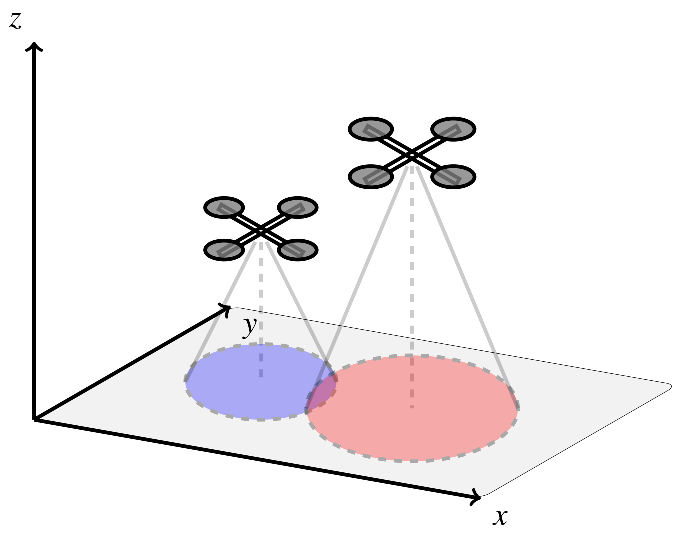


Fig. 2.9 Robots in a coverage scenario may have to tackle overlapping sensor coverages. For the case of UAVs, altitude might alter the scale and quality of coverage.

Attaching the sensors to mobile robot which are able to move in three dimensions allows for the deformation of the sensor coverage over the two dimensional ground plane. This is evident in the example of UAVs with downward facing cameras that have an FOV of the ground plane proportional to their altitude. Systems which consider this usually apply an information quality metric to disencourage the robots from placing themselves unrealistically high [87]. The ability to manipulate the sensors position and/or orientation on the robot further complicates the issue. Camera sensors with controllable pan- and tilt-angles are an example of this [89].

The CVAC problem can be translated to solve exploration tasks by having unmapped space be the area selected for coverage. Naturally, this area will shrink as unmapped areas are covered and subsequently mapped by the robots. Robots which are in some way constrained in their planning, such as being made to maintain a formation or maintain communication constraints with other robots in the group, can have this area reduced to the unmapped space that can be sensed given the robot constraints.

2.3.3 Formation control

Path planning in multi-robot systems can also be considered by requiring the robots to maintain a formation and considering the movement of the robots as one group. The advantages of this approach depend on the use-case. For MIPP and exploration, maintaining a static formation allows for path planning of the whole group as if it were one robot equipped with all the sensors of the individual robot (this naturally makes assumptions on the communication between robots).

Several metrics exist which characterize robot formations [90]:

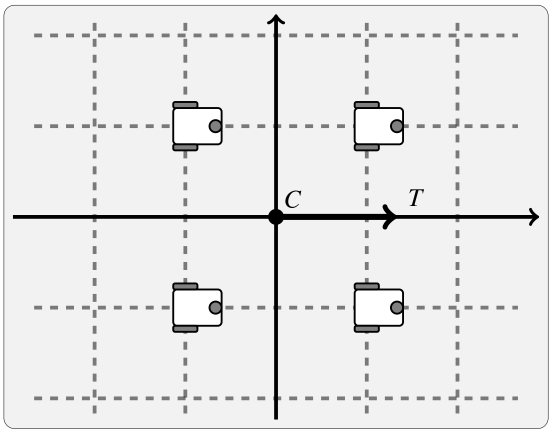
Rigid or flexible, relating to whether inter-robot distances are fixed or are allowed to vary over time.

Point-referenced or neighbour-referenced, relating to whether robots reference their position in the formation to some common reference point or to if they position themselves relative to one or more neighbors.

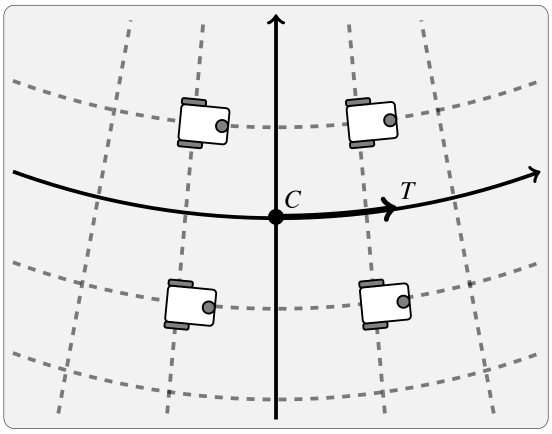
Rigid formation systems are usually applied when there are physical restrictions involved, such as the robots having to carry or manipulate a physical object by holding the object's corners. When such restrictions are not in place, flexible systems allow robots to make local control decisions that deviate from their formation position. Relaxing the formation constraints even further leads to flocking type systems, where robots are required to move together along some path with only minimal requirements for the paths taken by specific robots (usually only collision avoidance, such as in swarm systems) [71, §53.10.2].

Neighbor-referenced formations rely only on the robots' ability to sense and locate themselves relative to their immediate neighbors. This is ideal for robots with limited sensing and control capabilities where one wants to achieve a certain behaviour with simple control laws [91]. Despite this, intentionally cooperative multi-robot systems are usually point-referenced, as this allows for more complex and coordinated behaviour. The reference point can be some arbitrary point on the trajectory that is to be followed or the position of one of the robots, the latter being an example of a leader-follower type formation system.

For point-referenced systems where the point to be followed travels along a curving trajectory, the formation could also be maintained within a curvilinear coordinate system rather than the normal rectilinear one. Such a coordinate system has an axis that curves along with the reference trajectory, as shown in Figure 2.10. This takes advantage of the nonholonomic constraints present in some robot systems and ensures collision avoidance if the turn is not sharper than a threshold curvature [90].



(a) Square formation while moving straight.



(b) Square formation while turning.

Fig. 2.10 Curvilinear coordinates applied to four robots maintaining a square formation relative to the reference point C while following a trajectory T. Adapted from Figure 1 in [90].

Robot position and distance relative to the reference point can be maintained by the local control laws of the robots. For holonomic robots, maintaining the formation is trivial as simple control laws can be applied to drive the robots to the desired formation position. Several methods have been investigated for nonholonomic robots that have to follow a specific trajectory. These include the aforementioned curvilinear formation specification and path synchronization algorithms [92]. However, local controllers for nonholonomic systems can in many cases be relied upon to drive the individual robots to their desired positions such that these constraints don't have to be considered globally.

Some situations may also benefit from having the formation itself change over time, such as in environments populated with obstacles where maintaining a predefined formation would lead to collisions. One alternative is to scale the predefined formation, such that it can shrink to pass through narrow corridors and grow to sweep larger unoccupied areas [90]. Formations can also be selected from a predefined set of formations, potentially preferring certain formations and weighting this preference against the need to avoid collisions [93]. The utility of keeping a formation may also be weighted against other utilities, such as information gain, such that a formation adapts to gather more information while still keeping group cohesion and tracking a trajectory. This specific use-case has been touched upon in the previous section on MIPP.

For both static and varying formations, tracking a trajectory in an environment populated by obstacles remains a challenging task. Systems which calculate the required robot trajectories offline may make use of optimization techniques, such as reformulating the problem as a Mixed Integer Linear problem [94]. Systems which must calculate these trajectories online may make use of heuristics, such as adapting the sampling-based RRT algorithm to apply to formations of robots [95].

Another direction for online collision free formation control is the formation-containment control problem, where a convex region of free space is calculated around the formation within which the formation must reside. This is usually applied to leader-follower type systems where the convex region is defined with respect to the leader position [96]. Methods exist which can quickly calculate such regions in two- and three-dimensional space [97]. Finding collision free trajectories within these well-defined regions can then potentially be done through constrained optimization and consensus [98] or random sampling [99].

2.3.4 Communication constrained planning

In Section 2.3.1, the requirements for communication in intentionally cooperative networked robotics systems was mentioned. Multi-robot planning methods which don't consider communication constraints explicitly have been widely studied [100], and a majority of the previously presented systems fall into this category. Systems which do take such considerations have been investigated less, though they have been gaining attention in recent years. A key implication of communication constrained planning is that robots rely on communication with other robots to share information, often about the environment they are operating in. This is especially applicable to collaborative exploration scenarios where a shared map is assumed, as presented in Section 2.3.2. This section will present some such Communication Constrained Multi-robot Informative Path Planning (CCMIPP) systems, while also presenting general background theory on communication constrained planning.

Communication constrained robot systems can be classified in several ways [101]. The first way to differentiate such systems is in what communication models they use. These models define the circumstances under which two or more robots are considered to be in communication. Though such models may not accurately model the signal bandwidth between robots, they are usually designed to be conservative enough such that a certain level of communication is ensured when robots fulfill the requirements. Some common communication models include:

None: Communication between robots is not considered during planning, and robots rely on communication occurring due to robot encounters.

Range: Communication is possible between robots that are within a set range of each other.

Line-of-Sight (LoS): Communication is possible between robots that have an unobstructed line of sight between each other, meaning the line between the robots lies entirely in free space.

Signal: Communication is possible between robots according to a probability function that estimates the signal power between their respective positions. A possible equation for the signal power $P(d)$ at distance d, based on empirical relations, is the *Floor Attenuation Factor* propagation model shown in Equation (2.1) [102]. Here, $P(d_0)$ is the reference signal value at distance d_0, n indicates the rate at which the path loss increases with distance, nW is the number of walls/obstructions between the locations and C is the maximum number of walls/obstructions before wall attenuation factors WAF make a difference.

$$P(d) = P(d_0) - 10n\log_{10}\left(\frac{d}{d_0}\right) - \begin{cases} nW * WAF, & \text{for } nW < C \\ C * WAF, & \text{for } nW \geq C \end{cases} \tag{2.1}$$

For range and LoS constrained systems, location pairs can be determined as either fulfilling or breaking the communication constraints. As such, with knowledge about the environment for the LoS case, robot paths can easily be checked such that they conform with the constraints. Range and LoS constraints are usually chosen in communication constrained planning systems, since such hard constraints can be checked and included relatively easily to existing systems while also emulating more realistic communication constraints [101].

Signal models may also be used as binary models when they are considered with a threshold, or they can be added in the utility function of the planner. For CCMIPP problems, some systems factor in the signal strength to assess the information quality that can be transmitted from a robot location to a fixed Base Station (BS) [103]. The signal communication model can also be combined with other models, such as by assuming that high-priority short messages can be passed without major loss to robots within a certain range while one must account for loss when sending dense data such as maps [104].

Beyond the communication model used, communication constrained planning systems also differ in when they require communication to be in place. Communication constrained systems can broadly be categorized into two categories depending on when communication constraints are enforced:

Event-based connectivity: Connectivity must be established according to an event-based policy. Such events include the discovery of new information that must be communicated, chance meetings with other robots in the system or can be tied to a fixed time or rate. Outside of these events the robots may operate independently without communication constraints.

Continuous connectivity: Connectivity must be maintained continuously with other robots in the group, either directly or in a multi-hop fashion through other robots. Losing communication may trigger backup- or recovery behaviour to regain communication.

Requiring only event-based connectivity allows for path planning that is "more free". A naive solution demonstrating this is to ignore communication constraints until the event is triggered, at which point the relevant robots are tasked with rendezvousing at a set location to regain connectivity. This can be exploited in the exploration scenario where a fixed non-mobile BS is present, as robots can return to relay newly discovered information to the base station once a sufficient amount of information has been gathered by a robot [105]. Coordination in systems with event-based connectivity requirements can also be beneficial, such as through coordinating robot paths where connectivity is regained in the final path configuration [106].

Requiring continuous connectivity between the robots imposes higher levels of restriction to robot movement. Such requirements may be necessary where continuous streams of data need to flow in the system, such as video streams from robot cameras to human operators at a BS. Some systems use a "milder" form of continuous connection by not taking connection into account when planning but immediately reverting to recovery behaviours once connection is lost [107]. However, a majority of continuously constrained systems plan their actions such that connection is maintained. This can however lead to deadlock situations, such as when multiple robots move in opposing directions to reach their respective exploration frontiers [108].

The most restrictive form of continuous communication occurs when communication must be maintained with a fixed BS. Systems where each robot must have a direct connection to the BS are limited to moving in an area close to the BS, such as within the communication radius of the BS in the range constraint case. This problem can be overcome by allowing robots to act as relays, and many such systems have been investigated. One solution is to specifically assign robots as being relay- or explorer robots, with the former maintaining a communication network to the BS while the latter explore the frontiers [109]. The problem may also be considered as a static formation control problem, where the formation to be held is one which allows communication between the BS and the robot group [110]. Online switching of robots to act either as explorers or relays has also been investigated [111].

2.4 Conclusion

This chapter presented theory and background which is relevant to the systems that were implemented for this project. The first section covered some supplementary material which is relevant in understanding model robotics systems and their components. This background knowledge is needed to contextualize the next section, which covered path planning theory for single-robot systems. Special emphasis was placed on sampling-based path planning algorithms such as RRT, and on path planning in information-related scenarios such as exploration.

The last section covered both theory and literature on multi-robot systems, which is the main focus for this thesis. Metrics to classify multi-robot systems were presented, along with common applications of multi-robot systems in informative path planning, formation control and communication constrained planning. Special attention was given to systems in the literature which can be applied to the problem presented in Section 1.2, beyond the systems presented in section 1.1 which focussed mainly on UGV-UAV collaborative systems specifically.

The material presented in this chapter will be referenced when presenting and justifying the design and implementation of the system, which is presented in Chapter 4. In the next chapter, background material related to the robotics framework and previously implemented systems will be covered.

Chapter 3

Framework

This chapter presents the software frameworks that have been used to actualize the systems created in this project. The frameworks include open-source robotics frameworks, simulator software and robotics systems which drive the individual robots. These topics will however not be covered in depth. Most of the work in setting up and creating the necessary framework was conducted in the Specialization project prior to this master thesis. Any minor changes to the general framework which were necessary to create the new project systems, but which do not constitute a main contribution of this project, are presented within this chapter. For more details the reader is referred to the Specialization project thesis [1]. This includes information on the implementation of the RRT algorithms, detailed descriptions of the provided systems and the specific changes made to these systems.

The chapter begins by presenting the main overarching software frameworks that enable the development and testing of the project systems. These frameworks do not constitute a contribution from the author, but an understanding of them is necessary when presenting the authors contributions. The subsequent sections present the robotics systems used for each vehicle platform, which are a combination of open-source software and systems developed by the author in the Specialization project. This includes the individual exploration planners that will be utilized in some of the planners presented in Chapter 4.

3.1 General robotics framework

Modern robotics have trended towards standardized frameworks and a growing collaborative community. Central in this trend are open-source frameworks such as the Robot Operating System and the accompanying ecosystem. Simulation software such as Gazebo has also been critical in testing systems safely and quickly. Both of these frameworks are central in this project work.

3.1.1 The Robot Operating System

The ROS is a collection of tools, libraries and conventions which make up an open source robotics framework. The goal of ROS is to simplify the creation of complex and robust robotics systems for many platforms and use-cases[1]. A key element of ROS is collaboration, and a wide variety of robotics systems are available as `ROS packages` in the ROS Ecosystem. The core packages available in the default ROS installation handle monitoring of processes and message passing between systems[2], while packages for sensor data processing, Graphical User Interfaces (GUIs), control, hardware interfacing, etc. can be installed by the user.

ROS processes are typically contained within ROS Nodes[3], which can communicate with other nodes through communication channels such as Topics[4], Services[5] or Actions [6]. A Topic can be advertised and published to by one Node, while other Nodes can subscribe to the topic to receive the data that is published there. Services and Actions operate with a client-server structure, with the former being used for short queries and the latter being able to handle queries which require a longer time to fulfill. As Nodes only interface with other Nodes through these communication channels, the ROS structure can be very modular as packages can switch out Nodes so long as the communication interface is maintained.

The core functionalities of ROS are augmented by a suite of tools and libraries. The `tf` library maintains the relationships of transformations between coordinate frames, allowing sensor data captured in one frame to be related and represented in other frames efficiently [7] [112]. Process introspection can be achieved using the `rqt` package, which allows the viewing of the ROS Node graph and `tf` transformation tree. `RViz` is a 3D visualization tool which allows users to view ROS data graphically, and is the standard GUI for viewing sensor data, the robots representation of the environment, its plans and its current state[8].

[1]https://www.ros.org/about-ros/
[2]https://www.ros.org/core-components/
[3]http://wiki.ros.org/Nodes
[4]http://wiki.ros.org/Topics
[5]http://wiki.ros.org/Services
[6]http://wiki.ros.org/actionlib
[7]http://wiki.ros.org/tf
[8]http://wiki.ros.org/rviz

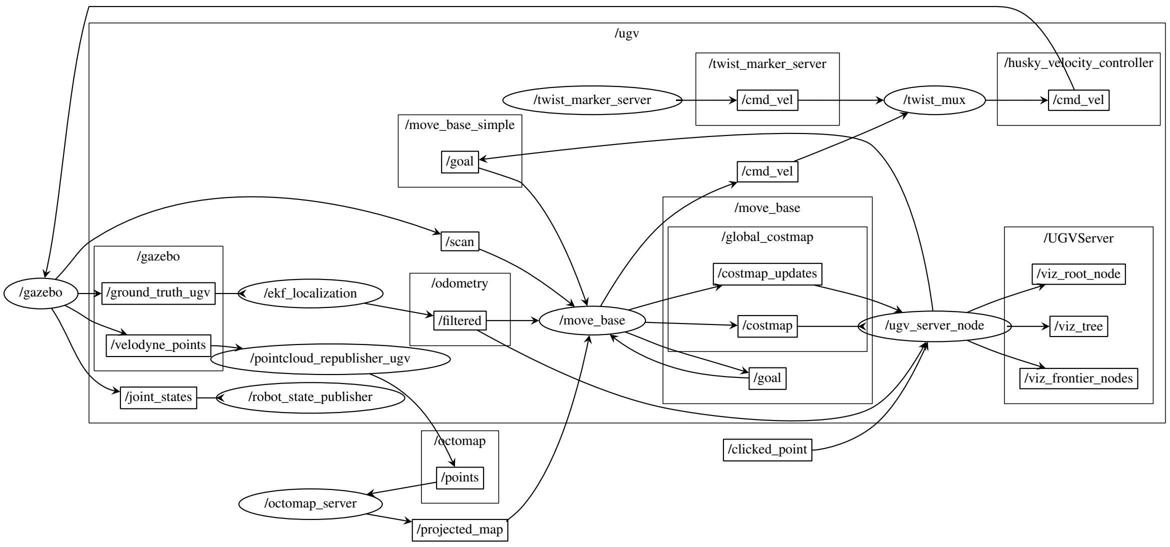
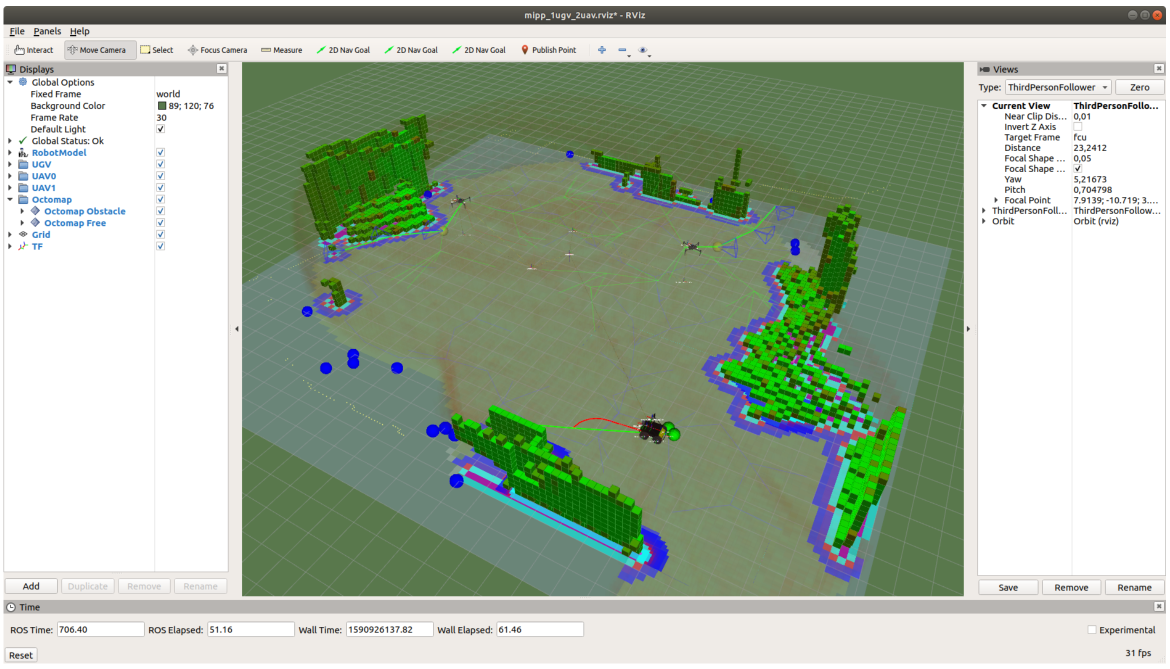


Fig. 3.1 Example of the ROS Node and Topic graph, here for the UGV system [1].



Global Options
Fixed Frame world
Background Color 🟩 89; 120; 76
Frame Rate 30
Default Light ✓
Global Status: Ok
RobotModel ✓
UGV ✓
UAV0 ✓
UAV1 ✓
Octomap ✓
Octomap Obstacle ✓
Octomap Free ✓
Grid ✓
TF ✓

Type: ThirdPersonFollower

Current View	ThirdPersonFollo…
Near Clip Dis…	0,01
Invert Z Axis	☐
Target Frame	fcu
Distance	23,2412
Focal Shape …	0,05
Focal Shape …	✓
Yaw	5,21673
Pitch	0,704798
Focal Point	7.9139; -10.719; 3.…
ThirdPersonFoll…	ThirdPersonFollow…
Orbit	Orbit (rviz)

ROS Time: 706.40 ROS Elapsed: 51.16 Wall Time: 1590926137.82 Wall Elapsed: 61.46 ☐ Experimental

31 fps

Fig. 3.2 `RViz` interface for the combined Specialization project system [1].

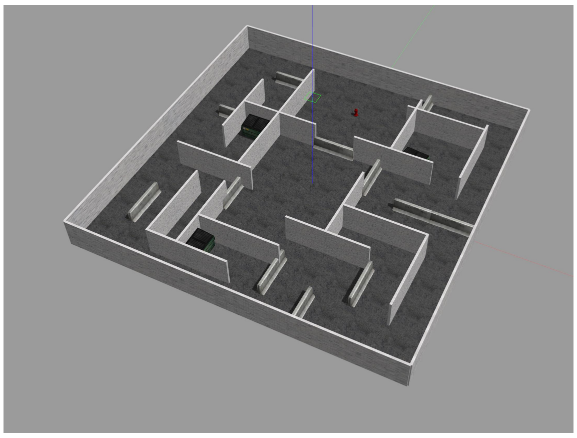
During the Specialization project, a multitude of the presented tools and frameworks were used when implementing the multi-robot simulation framework. All systems were written to interface with ROS. A selection of these systems are presented later in this chapter.

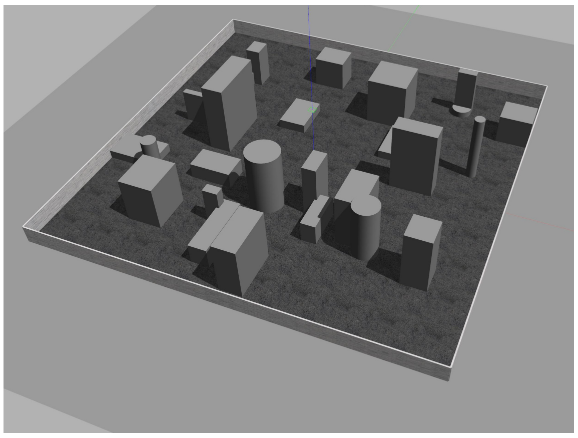
3.1.2 Simulation and representation

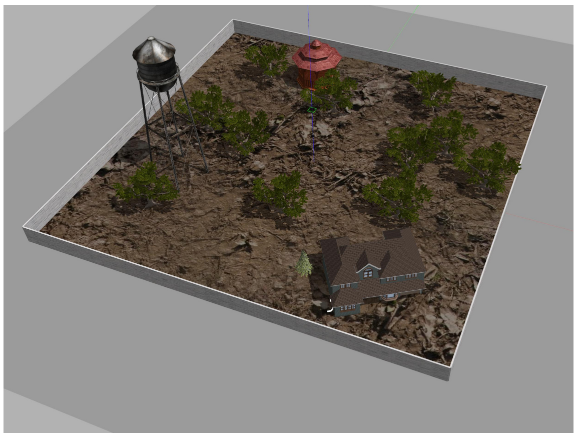
The Gazebo simulation environment is an open source 3D robotics simulator which makes use of high fidelity physics engines to simulate rigid body dynamics of robots and environments. Support code allows for the simulation of sensors and robot actuators, with an interface to ROS through the `gazebo_ros_pkgs` package group.

The Gazebo GUI allows for the creation of custom world environments for the robots to operate in. These can be populated by realistic objects with complex collision geometries, or simple shapes like cubes and cylinders. Several worlds were developed for this project, both in the Specialization project and in this project work. The worlds created for the Specialization project are shown in Figure 3.3. The worlds created for this project are shown in Section 5.2.
The OctoMap framework, and its interface with ROS through the `octomap_ros` package, are used extensively in this project. The framework can generate and represent a 3D volumetric environment using point cloud data from the simulated robot sensors [113]. A hierarchical octree-structure is used by the framework to recursively subdivide 3D cubic volumes into subcubes down to a fixed minimum voxel-resolution. Each cube will have a value representing either a binary or probabilistic occupancy value. Unmapped space is not registered with the tree, such that no value exists for unmapped voxels. Cubes which share occupancy values may be merged into larger cubes for efficiency. As the cubes are stored as vertices in a tree, traversal of the octree will have a logarithmic complexity.

For path planning, collision checking for points of interest in the OctoMap can be done both with an occupancy check and with ray casting. By defining a bounding box around the robot of interest, one can iterate through the leaf-vertices of the OctoMap contained within the bounding box and check their occupancies. The OctoMap API also has efficient methods for ray casting, where a ray intersection query is made from a point in a direction with a given maximum ray distance. If the ray intersects with an obstacle the collision can be detected, and this method is useful for checking the occupancy of straight-lines sections of the robot paths.


(a) `Maze`: A 30x30 meter maze with walls of varying heights.


(b) `Shapes`: A 60x60 meter world with shapes of varying heights and sizes.


(c) `Outdoor`: A 60x60 meter world simulating an outdoor scene, with trees and some buildings.

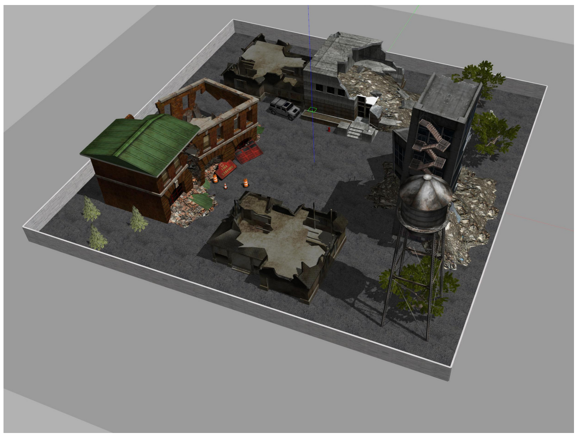
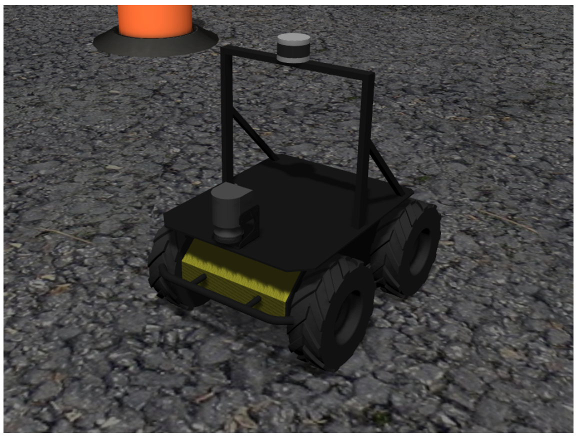

(d) `Disaster`: A 60x60 meter world simulating a disaster scene, with buildings and rubble.

Fig. 3.3 Gazebo worlds developed for the Specialization project [1].

3.2 UGV systems

The Clearpath™ Robotics Husky[9] is a four-wheel UGV. It has the possibility to be equipped with a multitude of sensors and is relatively simple to control due to the differential-drive nature of the platform. The Husky was chosen to simulate a general UGV system due to the availability of software to simulate it and interface it with ROS, as well as having a real life counterpart that is suitable for outdoor operations. A render of the UGV is shown in Figure 3.4, together with its simulated Gazebo counterpart. Software for simulating the UGV in Gazebo are provided, along with demo use-cases.

(a) Render of the Husky UGV without sensor load-out.



(b) Gazebo visualization of the Husky UGV, with 2D and 3D LiDARs.

Fig. 3.4 The Husky UGV by Clearpath Robotics. A medium sized four-wheel differential drive robotic development platform.

3.2.1 Sensors and planning

The UGV is by default equipped with sensors for both localization and range sensing. Data from a GNSS module, IMU and wheel odometers are fused by an Extended Kalman Filter (EKF) to provide the estimated UGV pose[10]. The default range sensor for collision avoidance is a 270 degree 2D LiDAR mounted to the front of the UGV. For the Specialization project, the depth camera mounted on the sensor arch was replaced by a 360 degree 3D LiDAR which could provide a point cloud measurement of the UGV surroundings for mapping.

[9]https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/
[10]EKF implemented in the `robot_localization` package[114]: http://docs.ros.org/melodic/api/robot_localization/html/index.html

UGV motion is controlled using the `move_base` ROS packages. Motion control and path planning systems are structured as a Navigation Stack[11] [115] made up of `move_base` packages. Planning is split into a global- and a local-level. Global planning makes use of a static user-provided map or a map constructed during run-time, and plans higher level paths in this map. The paths generated by the global planner are then followed by a local planner which uses a local map of the environment directly around the robot to plan around obstacles. Both maps come in the form of 2D occupancy grids and make use of the robot geometry and localization uncertainty to inflate occupied cells, assigning costs to having the robot center close to obstacles. The Navigation Stack is visualized in Figure 3.5.

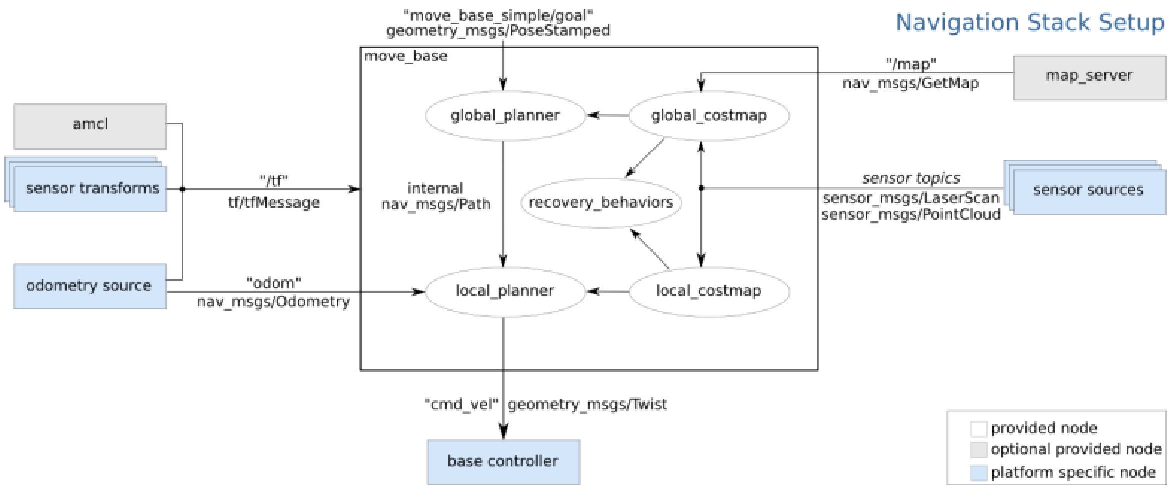
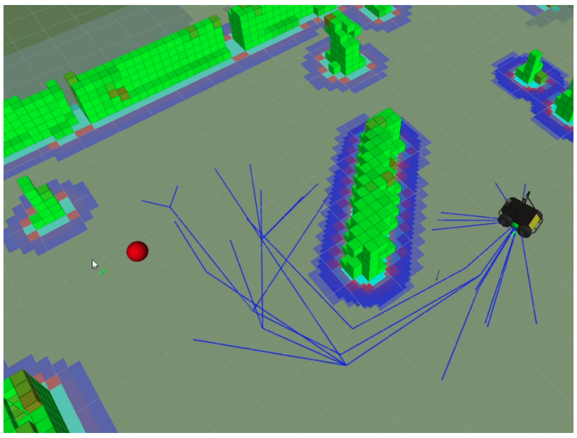


Fig. 3.5 The ROS structure of the 2D `move_base` Navigation Stack nodes and topics. Image taken from the ROS Wiki http://wiki.ros.org/navigation, licensed under the Creative Commons Attribution 3.0.

For the Specialization project, the default local planner was replaced with the `teb_local_planner` plugin for car-like robots [116]. The planner optimizes local trajectories with respect to execution time, obstacle separation and compliance with kinodynamic constraints using an underlying method called Timed-Elastic-Band (TEB). A global planner using the Informed RRT* algorithm presented in Section 2.2.3 was made for UGV path planning. The planner returns a collection of straight-line paths from the current UGV location to the inputed goal, if one such path is found before a timeout occurs. This is then used by the aforementioned local planner to to drive the robot along the path.

In this project, two major changes were made to the UGV path planner. First, a post processing step was added to optimize the paths. This involves iterating through the path nodes in a

[11] http://wiki.ros.org/navigation


(a) An informed-RRT* tree is grown from the UGV to the goal, visualized by a red sphere.

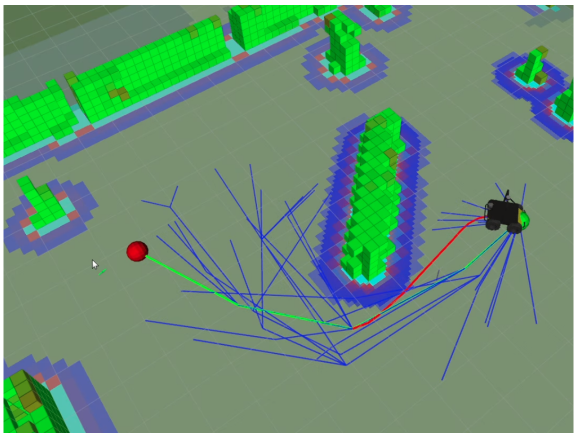

(b) A path to the goal is found by the global planner, visualized by a green path.

Fig. 3.6 Running the implemented RRT Global Planner on a UGV in a 3D environment populated by obstacles. The algorithm tree is visualized in blue. The chosen navigation goal is behind an obstacle, and an Informed-RRT* tree is grown to find a path. Obstacle-voxels displayed in real time by OctoMap. The `teb_local_planner` trajectory is visualized in red [1].

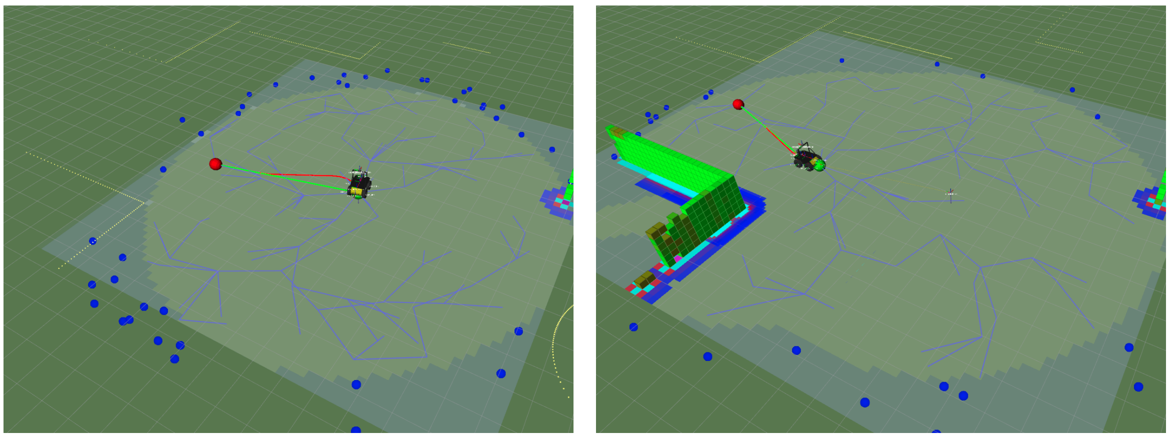
pairwise fashion and attempting to replace segments of the path with collision-free straight line segments. A dynamic collision detection and replanning system was also implemented which would periodically check the current global path for collisions, which may occur if obstacles are discovered on the path that were not present during planning.

For the majority of the scenarios covered in this project, the 3D LiDAR was removed such that all mapping would be done by the UAVs flying above the UGV. The 2D LiDAR and localization sensors are however still in use to provide local path planning and collision avoidance.

3.2.2 Frontier exploration

A system to conduct frontier exploration was made for the UGV system during the Specialization project. The system makes use of the RRT* algorithm to detect frontier points in the global map that is constructed during run-time. To achieve this, a tree is grown from the UGV position. When a potential tree vertex is made which lies within unmapped space, it is flagged as a frontier vertex and not added to the tree. The potential parent vertex of the frontier vertex, which will be in mapped free space since it is a part of the tree, is used as the navigation goal should its frontier vertex be chosen.

Several different methods of processing the results of the frontier detecting RRT algorithm were experimented with, both during the Specialization project and in this project work. The choice of which frontier vertex to navigate to can be determined simply by the vertices distance to the UGV position, either the straight line distance or the distance recorded as the cost in the RRT* algorithm. One could also factor in the orientation of the UGV, taking into account the time it takes for the platform to turn to face frontiers, or the density of frontier points.


(a) The closest frontier point is chosen and its parent node is selected as the UGV navigation waypoint.

(b) Upon reaching the frontier, the search is repeated and a new frontier is selected.

Fig. 3.7 Running the implemented Frontier RRT exploration planner on a UGV in a 3D environment populated by obstacles. The tree i visualized in blue, while frontier nodes are visualized as blue spheres in unmapped space [1]

In this project, a major change was made to the frontier exploration algorithm such that the tree was not discarded after each run, only the frontier vertices. This made the use of euclidean distances instead of RRT* costs necessary for deciding which frontier to explore next. It did however lead to a big speedup in frontier detection time, especially for frontier points far away from the current UGV position.

3.3 UAV systems

The criteria of ROS and Gazebo support as well as ease of transfer to outdoor applications on a real quadrotor UAV were central when deciding on which UAV systems to expand on for the Specialization project. Due to its popularity and support in the autonomous vehicle community, the PX4 autopilot system was chosen as the base for the UAV simulation and systems. An included demo of the Iris quadrotor equipped with a forward facing depth camera was chosen and heavily modified during the Specialization project to allow multi-robot simulation.

3.3.1 Autopilot software and interface

The choice of using the PX4 autopilot system informed the rest of the decisions regarding UAV software[12]. The open-source autopilot system consists of a flight stack for state estimation and control of multiple types of vehicle platforms, as well as general robotics middleware for communication, hardware integration and support for autonomous behaviour. The system is designed to run on a physical or simulated Flight Controller Unit (FCU), which handles motor control and sensor data from a physical or simulated drone system respectively.

External connectivity is provided using the MAVLink communication protocol[13], which is in turn used by the `mavros` package to provide an interface to ROS. The `mavros` package allows flight control data, such as the UAV status, current pose and position goal, to be exposed as ROS topics and services. The package also allows the UAV to be given pose- and body velocity goals which the flight controller software translates to motor commands. More intelligent control can be achieved using the built in Obstacle Avoidance (OA) software, which is an advanced feature that allows the UAV to drive around obstacles to reach given goals.

The PX4 system includes Gazebo models for many different platforms, such as quadrotor UAVs, ground vehicles and underwater vehicles. This includes simulation of the FCU which the autopilot system runs on. The system also simulates UAV sensors like a GNSS-receiver, IMU, barometer and altitude LiDAR, which are fused by the FCU's EKF. Range sensors can also be added, and as mentioned the project makes use of an Iris quadrotor model with a $64x48$ pixel RGB-Depth camera running at a frequency of 20 Hz with a maximum range of 8 meters.

In the Specialization project, the PX4 software for OA was used as a local planner for the UAV. However, several issues with the planner were identified, leading to it being replaced for this project. For normal operations it was determined that intelligent control for OA was not necessary since the planned paths would always be relatively short straight-line paths in known free space. Therefore, the end of this path line segments can be used directly as the position setpoint for the flight controller software. For longer navigation tasks that may need

[12]https://dev.px4.io/v1.10/en/concept/architecture.html
[13]https://mavlink.io/en/

obstacle avoidance, such as during the UAV recovery behaviour presented in section 4.1, the navigation stack used for the UGV was adapted for the UAV software.

3.3.2 View-based exploration

For the directional camera sensors on the UAV, orientation had to be taken into account when performing IPP for the individual UAVs. A system was developed using the RRT* algorithm to sample configurations and calculate their information gain metric at each tree vertex. Each UAV would have a tree grown from its current position and within free space[14].

Each tree segment, consisting of a series of connected vertices from the root-vertex to a tree-vertex (not necessarily a leaf-vertex), was recorded and assigned a utility as a function of the information gain at the vertex, the vertex rank and the utility of the parent vertex. The classic distance cost metric used in the RRT* algorithm was used for tree expansion, but not taken into account in the information utility function. The information utility $Utility$ was calculated as shown in 3.1, using the function $Gain(q)$ to calculate the information gain at a configuration q and $Rank(q)$ to get the vertex rank for configuration q in the tree.

$$Utility_{new} = Utility_{parent} + \frac{1}{Rank(q_{new})} Gain(q_{new}) \tag{3.1}$$

Information gain was calculated for each viewpoint using the OctoMap framework and a set number of equally spaced camera rays within the UAV FOV. For each viewpoint, rays were cast using OctoMaps raycast function in the predefined camera ray directions with a maximum range equal to the depth camera's maximum range. The information gain is then calculated as the ratio of unmapped voxels hit by the rays to the total number of rays cast. A horizontal FOV of 1.02974 and maximum range of 8 meters was used, with the rays cast in a 6 wide by 3 tall grid resulting in 18 rays.

After growing the tree for a set amount of time, the RRT* algorithm returns the tree and a list of the tree segments sorted in descending order by their utilities. The tree segment with the highest utility is selected, and the configuration represented by the first vertex in the segment is selected as the navigation goal for the UAV. After the first vertex is reached, the RRT* search is repeated, with the remainder of the previously used tree-segment included as an

[14]Collision checks were performed along the whole tree to ensure that each straight line segment of the tree would be collision free for a UAV following it. Configurations were also not allowed to lie within unmapped space.

initial tree with newly calculated costs and information utility.

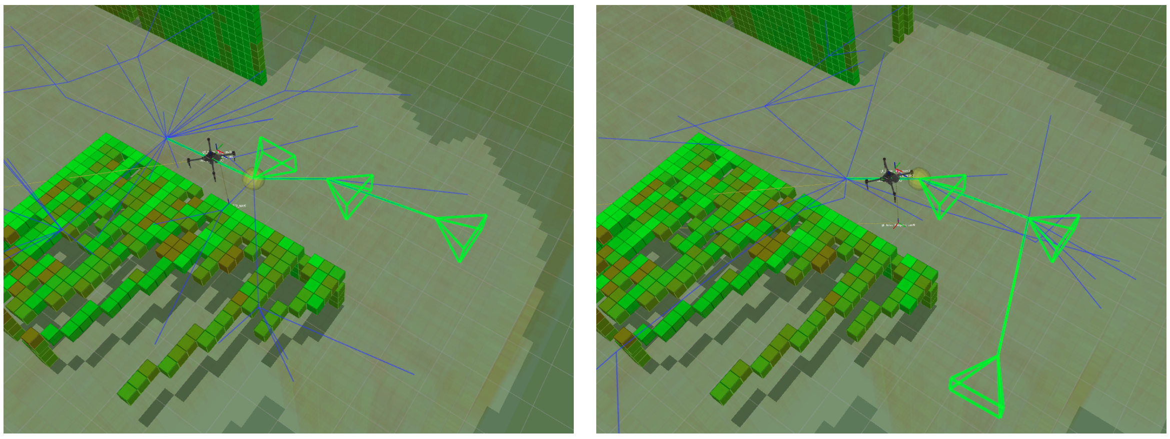

(a) View-sequence 1 (b) View-sequence 2

Fig. 3.8 Running the implemented View-based RRT exploration planner on a UAV in a 3D environment populated by obstacles. The search tree is visualized in blue, with the selected path visualized in green. Pyramids on the path visualize the FOVs at those particular locations [1]. (a) After running the RRT algorithm to sample view points, a sequence of views is selected and the first view is chosen as the navigation goal of the UAV. (b) After reaching the selected view point, the RRT search is repeated with the remainder of the previously chosen path is resampled.

The view-based RRT algorithm is shown in practice on a UAV in Figure 3.8. Pseudo-code for how the view-based RRT algorithm modifies the EXTEND function of the classical RRT algorithm shown in Algorithm 2 is shown in Algorithm 6.

For this project work, the View-based exploration planner plays a central role in the Collaborative exploration planner presented in Section 4.2. Here, the results of running the View-based exploration planner on multiple UAVs are processed to ensure collaborative exploration and communicaiton constraint satisfaction.

3.4 Conclusion

This chapter presented software frameworks and systems which make up the foundation that this project work is built upon. The general software frameworks for robotic systems and simulation were first covered. Following this, the UGV systems were presented, including the global path planner and frontier exploration planner implemented in the Specialization project. Finally, UAV systems for controlling the simulated UAVs were presented. Included in these systems was the View-based exploration planner, which will be expanded upon in Chapter 4.

Chapter 4

Design and implementation

This chapter presents the design and implementation of the systems that were developed for solving the problem presented in Section 1.2. As mentioned in Section 1.3, implementing these systems constitutes part of the contribution of this project work. Background material from Chapter 2 and Chapter 3 will be referenced when motivating design decisions in the implementation of the systems. Furthermore, results from testing the systems influenced some design decisions, as well as the creation of the systems presented in Section 4.4. These results will be presented later in Chapter 5. Several system parameters will be presented in this chapter. The default values of these can be found in Chapter B.

A simplified overview of the system architecture is presented in Figure 1.1. The main project systems can be grouped in a `MIPP Planner` system as presented in Figure 4.1. The system takes input from the UGV `Path planner` that is presented in Section 3.2.1, and uses the immediate UGV navigation plan to create a plan for the UAVs. Depending on the specific UAV planner mode that is in use, each `UAV Planner` subsystem instance within the `MIPP Planner` will formulate a goal for their assigned UAV. This goal will be sent to the relevant UAV `Path planner`. This plan can be aborted should a recovery maneuver be needed, as is the case when a loss of communication is detected.

This chapter begins by presenting the overall system structure in greater detail. The implementation will not be presented in detail, however some attention is given to ROS-related aspects that influence the structure of the system. The design of specific planners that handle multi-robot planning for the UAVs around the UGV will then be presented. First, the Collaborative exploration planner is presented, which extends the functionality of the View-based exploration planner presented in Section 3.3.2 to the multi-UAVs case. Next, the Simple-

and Sampling-based formation planners are presented. Finally, Combined planners which combine elements of the aforementioned planners are presented.

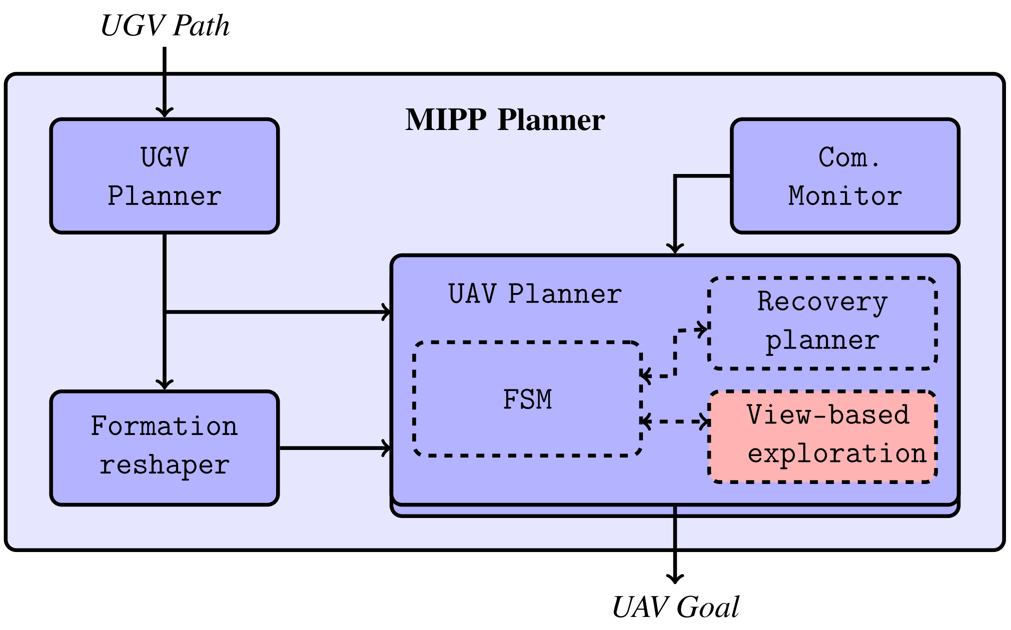


Fig. 4.1 Overview of the subsystems that make up the `MIPP Planner` system. Taken from Figure 1.1.

4.1 System structure

The main task of the `MIPP Planner` system is to coordinate UAV movement around the UGV during operation. UGV operation is initiated by giving the UGV `Path planner` a navigation goal. This goal can either be given directly by the user or, as is done in this project, from other systems. An example of one such system is the Frontier exploration planner presented in Section 3.2.2. Once the `Path planner` is given a goal, the Informed-RRT* global planner creates a global path from the UGV to the goal. This path is then published, and navigation begins.

The `UGV Planner` subsystem of the `MIPP Planner` receives the global path plan that is published when UGV navigations starts. This plan comes in the form of a series of waypoints with varying spacing, with the first waypoint being the initial UGV location and the final waypoint being the navigation goal. The path is continually processed by the `UAV Planner` subsystem, which creates a number of local navigation waypoints on the immediate UGV path starting from the current UGV location.

The number and spacing of these local navigation waypoints are defined by the parameters n_{ugv_wp} and l_{ugv_wp} respectively. These waypoints are visualized for a UGV in navigation in Figure 4.2, with $n_{ugv\_wp}= 4$ and $l_{ugv\_wp}= l$. In the special case that the remaining UGV path is shorter than the distance $n_{ugv\_wp} \cdot l_{ugv\_wp}$, the final global path waypoint (the navigation goal) is chosen as the final local waypoint such that the number of local waypoints may be less than n_{ugv_wp}. For a stationary UGV, the current UGV position is the only local navigation waypoint. The parameter l_{ugv_wp} is chosen relative to the set maximum UGV speed v_{ugv}, such that a certain amount of time t_{ugv_wp} can elapse before the UGV reaches its next local waypoint.

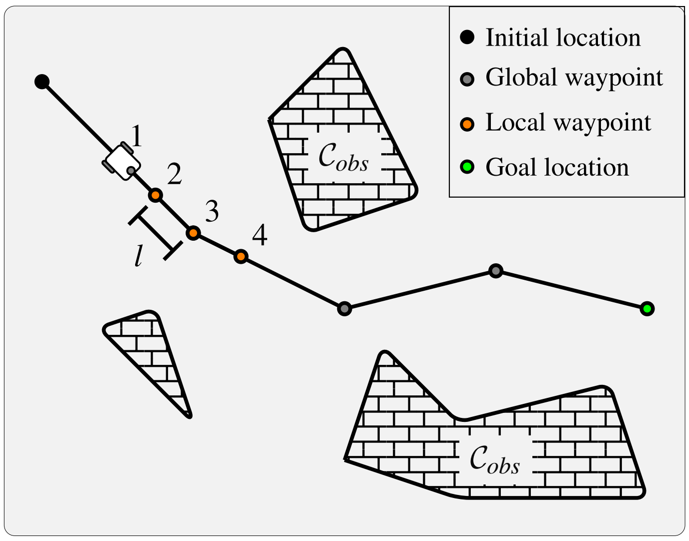


Fig. 4.2 Local UGV navigation waypoints are created by the `UGV Planner` subsystem of the `MIPP Planner`. Waypoints are placed along the global path, starting from the current UGV location.

The local navigation waypoints are passed to the `UAV Planner` subsystem(s) of the `MIPP Planner`, with a `UAV Planner` subsystems being initialized and assigned to each UAVs in the system. Central for each individual `UAV Planner` is a Finite-State Machine (FSM) which manages the current planner state of its assigned UAV. The FSM is visualized in Figure 4.3.

The `INIT` state is responsible for initializing the `UAV Planner` variables after the FSM has been started. It also blocks until other systems are ready, waiting for both the UAV to take off and for communication with the UGV to be established. When this is complete, the FSM

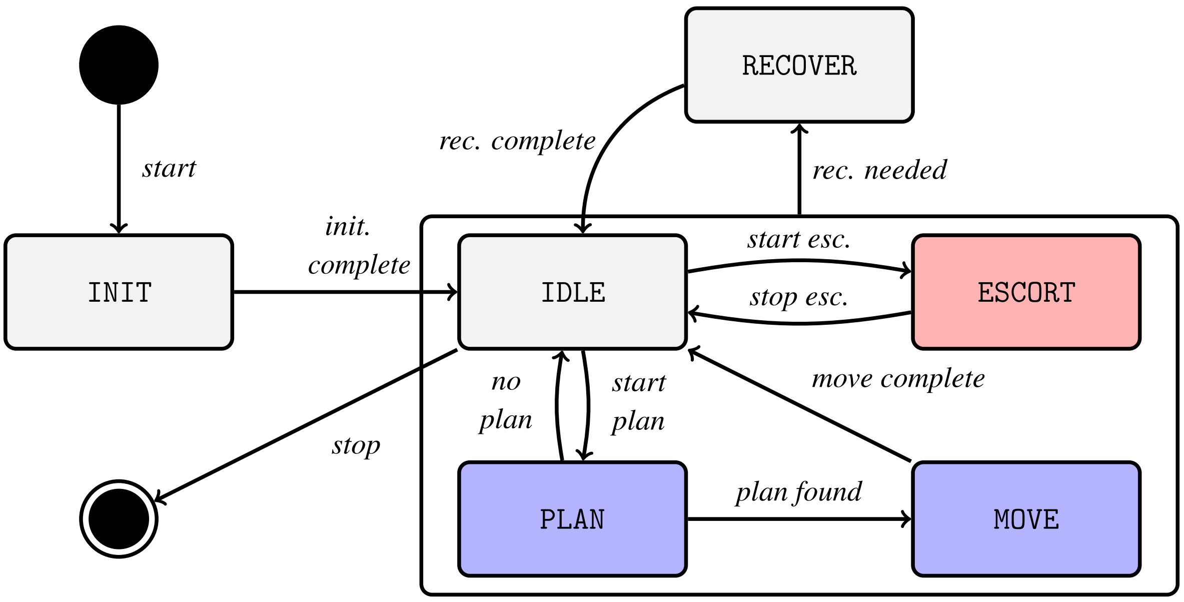


Fig. 4.3 The FSM architecture for coordinating UAV planning in the `MIPP Planner` system. Blue states are used in the Collaborative exploration planner. Red states are used in the Formation-based planner.

transitions to the `IDLE` state and enters the main operational loop.

The FSM-transitions that occur within the main operational loop depends on the selected UAV planner mode that is in use. While in the `IDLE` state, the UAV maintains a stable hover. The next transition may either be to the `PLAN` state or the `ESCORT` state, depending on if the Collaborative exploration planner or the Formation-based planner are in use respectively. The functionality of these planners are covered in Section 4.2 and Section 4.3.

Normal operation may be interrupted by a call for recovery, in which case the FSM transitions to the `RECOVER` state. A transition to the `RECOVER` state may be triggered internally by the `UAV Planner` or by an external call. When entering the `RECOVER` state all current plans are aborted and cleared. The current UGV location is used as a recovery navigation goal for the UAV and is passed to the UAV `Path planner`. A path to the recovery navigation goal is created using a modified variant of the UGV Navigation Stack presented in Section 3.2.1. This differs from the normal operation of the UAV `Path planner` presented in Section 3.3.1, where goals are assumed to be within a short distance in free space such that obstacle avoidance isn't necessary. Once the UAV in recovery reaches the area above its recovery navigation goal, the UAV FSM transitions back to the `IDLE` state.

The `Communication monitor` system is a separate ROS Node which monitors the current satisfaction of communication constraints between the UGV and individual UAVs. Possible communication constraints are presented in Section 2.3.4. In this project, both the range and Line-of-Sight (LoS) constraints are enforced. The range constraint is continuously checked by monitoring the ground truth positions of the vehicles in the simulator and comparing the euclidean distance in three dimensions with the maximum communication range l_{com}. Checking whether or not a UAV has LoS with the UGV is done using the OctoMap `castRay()` function on the current map built by the system. If either constraint is broken for a period longer than the communication timeout t_{com}, communication to the UAVis considered broken and a recovery call is made for the UAV to recover.

The `MIPP Planner` system is implemented as a number of ROS nodes which communicate through Topics, Services and Actions. The `UGV Planner`, `UAV Planner` and `Formation reshaper` subsystems are implemented as one Node, where each subsystem is controlled by asynchronous ROS timer functions running at set frequencies of f_{ugv_pl}, f_{uav_pl} and f_{uav_sfr} respectively. The `UGV Planner` subscribes to the UGV global planner path and odometry Topics. When the `UGV Planner` timer function is called, local navigation waypoints are made using the latest UGV odometry. At each `UAV Planner` interval, the UAV FSM is checked for transition conditions and the state is updated. This includes whether or not a recovery has been called, either by an internal system or by the `Communication monitor`. The `Formation reshaper` subsystem will be presented in further detail in Section 4.3. As mentioned previously, the `Communication monitor` is implemented as a separate node. Instances of the `View-based exploration` planner also run as independent Nodes assigned to each UAV, and will be presented in further detail in Section 4.2.

4.2 Collaborative exploration planner

The Collaborative exploration planner extends the functionality of the individual View-based exploration planners implemented for the UAVs in the specialization project. As presented in Section 3.3.2, the planner uses the RRT* algorithm to sample sequences of views and estimates the information gain related to these views. A maximum ray and neighbor distance l_{uav_rrt} is used by the RRT* algorithm to create the tree. Information gain is estimated by casting a set amount of rays within the FOV of the sampled configuration and returning the ratio of rays which hit unmapped voxels to the total amount of rays cast. Samples are taken uniformly within a circle with a circle center $p_{rrt\_samp,i}$ and a given radius r_{rrt_samp}. For the specialization project, $p_{rrt\_samp,i}$ was chosen to be the current position of UAV$_i$. The planner is given a maximum running time t_{rrt_samp}, after which the sampled view-sequences are returned.

4.2.1 Modified View-based exploration planner

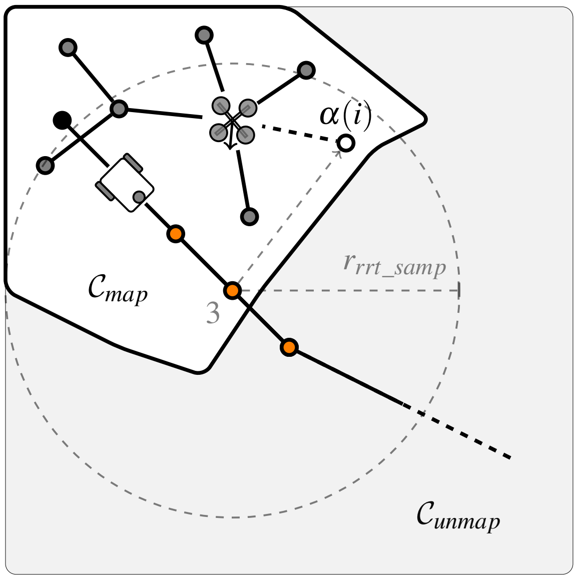
The View-based exploration planner is modified for this project work to act as an Action server. Action calls can be made to the server to initiate the View-based planner, with the ability to specify the sampling center $p_{rrt\_samp,i}$, radius r_{rrt_samp} and time t_{rrt_samp} as part of the call. A call to the Action server is made when a UAV FSM transitions the `PLAN` state. Each UAV has an instance of the View-based planner running, which run independently of other UAV systems and UAVs.

The View-based planner Action server is also modified to accept a list of sampling centers $p_{rrt\_samp,i}$, selecting one sampling center with a uniform probability when creating a sample. The sampling centers passed to the View-based planner Action server are the local navigation waypoints made by the `UGV Planner` subsystem. The sampling radius r_{rrt_samp} is chosen to be equal to the maximum communication range l_{com}. By using these parameters for the planner, all sampled views are ensured to be within communication range of at least one local UGV navigation waypoint.

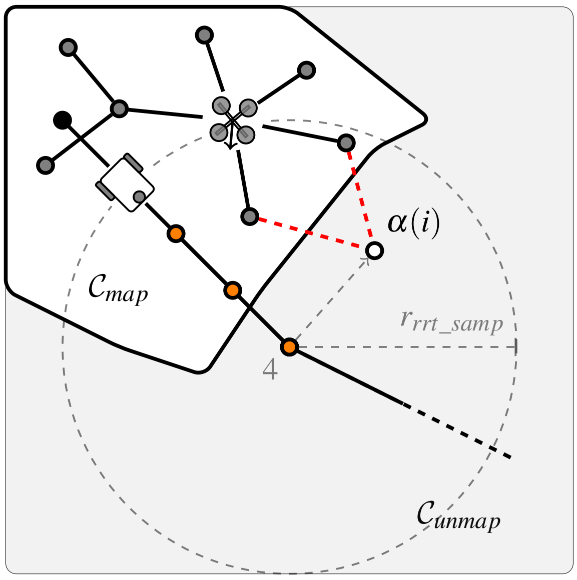
The RRT* ray and neighbor distance parameter l_{uav_rrt} is chosen relative to the set maximum UAV velocity v_{uav} and the local UGV navigation waypoint time t_{ugv_wp}. By ensuring that l_{uav_rrt} is sufficiently less than the product of v_{uav} and t_{ugv_wp}, the UAV is able to reach the first vertex of the returned tree before the UGV reaches the next UGV waypoint. The maximum rank rnk_{rrt_v} of any vertex in the RRT* algorithm is chosen to be less than n_{ugv_wp}. A visualization of the modified View-based exploration sampling is shown in Figure 4.4.

Collision checking is performed for sampled configurations using the collision checking mechanism described in Section 3.1.2. For the purpose of collision checking, unmapped space is considered occupied. To avoid collision between individual UAVs in the system, each UAV is assigned a distinct height/altitude at which it must plan and move. This altitude $h_{uav,i}$ is determined using Equation (4.1) with a minimum UAV operational altitude h_{uav_min} and altitude interval h_{uav_int} for a UAV with ID i.

$$h_{uav,i} = h_{uav\_min} + i \cdot h_{uav\_int} \tag{4.1}$$



(a) Sample is not in collision.



(b) Sample is in collision.

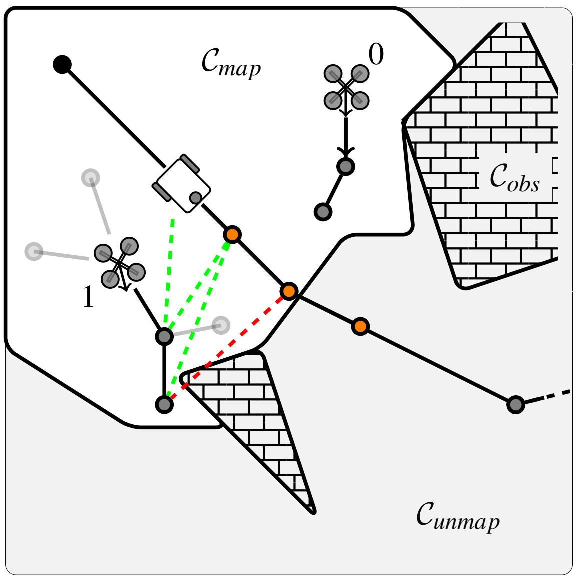
Fig. 4.4 The View-based sampler is modified to sample configurations uniformly within a circle of radius l_{com}. The sampling circle center is chosen randomly among the list of local navigation waypoints constructed by the `UGV Planner` subsystem. (a) A configuration $\alpha(i)$ is sampled with waypoint nr. 3 as the sampling center. The configuration is collision-free and is connected to the tree. (b) A configuration $\alpha(i)$ is sampled with waypoint nr. 4 as the sampling center. The configuration is in collision (unmapped space) and not connected to the tree.

Once the sampling time t_{rrt_samp} has elapsed, the planner Action server returns a list of view-sequences and the total sequence information gain, as well as the information gain tied to each view. The view-sequences are then processed further, considering both communication constraints to the UGV and the plans of other UAVs in the system.

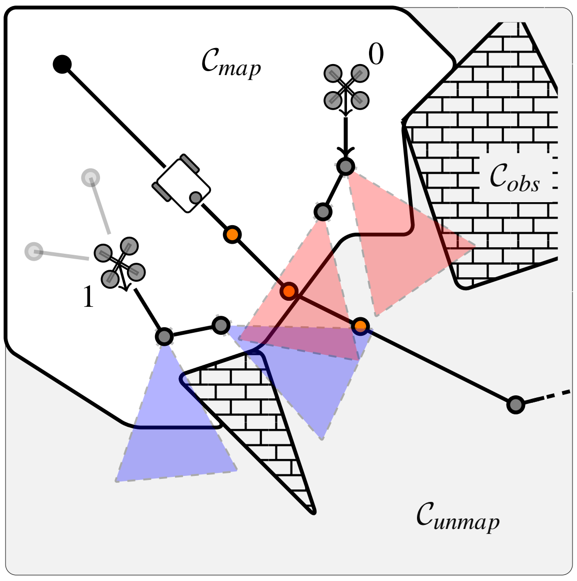
4.2.2 Communication constraints and coverage overlap

Communication constraints are check relative to the local UGV navigation waypoints. Individual view configurations in the view-sequences are checked against a pair of waypoints, with the rank of the view determining which waypoints it is checked against. The rank is determined in ascending order, with the first view in the sequence being assigned a rank of 1. For a view of rank n, both range and LoS communication constraints are checked against waypoints n and $n+1$. If a view configuration doesn't satisfy the communication constraints, its view-sequence is discarded. This process is visualized in Figure 4.5a.

If a view-sequence satisfies the communication constraints, its information gains are recalculated to take sensor coverage overlap into account. Sensor coverage overlap is considered both internally with regard to the other views within the view-sequence and externally to the chosen plans of other UAVs. Internal overlap is considered in ascending order of rank, such that the first view will have no internal overlap, the second view may have internal overlap with the first, etc. External overlap is checked for each view by iterating over the views contained in the plans of other UAVs. These plans are recorded once a UAV finishes planning and transitions from the `PLAN` to the `MOVE` state. A visualization of possible sensor overlap between UAV plans is shown in Figure 4.5b.



(a) Communication constraint satisfaction.



(b) Sensor coverage overlap.

Fig. 4.5 The results of the View-based exploration planner are processed to take communication constraints and collaboration into account. In this example, UAV 0 already has a plan that is being executed, while UAV 1 is processing the results of the View-based exploration planner. (a) A view-sequence is checked for communication constraint satisfaction. The last view in the sequence does not satisfy the constraints (no LoS), and the sequence is discarded. (b) Information gains for views in the sequence are recalculated to take sensor coverage overlap into account. The last view's coverage overlaps with a view in another UAV's plan and its gain is reduced.

To calculate sensor coverage overlap, the UAV sensor FOV is approximated as a circle on the two dimensional ground plane. A conservative approximation is used, such that the coverage circle is larger than the actual sensor coverage. The circular coverage is defined by the distance from the UAV sensor to the circle center l_{uav_cov} and the radius of the circular coverage r_{uav_cov}. The calculation is simplified further by restricting all sensor coverage circles to have the same radius. These simplifications allow the calculation of sensor coverage overlap as shown in Equation (4.2) [117]. Here, rat_{ovr} is the ratio between the area of overlap A_{ovr} and the total area of the circle A_{tot}, C is a circle that has an area overlapping another circle C^*, r is the common circle radius and d is the distance between the circle centers of C and C^*. This is visualized in Figure 4.6.

$$rat_{ovr}(C,C^*) = \frac{A_{ovr}}{A_{tot}} = \begin{cases} 0, & \text{for } d \geq 2r, \\ 1, & \text{for } d = 0, \\ \frac{2}{\pi}\cos^{-1}(\frac{d}{2r}) - \frac{d}{2\pi r^2}\sqrt{4r^2 - d^2}, & \text{for } 0 < d < 2r. \end{cases} \tag{4.2}$$

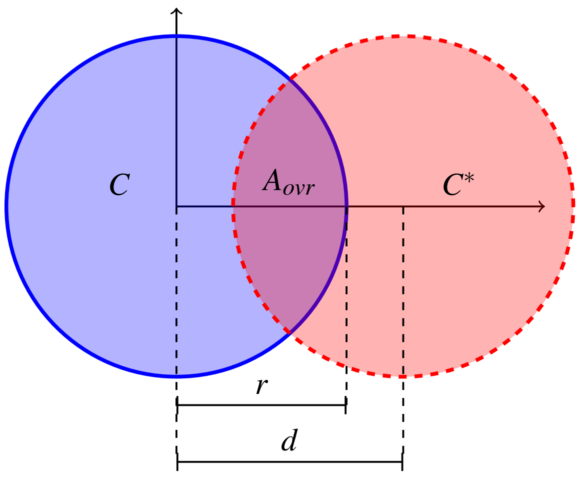


Fig. 4.6 The intersection area of two circles with the same radius r, with a distance d between the two circle centers.

4.2.3 View-sequence utility calculation

Given a view configuration q and corresponding sensor coverage circle $C(q)$, other sensor coverage circles (internal and external) C_{other}, the ratio rat_{unmp} of rays cast within the FOV which intersect with unmapped voxels to the total number of rays cast and the ratio of overlap $rat_{ovr}(C,C^*)$ between two circles C and C^*, the information gain $Info(q)$ for a view configuration is recalculated as shown in Equation (4.3). To account for the possibility that multiple coverages may overlap with the sensor coverage circle of the configuration in question, the function sums the overlap ratios individually and caps this sum as to not allow negative information gain values. This is only an approximate method which doesn't account for overlap between more than two circles in the same area, and isn't the correct way to calculate overlap between more than two circles.

$$Info(q) = rat_{unmp} \cdot \left(1 - \min\left\{\sum\nolimits_{C^* \in C_{other}} rat_{ovr}\left(C(q), C^*\right),\ 1\right\}\right). \tag{4.3}$$

Unlike in the specialization project, the distance travelled by the UAV to get to a view configuration is also considered when selecting the UAV plan. Both euclidean distance d and angular (yaw) distance ψ are considered. The latter is transformed to a standardized unit for angular distance ψ_u that is defined by the author. The former is considered in meters. When considering sequences of view configurations, both distance metrics are considered relative to the previous view configuration. For the case of the first view configuration, the distance metrics are considered relative to the current UAV position and orientation.

For a UGV in movement, a method to discount UAV movement for communication constraint maintenance was implemented. To encourage UAV movement in the direction of the moving UGV, a certain amount of "free movement" d_{free} is subtracted from the euclidean distance considered in the overall utility function. This is achieved by iterating through the local UGV navigation waypoints and the corresponding (by rank) view configurations and performing a vector subtraction on the corresponding UGV and UAV path segments. The discount is also capped to zero from below as to not allow a negative distance discount, which would correspond to a penalty for UAV plans which move opposite to the UGV plans. The discount for each view-sequence path segment is also naturally capped from above by the maximum discount of l_{ugv_wp}.

The utility of a view-sequence can be determined with the utility function shown in Equation (4.4). Here, Seq is a non-empty sequence of view configurations q. $Info(q)$ is the information gain with coverage overlap taken into account as presented in Equation (4.3). The distance d is the sum of euclidean distances travelled between each consecutive view configuration in the sequence, with d_{free} being the total free distance described above. The total angular distance travelled along the sequence ψ is transformed to a standardized unit ψ_u defined by the author. The cost parameters c_{info}, c_{euc} and c_{ang} weight the information gain, euclidean distance and angular distance of the sequence, with $c_{info} \geq 0$, $c_{euc} \leq 0$ and $c_{ang} \leq 0$.

$$Utility(Seq) = c_{info} \cdot \sum\nolimits_{q \in Seq} Info(q) + c_{euc} \cdot \left(d - d_{free}\right) + c_{ang} \cdot \left(\frac{\psi}{\psi_u}\right). \tag{4.4}$$

Once the utility of all communication constrained view-sequences has been calculated, the Collaborative exploration planner selects the sequence with the highest utility as its plan. The sensor coverages of the view-sequence are recorded so that other UAVs can consider coverage overlap with the selected plan. Finally, the first configuration in the view-sequence is selected as the navigation goal of the UAV and the UAV FSM transitions to the `MOVE` state.

When a UAV reaches its navigation goal, the UAV FSM transitions to the `IDLE` state. If the `MIPP Planner` system is still running and the Collaborative exploration planner mode is still in use, the FSM immediately transitions to the `PLAN` state. The planning sequence is then repeated, with any remaining views in the previously selected view-sequence being added to the planning action call. These views are resampled by the planner to enable receding horizon planning.

4.3 Formation-based planner

A Formation-based planner was created for UAV coordination around the UGV as an alternative to the Collaborative exploration planner. Two variants were made: a Simple "formation bank" planner which makes use of a bank of user specified formations, and a Sampling-based Formation Reshaping (SFR) planner which samples new formation configurations. Common for both methods is the way UAV planning is done relative to a coordinate frame fixed on the UGV (which might not be stationary).

A distinction is made between a UAV's escort pose and its formation pose. The formation pose refers to the pose of the UAV in formation relative to the UGV. In this coordinate frame, the UGV is at the origin with an orientation of zero. Formation poses are either kept in a formation bank in the case of the Simple formation bank planner or continuously sampled in the case of the Sampling-based formation reshaper. The escort pose refers to the formation pose transformed into a global inertial frame. This transformation includes the translation and rotation of the formation pose by the current pose of the UGV. In the case of UAV path planning, the escort pose is the navigational goal sent to the UAV `Path planner` system. This navigation goal is published continuously at a high frequency f_{uav_pl} such that the UAV follows the UGV closely.

The Formation-based planner is activated when the UAV FSM enters the `ESCORT` state. Unlike the Collaborative exploration planner covered earlier, the FSM does not transition out of this state during normal operation. While in the `ESCORT` state, the formation is updated at each `UAV Planner` interval with a frequency of f_{uav_pl}. This includes transforming the current formation pose and sending the appropriate navigation goal to the UAV. For the Sampling-based formation reshaper, an additional loop is run at the frequency f_{uav_sfr} which updates the formation through sampling.

4.3.1 Simple formation bank planning

The first variation of the Formation-based planner makes use of a set, or bank, of formation poses which the UAVs can vary between individually. These poses are made ahead of time and consist of the series of poses ranging from a pose close to the UGV to a pose further form the UGV. The far poses are preferred, as they increase the area that is swept by the UAV sensors as the UGV moves. Closer poses are used in case the further poses are not viable, for example if they are in collision, with the closest pose being assumed collision free. The orientation of the poses are also varied, with the orientation facing forwards when in the far pose and to the side in the close pose. A visualization of the formation bank poses is presented in Figure 4.7.

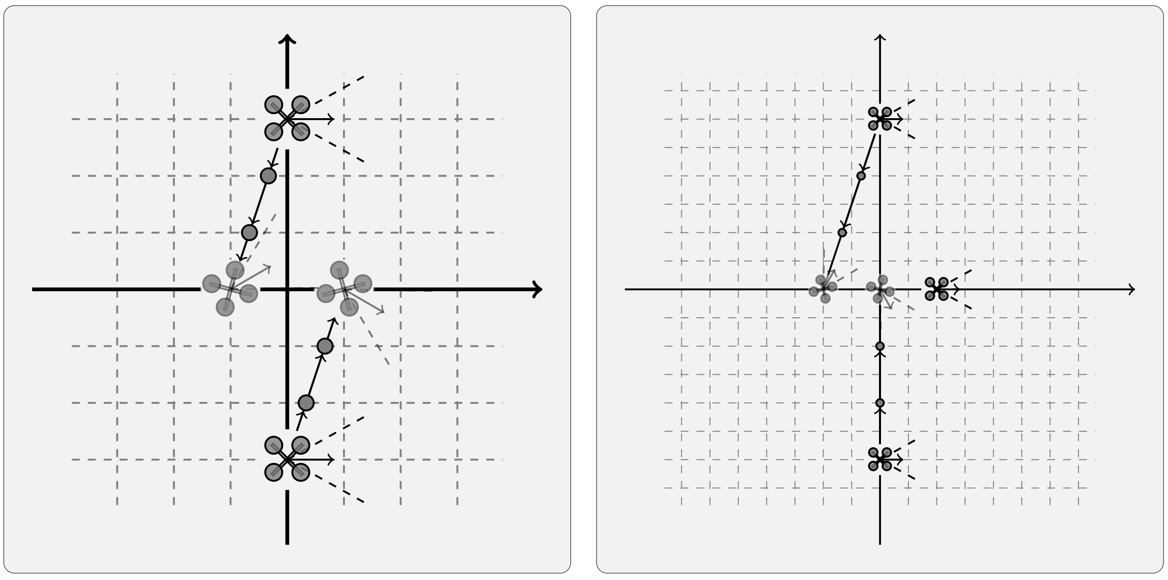

(a) Formation bank for 2 UAVs. (b) Formation bank for 3 UAVs.

Fig. 4.7 The Simple formation bank planner makes use of a bank of predefined formation poses that the UAVs may vary between individually. These include a close pose, a far pose and a linear interpolation between these poses.

Each formation bank is parameterized by the close formation pose p_{close}, a far formation pose p_{far} and the amount of poses in the bank n_{form_bank}. Usually, $n_{form\_bank} > 2$, such that linear interpolation is used to make the remaining poses between p_{close} and p_{far}. This includes the yaw values. When these values are given, the planner calculates all the poses beforehand and saves them for later use. The number of UAVs in the system also affects the poses in the pose bank for each UAV. The formation bank pose parameters used for this project are presented in Chapter D.

Collision avoidance for a formation pose is check by propagating the formation pose forwards along the UGV path and checking the resulting path for collision. An escort pose is calculated for the UAV for each local UGV navigation waypoint. The projected orientation of the UGV at each waypoint is determined in relationship to the previous waypoint, with the first UGV waypoint assuming the current UGV orientation. A path is then created for each UAV which joins these propagated formation poses. This is demonstrated in Figure 4.8a.

The same collision checking mechanism used for the Collaborative exploration planner is used to check for collisions on the escort path. Any potential escort pose is checked to both not be in collision and to not be in unmapped space. The path from the current escort pose to a potential escort pose is also checked. For the propagated escort poses on the escort path, only collision with occupied space is considered and a path is allowed to have portions in unmapped space. This assumes that all formation poses are facing in a direction which allows the front-facing UAV sensor to map the area ahead, such that any obstacles on the escort path will be detected.

During operation, the planner will attempt to increase the width of the formation by iterating through the poses in the formation bank and checking them for collision. The formation bank is ordered such that p_{close} is checked first and p_{far} last, with all poses being iterated through regardless of the current formation pose. If a collision is detected on one of the propagated formation paths, the preceding formation pose will be chosen as the new formation pose. As the UGV moves through the environment and the UAV sensors explore more area, a newly detected obstacle may lead to the current formation pose being in collision. This results in the formation bank iteration selecting a new, collision free formation pose and sending the corresponding escort pose as the navigational goal for the UAV `Path planner`. A visualization of this is shown in Figure 4.8b.

Communication constraints are check along with the collision check for each potential pose and propagated path in the formation bank. Communication range can be checked simply by checking the euclidean distance from the origin to the formation pose, as this corresponds to

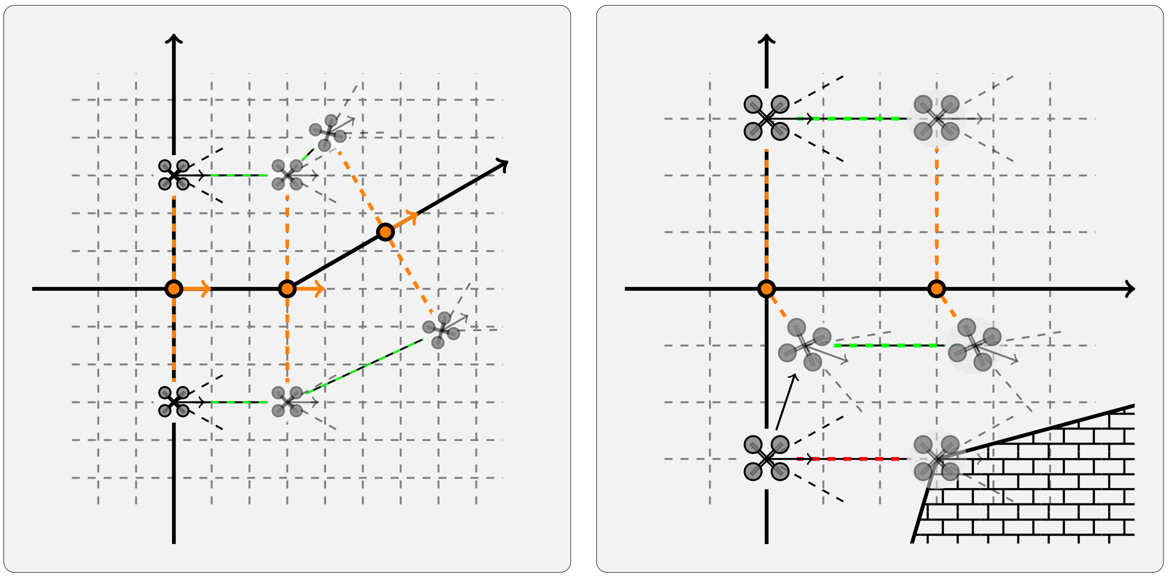

(a) Formation along a non-straight path. (b) Formation collision avoidance.

Fig. 4.8 The Simple formation bank planner modifies and selects formation poses for the UAVs depending on the projected local navigation waypoints of the UGV. (a) The formation poses maintain their orientation relative to the local navigation waypoint, leading to propagated escort paths which curve along with the local navigation path. (b)

the distance from the UGV to the UAV when the UAV is in formation. LoS constraint checks are performed pairwise between the propagated escort path poses and the corresponding local UGV navigation waypoints.

4.3.2 Sampling-based Formation Reshaping

The second variation of the Formation-based planner is the Sampling-based Formation Reshaping (SFR) planner. The SFR planner continuously samples potential formation poses and selects the best formation, both on the basis of collision avoidance and communication constraint but also with regards to formation utility. Unlike the Simple formation bank planner, the formation bank is not specified beforehand. Samples are however taken within certain limits of distance and orientation.

The initial formation pose of each UAV in the SFR planner is their respective p_{close} pose taken from the Simple formation bank planner. After initializing, new formation poses are sampled. These poses are sampled within a radius r_{sfr_samp} of the current formation poses and have an orientation which is uniformly sampled in a range with a maximum deviation

ψ_{sfr_samp} from the UGV orientation. The orientation deviation ψ_{sfr_samp} is increased when the UGV approaches its goal. One pose is sampled for all UAVs in the system simultaneously as to create a formation sample.

When a formation sample is taken, it is checked for collision avoidance and communication constraint satisfaction. This is done in the same way as formations were checked in the Simple formation bank planner, including propagating the resulting escort poses into escort paths. Formations which are found to be in collision or which violate the communication constraints are discarded, and a new sample formation is taken.

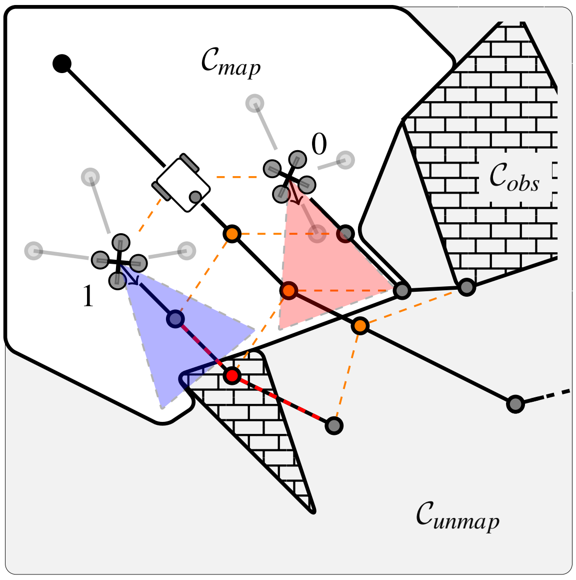
Formations which are viable, meaning they are not in collision and are communication constrained, are kept in a "viable formation bank". New formation samples are taken until the bank contains a number n_{sfr_bank} of viable formations. This viable formation bank is kept until the next iteration of the SFR planner, where the viability and utility of the formations are recalculated. Following this, only a number n_{sfr_keep} of the best viable formations are kept, and the rest are discarded. Sampling then resumes to fill the bank.

Formation utility is determined much in the same way as for the Collaborative exploration planner. For information gain, the propagated escort path is treated similarly to the view-sequence paths. Overlap is considered both internally and externally, with external overlap being checked sequentially in the ascending order of the UAV id. A combined information gain utility is then calculated which sums the information gain utilities of the individual UAVs in the formation. Euclidean and angular distances are also calculated from the combined distances the UAVs have to travel in the formation frame to get to the new formation from their current formation.

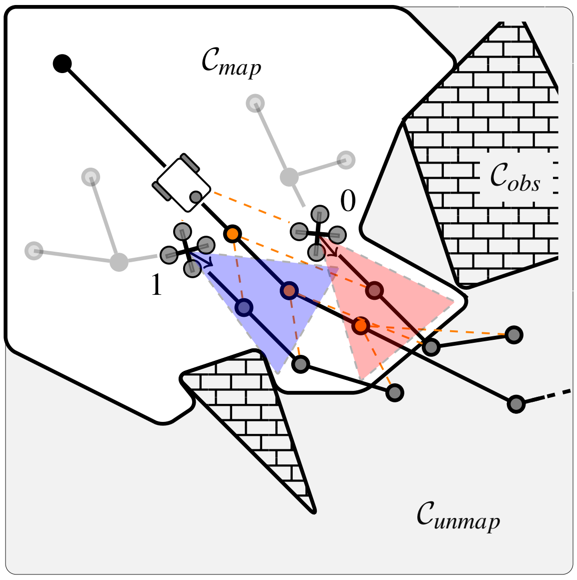
The utility for a formation is calculated as shown in Equation (4.5). The cost parameters c_{info}, c_{euc} and c_{ang} are reused from the Collaborative exploration planner to weight the information gain, euclidean distance and angular distance. For the information gain utility, view-sequences are constructed from the propagated escort paths and overlap is check sequentially. Unlike in Equation (4.4), the "free distance" discount has been removed, as the euclidean distance is not measured along the view-sequence but is measured in the moving formation frame. This distance is combined for all UAVs in the combined distance d_{comb}. Similarly, the combined angular distance ψ_{comb} is calculated in the formation frame.

$$Utility(Form) = c_{info} \cdot \sum_{Seq \in Form} \sum_{q \in Seq} Info(q) + c_{euc} \cdot d_{comb} + c_{ang} \cdot \left(\frac{\psi_{comb}}{\psi_u} \right). \quad (4.5)$$

After calculating the utility of each formation in the viable formation bank, the formation with the highest utility is selected as the new formation. This new formation is reported to the `UAV Planner` subsystem and is broadcast continuously to the UAV `Path planner` at a frequency of f_{uav_pl}. Iterations of the SFR planner are run at a frequency of f_{uav_sfr}, with the relation $f_{uav\_pl} > f_{uav\_sfr}$ being required to allow the UAV to assume its new formation before any new formations are sampled and selected. A visualization of the SFR planner is shown in Figure 4.9.



(a) Formation in collision.



(b) New formation assumed.

Fig. 4.9 The SFR planner continuously samples new formations which the UAVs may assume around a moving or stationary UGV. (a) A newly detected obstacle may make the current formation not viable due to a collision on the propagated escort path. (b) A new, collision free formation is assumed which also takes into account the utility of the new formation.

In situations where no viable formations can be sampled, a timeout $t_{sfr\_samp} < \frac{1}{f_{uav\_pl}}$ is enforced after which a viable formation must be selected. If the viable formation bank is empty when the timeout occurs, a recovery formation pose is selected which should be viable. This formation pose is the p_{close} formation pose which is used by the Simple formation-bank planner presented earlier. After assuming this formation, formation sampling is resumed.

4.4 Combined planners

Several planning systems which combine elements of the aforementioned planners were created for this project. The implementation of these Combined planners was motivated by some of the preliminary results presented in Chapter 5, specifically Section 5.3 and Section 5.4. These results identified that the different planners had strengths and weaknesses for different situations and scenarios.

4.4.1 Hybrid planner

The Hybrid planner is motivated by the observation that the planners performed differently in stationary- and path following scenarios. Specifically, the Collaborative exploration planner was observed to excel in scenarios where the UGV was either stationary or moving slowly, while not performing as well when UGV speed was increased. These results contrast the results and behaviour observed for the Formation-based planners, which are not as affected by UGV motion but which explore slowly in stationary scenarios.

The central idea of the Hybrid planner is to switch between the planning modes once the UGV approaches a navigation goal where the UGV will remain stationary for some time. Such stationary navigation goals appear in the Tour- and Frontier-exploration scenarios presented in section 5.5 and section 5.6 respectively.

During UGV operation that involves movement, such as following a path, the UAVs will be tasked to use the Simple formation bank planner. The distance to the next stationary navigation goal is continuously tracked by the planner. A hybrid switching distance l_{hybrid} is defined and checked against the distance to the next goal. Once the distance to the next goal goes below the threshold defined by l_{hybrid}, UAV planning switches to using the Collaborative exploration planner. This persists until the UGV receives a new navigation goal and begins moving, at which point the UAV planning switches back to the Simple formation bank planner.

4.4.2 Split planner

The Split planner is motivated by the observation that the Collaborative view-based planner would not sufficiently map the area directly in front of the UGV in movement, causing loss of LoS communication and recovery behaviour. This behaviour was not observed with the Formation-based planners, which were observed to move ahead of and map the area in front of the UGV.

The central idea of the Split planner is to allow the UAVs in a multi-UAV team to operate with different planners. In the case of two UAVs, one UAV would plan using the Collaborative exploration planner while the other would make use of the SFR planner. The assigned planners would be used throughout the entire UGV operation.

To make the Split planner collaborative, sensor coverage overlap must be considered between the different planning systems. As both the Collaborative exploration planner and the SFR planner make use of the same function to calculate coverage overlap and weight information gain taking overlap into account, collaboration could be ensured by sharing plans between the planners. This is straight forward for the Collaborative exploration planner, as it is already done during normal operation. For the SFR planner, a plan is created from the propagated escort path, which is updated and published with a frequency of f_{uav_sfr}.

4.5 Conclusion

This chapter described the design and implementation of multiple systems which were made in this project work. First, the general system architecture to enable CCMIPP planning and coordination of the robots was presented. Some attention was given to the system implementation ROS, specifically with regard to communication between processes. Following this, the Collaborative exploration planner was presented which extends the View-based exploration planner for the cooperative communication constrained case. Two variants of Formation-based planners were then presented: a Simple formation-bank planner making use of a predefined bank of formation poses and a Sampling-based Formation Reshaping (SFR) planner which samples new formation poses to switch to. Finally, two planners which combine aspects of the Collaborative exploration and Formation-based planners were presented.

Chapter 5

Simulation and results

This chapter presents the results gathered from running the implemented planners on a simulated robotics system in multiple scenarios. These results are a part of the contributions presented in Section 1.3, as they are used to demonstrate and compare the performance of the implemented systems. The robotics and simulation framework which these results were taken from are presented in Chapter 3, while the design and implementation of the planners that the results relate to is covered in Chapter 4.

Included in the presentation of the design and implementation is the definition of multiple parameters which modify the behaviour of the implemented systems. Some parameters had their values modified when gathering results to demonstrate characteristics of the systems. Whenever such a modification is made, the relevant parameter is referenced along with the new value and purpose of the change. Otherwise, all parameters can be assumed to have their default values. These can be found in Chapter B.

This chapter begins by presenting the performance metrics that are used to judge the system performance. The simulation environments will then be presented briefly. Following this, the four different scenarios which the planning systems were tested in will be presented, along with results from running the systems in these scenarios. These results will be accompanied with some discussion that is relevant to what the results show. However, the main discussion of the overall results will be presented in Chapter 6.

5.1 Performance metrics

Several performance metrics are considered when comparing the developed systems. As Communication Constrained Multi-robot Informative Path Planning (CCMIPP) systems are the focus of this project work, these metrics will mainly be tied to information gain and communication constraint satisfaction. Several metrics exist within the field of robot exploration which can be used for this purpose [118]. As mentioned in Section 1.2, the shift of focus from pure exploration towards efficient coordination of the UAVs around the UGV will also influence which metrics are used.

Total Information Gain (TIG) is defined as the area or volume which has been explored by the system. Since the implemented systems operate and map in three dimensional space, TIG is taken to mean the volume that is mapped by the system. This volume is calculated by multiplying the amount of voxels mapped $n_{octomap}(t)$ at time t by the voxel resolution $res_{octomap}$, which is done using functions from the OctoMap framework presented in Section 3.1.2. The function for calculating the TIG metric is shown in Equation (5.1), where the resulting volume will be in cubic meters [m^3].

$$TotalInfoGain(t) = n_{octomap}(t) \cdot (res_{octomap})^3. \tag{5.1}$$

Information Gain Efficiency (IGE) is defined as the TIG divided by some exploration cost. For the systems developed for this project, the most relevant cost is the distance travelled by the UAVs. As the TIG is defined with regards to the total volume explored up tot the current time t, the distance travelled that corresponds to the TIG metric is the sum of the distances travelled by all the UAVs up to the current time t. The function for calculating the IGE metric is shown in Equation (5.2), with the resulting unit being [m^3/m].

$$InfoGainEff(t) = \frac{TotalInfoGain(t)}{\sum_{UAV \in UAVs} Distance(t, UAV)}. \tag{5.2}$$

Communication Constraint Breakages (CCBs) are defined as loss of range or LoS communication between the UGV and one of the UAVs for an extended period of time t_{com}. This is defined further in Section 4.1. As with the previous performance metrics, the total number of CCBs up to the current time t are recorded.

5.2 Simulated environments

Three different environments were created to test the implemented systems. These environments are implemented as Gazebo simulator worlds with a fixed size of 60x60 meters. Walls are placed around the perimeter of the environments to limit UGV and UAV movement and mapping to be contained within the interior.

The environments created for this project are populated by simple geometric shapes of varying sizes and heights. These shapes include rectangles and cylinders. The simplicity of the environment geometry makes testing the implemented systems easier, as the local path planners do not have to avoid small or moving obstacles. The system is however tested in some of the realistic environments presented in Figure 3.3 for the Frontier exploration demo presented in Section 5.6.

The three different environments share the basic layout of shapes, however, they vary in the height of the shapes. The "Low shapes" Gazebo world is populated by obstacles with a height of 1 m, allowing UAVs to fly over the obstacles. In contrast, the "High shapes" Gazebo world is populated by obstacles with a height of 5 m, preventing UAVs from flying over the obstacles. Finally, the "Mixed shapes" world is populated by obstacles with a height of either 1 or 5 m, allowing UAVs to fly over some obstacles but not others. The worlds, along with maps of their layouts, are shown in Figures 5.1 to 5.3.

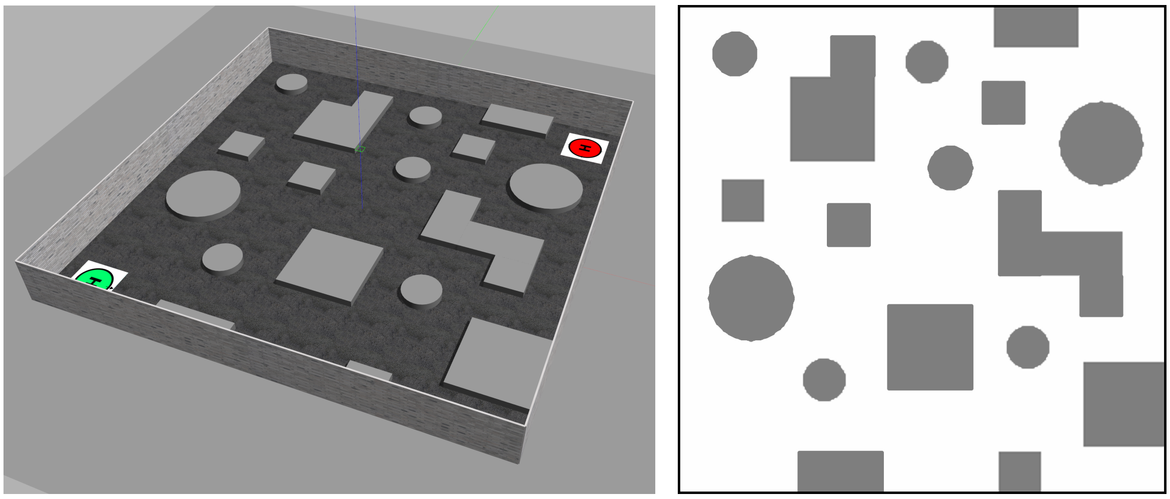
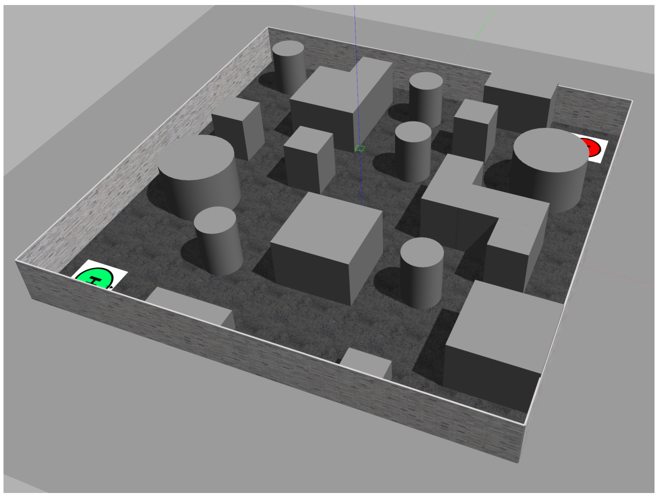

(a) Gazebo world. (b) Map.

Fig. 5.1 The "Low shapes" Gazebo world.


(a) Gazebo world.

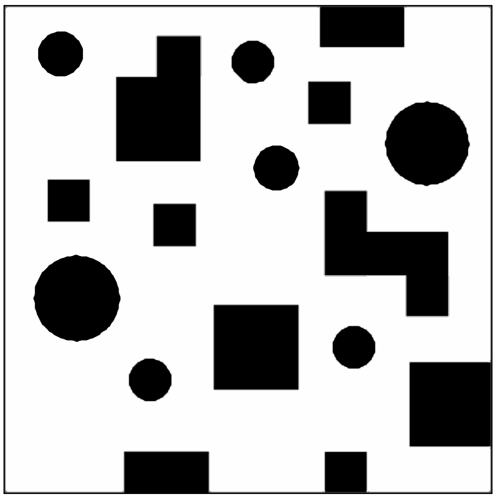
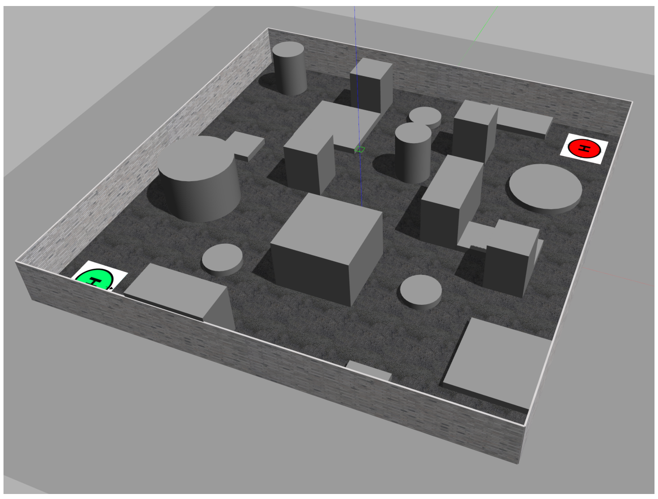

(b) Map.

Fig. 5.2 The "High shapes" Gazebo world.


(a) Gazebo world.

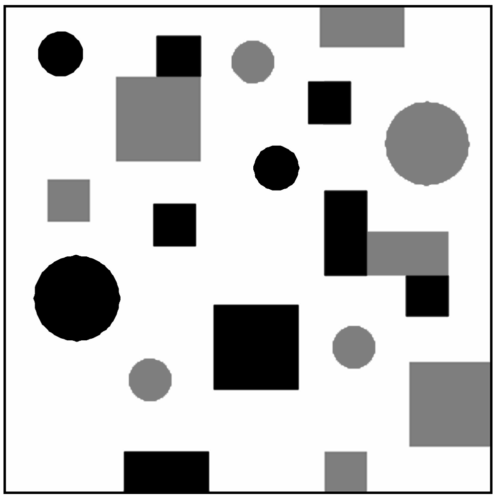

(b) Map.

Fig. 5.3 The "Mixed shapes" Gazebo world.

5.3 Stationary scenario

The first scenario the planners were tested in involves a stationary UGV, similar to a fixed Base Station (BS). In this scenario, the planners are tested to see how they perform in a given time-frame after takeoff and initialization is done. The planners must coordinate UAV movement to gather information about the environment around the UGV while maintaining communication constraints.

Results are gathered by running the planners on 1 to 3 UAVs in the different situations. Each variation is run 5 times in order to gain some indication of how consistent the planners are. The planners are given 30 s to coordinate UAV movement, after which planning is stopped and the performance variables are recorded.

Also of interest are the effects of increasing the communication range l_{com} from the UGV to the UAVs. Two communication ranges are tested in the stationary scenario: the default value l_{com}= 10.0 m and the increased value l_{com}= 20.0 m. The increase of communication range is visualized in Figure 5.4.

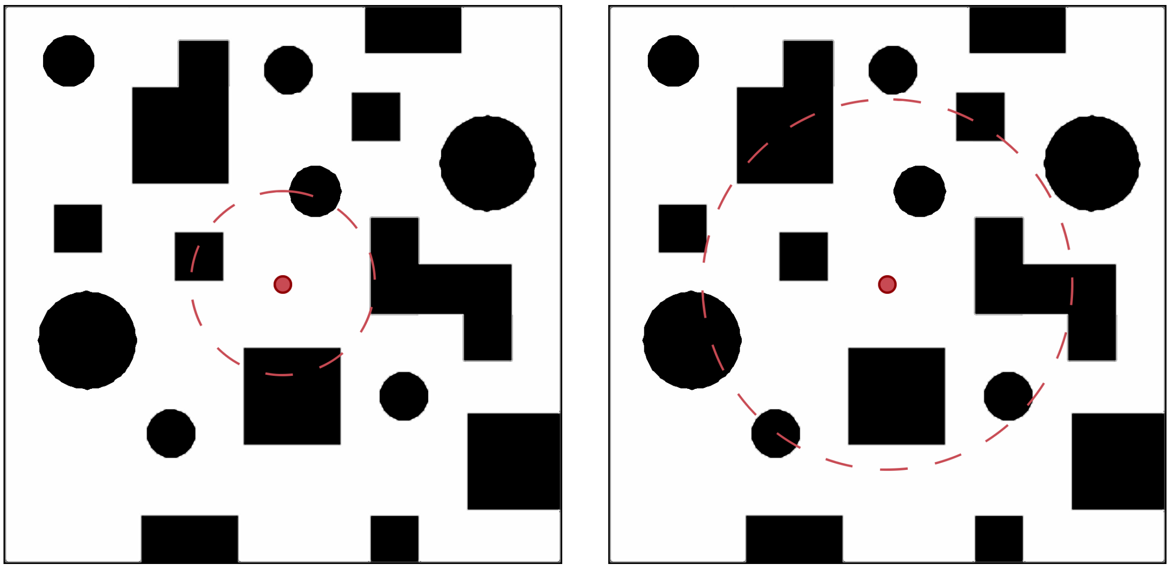

(a) l_{com}= 10.0 m. (b) l_{com}= 20.0 m.

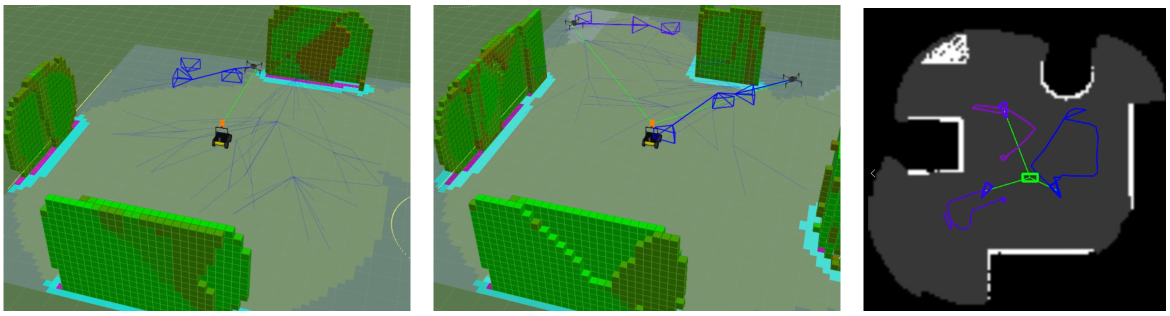
Fig. 5.4 Map of the Stationary scenario. The UGV position is indicated with a red circle in the middle. The dashed circle indicates the communication range l_{com}.

5.3.1 Different environments

Scenario	Environment	Nr. of UAVs	Variation
Stationary	"Low shapes", "High shapes"	$1-3$	Default parameter values in different environments.

First, the planners are tested in different environments with l_{com}= 10.0 m. The "Low shapes" environment is used to test performance for planning in unrestricted situations, where collision avoidance and LoS communication constraints are not a major factor. The "High shapes" environment is then used to see how the planners perform when these constraints must be considered.

In both environments, the Collaborative exploration planner is observed to have a greedy behaviour; quickly reaching the limits of the communication range and exploring the frontiers. With multiple UAVs on the "Low shapes" map, each UAV is able to occupy a "sector" of the communication constrained area and collaborative exploration is done efficiently. However, for the "High shapes" map, multiple UAVs are sometimes observed exploring the same "corridor" created by the tall shapes. This leads to one UAV having to backtrack or both getting stuck in sub-optimal positions.



(a) With 1 UAV. (b) With 2 UAVs. (c) Vehicle paths.

Fig. 5.5 Simulation of the Collaborative exploration planner in the stationary scenario on the "High shapes" map.

The Formation-based planners behave more conservatively. For all cases, the Simple formation-bank planner assumes the widest formation in the bank and stays there, leading to very little exploration beyond the initial movement to get into formation. The SFR planner is able to explore the environment, albeit at a much slower pace than the Collaborative exploration planner. Since whole formations are sampled, rather than individual poses, a formation change which would lead to higher information gain for one UAV may result in wasted distance travelled for another UAV. Therefore, the SFR planner is observed to get "stuck" in formations while new formations are being sampled, with the occasional formation change occurring when a good formation is sampled.

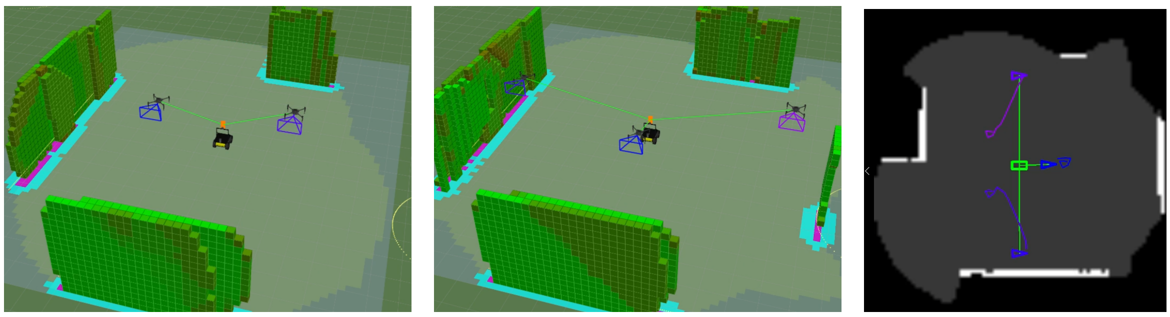

(a) With 2 UAVs. (b) With 3 UAVs. (c) Vehicle paths.

Fig. 5.6 Simulation of the Simple formation-bank planner in the stationary scenario on the "High shapes" map.

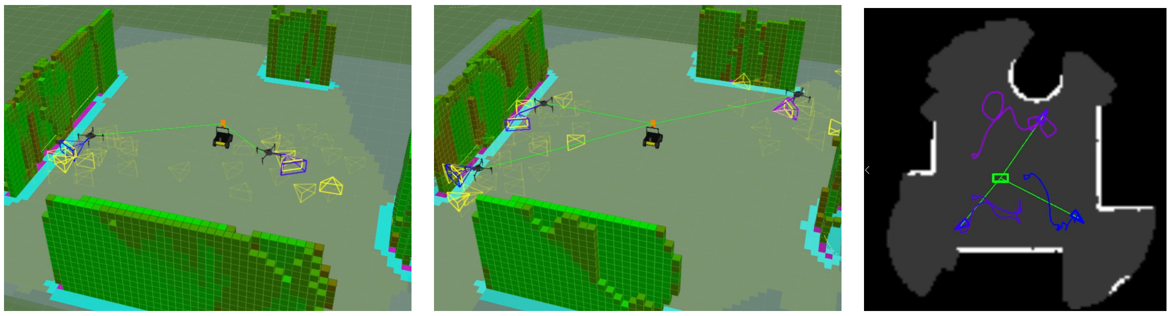
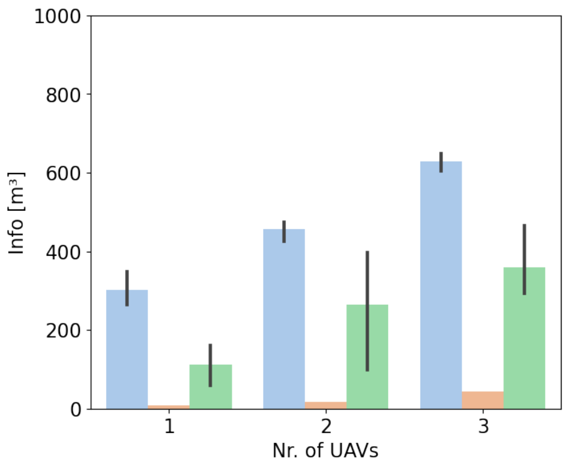

(a) With 2 UAVs. (b) With 3 UAVs. (c) Vehicle paths.

Fig. 5.7 Simulation of the SFR planner in the stationary scenario on the "High shapes" map.

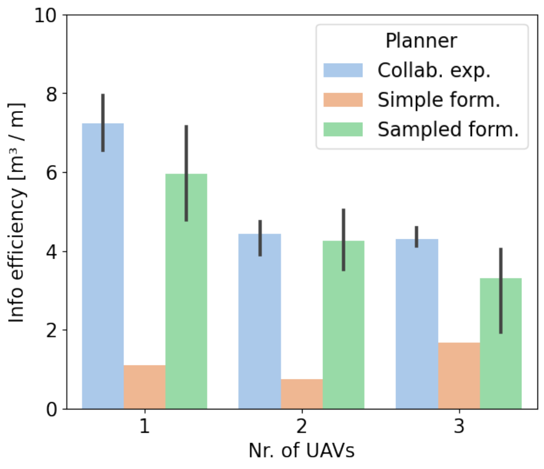
Low shapes

The results presented in Figure 5.8 support the observed behaviour, with the Collaborative exploration planner having a higher TIG than the other planners. The results for the SFR planner can also be seen to be more variable than the results for the Collaborative exploration planner. This can be tied to the formation sampling sometimes not being able to find good formations to transition to, as was observed in the behaviour.

In the IGE metric, the Collaborative exploration planner also performs better than its Formation-based counterparts. This could be due to the planner effectiveness in open environments where greedy exploration behaviour does not necessarily lead to the UAVs getting stuck, as is the case in more constrained environments.



(a) TIG.



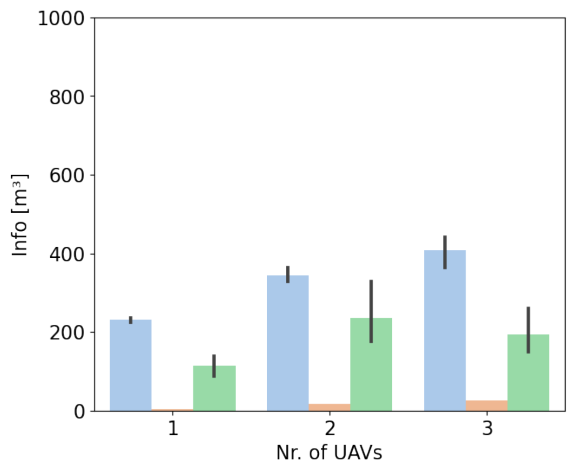
(b) IGE.

Fig. 5.8 Results from running the planners in the stationary scenario on the "Low shapes" map.

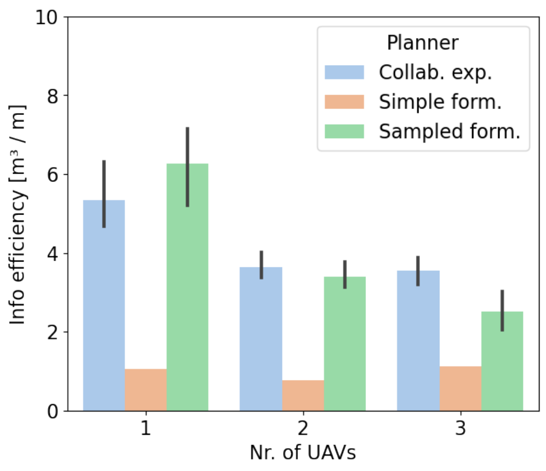
High shapes

The results presented in Figure 5.9 show that the TIG for the planners is lower, as is expected when obstacles cannot be flown over and the area is limited by communication constraints. The Collaborative exploration planner is observed to suffer more relative to the SFR planner in the TIG metric, however it is still able to gather more information. The variability of the SFR planner's performance is also reduced.

In the IGE metric the SFR planner is able to overtake the Collaborative exploration planner for the case of one UAV, but has similar results relative to the Collaborative exploration planner for the other cases as it did for the "Low shapes" map. Variability of performance is also reduced, with the Collaborative exploration planner and SFR planner having similar consistency in the TIG metric.



(a) TIG.



(b) IGE.

Fig. 5.9 Results from running the planners in the stationary scenario on the "High shapes" map.

5.3.2 Increased communication range

Scenario	Environment	Nr. of UAVs	Variation
Stationary	"Low shapes", "High shapes"	$1-3$	Communication range between the UGV and UAVs is increased.

Increasing the communication range effectively increases the area that the UAVs are able to explore, as can be seen in Figure 5.4. The "Low shapes" and "High shapes" environments are again used, so that results from increasing the communication range can be compared with the previously presented results.

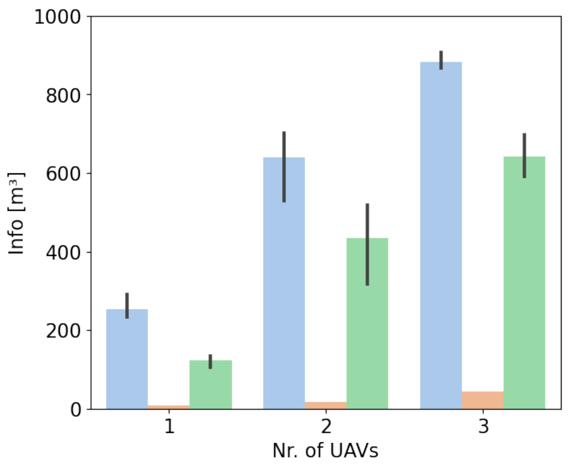
The Collaborative exploration planner is observed to utilize the increased range to explore more "deeply" into unmapped space. For the "Low shapes" environment this leads to more area being explored and a similar assignment of exploration "sectors" to each UAV as in the normal communication range scenario. In the "High shapes" environment, similar behaviour is observed. However, LoS communication constraints now begin to play a role in restricting the UAV movement.

The increased communication range naturally only affects the SFR variant of the Formation-based planners. A conservative behaviour similar to the behaviour with the normal communication range is observed, though the sampled formations are allowed to expand more. For the "High shapes" environment with multiple UAVs the planner is observed to struggle with sampling good formation changes as the UAVs are moved into the "corridors" created by the shapes. This can be due to the limits imposed by the LoS communication constraints in these areas.

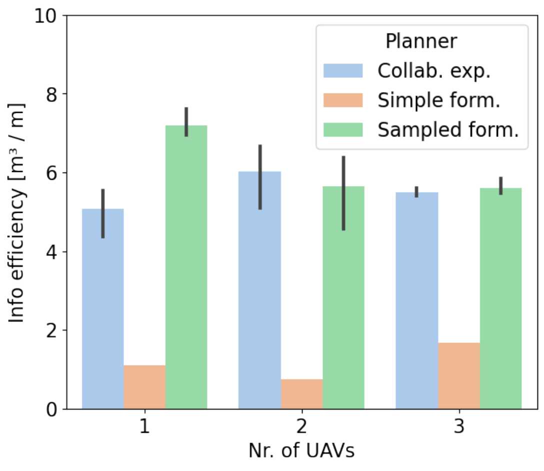
Low shapes

In the results presented in Figure 5.10 the TIG is seen to be higher for the Collaborative exploration and SFR planners, with the relative TIG between the two planners being similar to the previous cases. The TIG is also observed to scale to a higher degree when the number of UAVs is increased, with the increase from one to two UAVs being more than double.

IGE is now more consistent, unlike the previous cases where an increase in the number of UAV lead to worse efficiency. This is likely due to the increased amount of are the UAVs are allowed to explore.



(a) TIG.

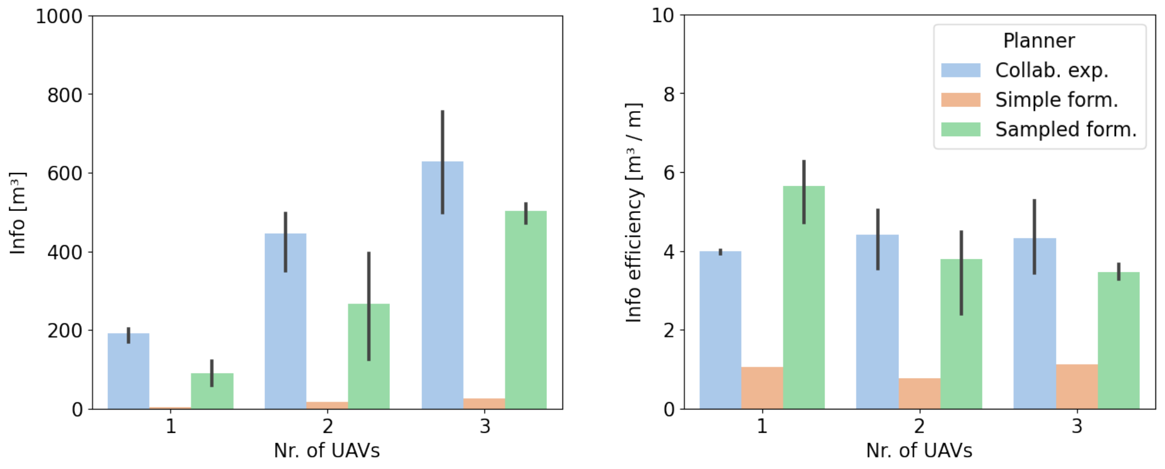


(b) IGE.

Fig. 5.10 Results from running the planners in the stationary scenario on the "Low shapes" map, with an increased communication range l_{com}= 20.0 m.

High shapes

The results presented in Figure 5.11 show a higher TIG relative to the lower communication range case, as expected. However, variability in the results is seen to increase relative to the "Low shapes" results, especially for the Collaborative exploration planner with multiple UAVs. This could be due to the increased negative effects of backtracking when multiple UAVs explore the same "corridor", as the backtracking distance to get to newly explored areas is increased together with the increased communication range.



(a) TIG.

(b) IGE.

Fig. 5.11 Results from running the planners in the stationary scenario on the "High shapes" map, with an increased communication range l_{com}= 20.0 m.

5.3.3 Conclusion

In all cases, the Collaborative exploration planner is able to gather more information about the area around a stationary UGV in a given time-frame than the other planners. IGE of the Collaborative exploration planner is also similar to the SFR planner in most cases, with the exception of the cases with one UAV.

Increasing the communication range naturally increased the amount of area to be explored. For the stationary case this naturally leads to higher TIG, and with the same time-frame as the lower communication range case this also translates to higher Information efficiencies for cases with multiple UAVs. Though the greedy behaviour shown by the Collaborative exploration planner leads to higher TIG, it may have consequences in cases where the UGV operation involves movement.

5.4 Path scenario

The second scenario centers on the UGV moving along a predefined path from one corner of the environment to the other. The path is relatively straight, with little requirement for the UGV to turn. During this operation, the planners must coordinate UAV movement to gather information around the moving UGV while considering collision and communication constraints. Results are again gathered by running the planners on 1 to 3 UAVs, with each variation being run 5 times in order to gain some indication of how consistent the planners are.

UAV coordination starts when the UGV receives its navigation goal and path, and finishes when the UGV is within a certain euclidean and angular distance from its navigation goal. Due to the nature of the simulated UGV movement, some variation in the completion time of the path navigation is expected. A mean completion time for the path scenario was recorded at 142.8 s, with a standard deviation of 0.68 s. As the observed variation in completion time is negligible, these variations were not taken into account when creating the results.

The path scenario is shown in Figure 5.12 for the "Mixed shapes" environment.

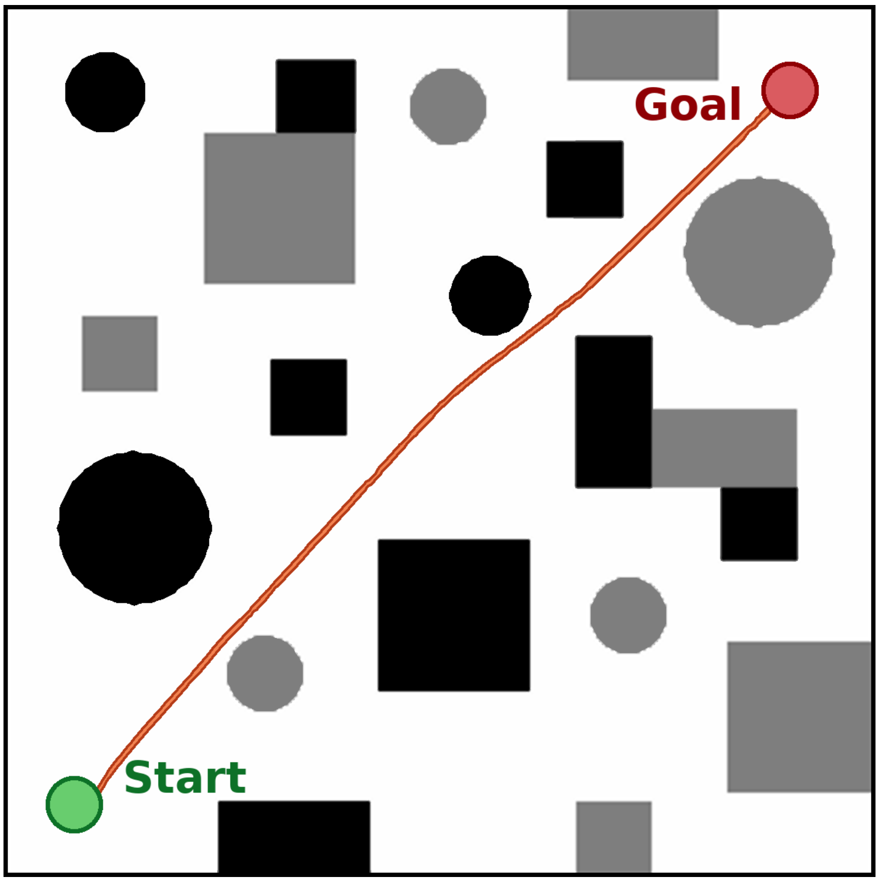


Fig. 5.12 Map of the path scenario. The UGV path is illustrated in orange.

5.4.1 Different environments

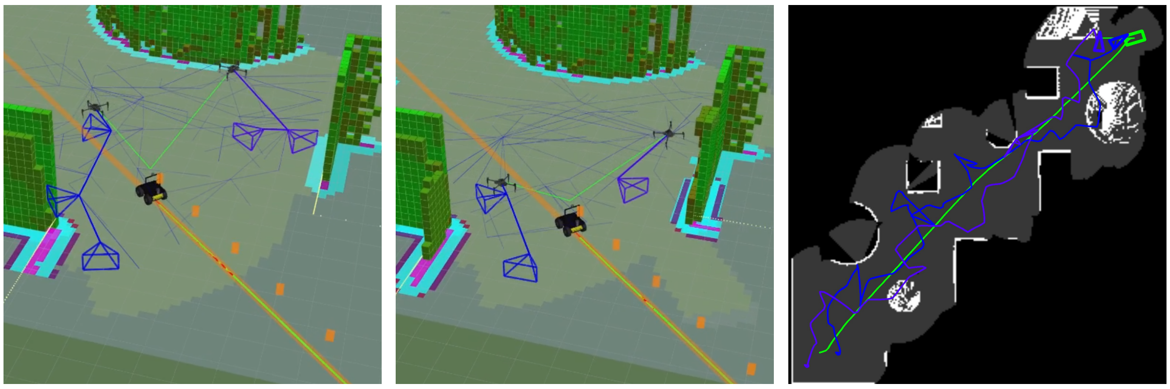
Scenario	Environment	Nr. of UAVs	Variation
Path	"Low shapes", "High shapes", "Mixed shapes"	$1-3$	Default parameter values in different environments.

First, all planners are run on the path scenario in the different environments to test how obstacles affect planning around a moving UGV.

The "Low shapes" environment is used to test the planner's ability to sweep over the largely unobstructed area around the UGV while it operates. Operation in the "High shapes" environment is analyzed to judge performance in obstructed areas, where the planners must both consider collision avoidance and LoS communication while attempting to map the area around the moving UGV. The "Mixed shapes" environment is then used, mainly to provide a benchmark for the other path scenario variations covered later in this section.

The Collaborative exploration planner is observed to perform well in the "Low shapes" environment with regards to the amount of area that is explored around the UGV. However, the planner displays a tendency to explore areas to the sides of the UGV rather than in front of it. This sometimes leads to the UGV entering unmapped space, in which case the Collaborative exploration planner calls for a recovery due to the conservative assumption that unmapped space is equivalent to occupied space with regards to LoS communication.

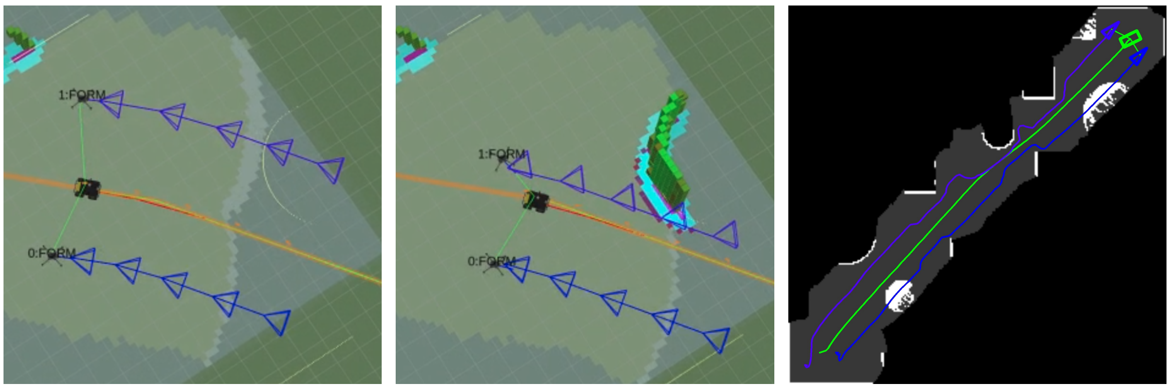
This issue is made worse in the "High shapes" environment, where the obstacles create situations where the UAVs get stuck in "corridors" to the side of the UGV path. In these situations, the UAVs are not able to move further out due to the communication constraints and the planner prefers not to move back to the UGV due to the negative utility of backtracking through mapped space. This either results in the UAVs only moving when there is no other option (when the current "stuck" position is no longer communication constrained for the pair of UGV waypoints) or recovery having to be called due to broken communication constraints.



(a) Plan 1. (b) Plan 2. (c) Vehicle paths.

Fig. 5.13 Simulation of the Collaborative exploration planner in the path scenario on the "Mixed shapes" map. (a) Plans are sampled and selected asynchronously. (b) New plans are selected as more information about the environment is acquired.

The Formation-based planners coordinate UAV movement relative to the UGV, and as such is able to handle a moving UGV situation well. The Simple formation-bank planner is able to sweep the area effectively in the "Low shapes" environment, and can quickly contract and expand to accommodate the obstacles encountered in the "High shapes" environment. Formation changes are observed to happen quickly but conservatively, with the formation contracting immediately after detecting an obstacle and only expanding once it has mapped space to expand into.

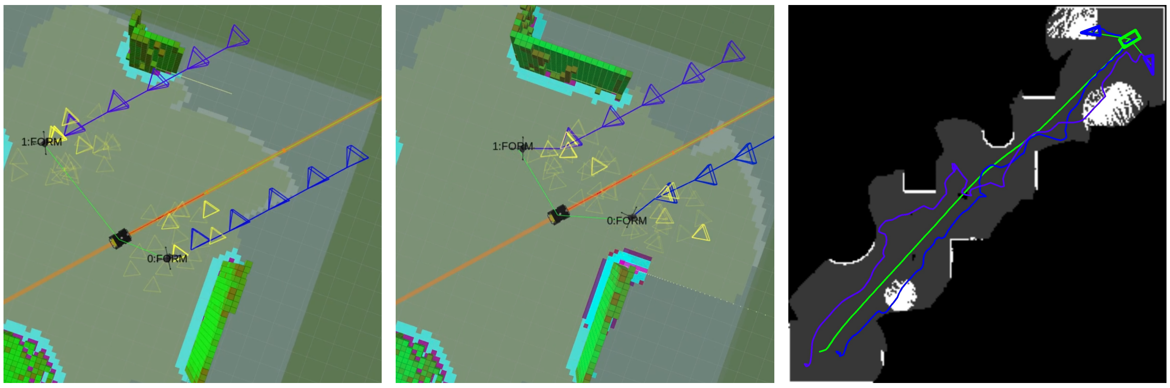


(a) Wide formation. (b) Narrow formation. (c) Vehicle paths.

Fig. 5.14 Simulation of the Simple formation-bank planner in the path scenario on the "Mixed shapes" map. (a) The formation is wide in open areas. (b) One UAV chooses a closer formation due to a discovered obstacle.

Similarly, the SFR planner is able to follow the UGV and sweep the area in the "Low shapes" map. The sampled formation changes also tend to make the UAVs sweep side to side, similar to the "lawn mower pattern" described in previous chapters. This results in more area being mapped when compared to the Simple formation-bank planner. In the "High shapes" environment the SFR planner is quick to detect and avoid new obstacles.

Some anomalous behaviour was observed for the SFR planner runs. In obstructed environments, the planner is observed to struggle with assuming a wider formation after having contracted to avoid obstacles, leading to it keeping a narrow formation even in open areas. Some oscillating behaviour was also observed when new obstacles are detected. This involves the formation contracting briefly to the viable recovery formation pose above the UGV and quickly sampling and assuming new formation poses further away. As more of an obstacle is revealed, this sometimes leads to oscillations between a wide sampled formation and the narrow recovery formation.



(a) Wide formation. (b) Narrow formation. (c) Vehicle paths.

Fig. 5.15 Simulation of the SFR planner in the path scenario on the "Mixed shapes" map. Formation samples are shown as yellow FOV pyramids, with an increased opacity indicating high utility (a) Formations are sampled in an open area. (b) New formations are chosen to avoid collision when an obstacle is discovered.

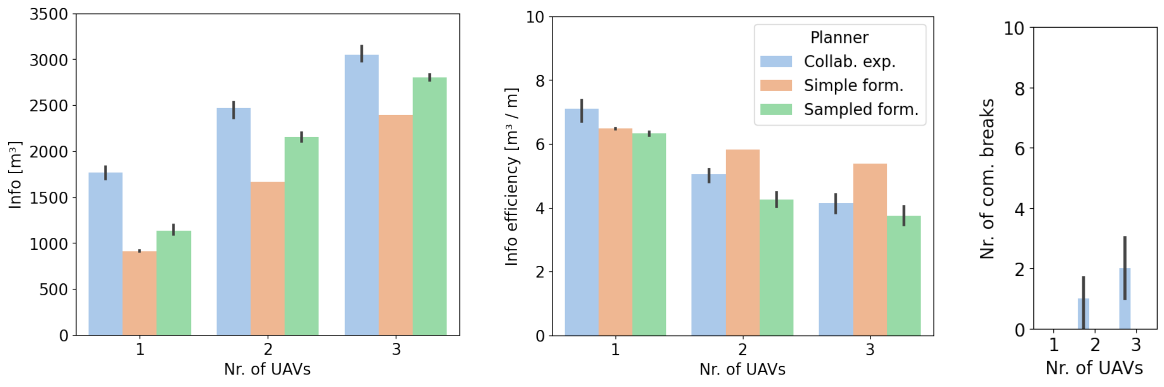
Low shapes

The results from running the planners in the "Low shapes" environment are shown in Figure 5.16. The Collaborative exploration planner is able to achieve the highest TIG, especially in the case with one UAV. The performance of the Formation-based planners does however scale better with the number of UAVs, with the SFR planner having similar results for three UAVs.

For IGE the Simple formation-bank planner is able to gather more information pr. meter travelled when compared to the other planners, with the exception of the single-UAV case.

This indicates that the simple wide formation that is kept by the planner through the whole path is an efficient way to gather information in unobstructed environments.

CCBs are also recorded for the runs. The Collaborative exploration planner is the only planner for which such breakages are observed. For the "Low shapes" environment the CCBs are mainly cause by the UGV entering unmapped space, causing the UAVs to not have LoS with the UGV. In most cases the planner is able to notice this and initiate recovery before a communication timeout occurs. However, in some cases this is not done quickly enough and a CCB is recorded.

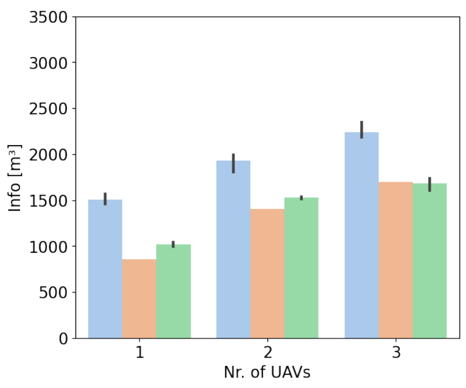


(a) TIG. (b) IGE. (c) CCBs.

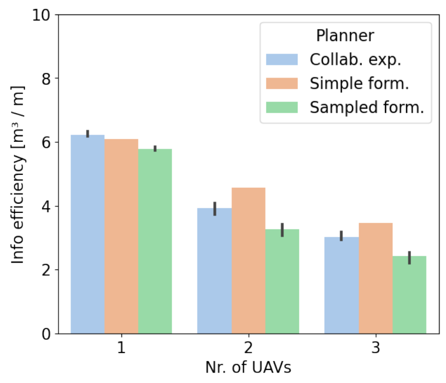
Fig. 5.16 Results from running the planners in the path scenario on the "Low shapes" map.

High shapes

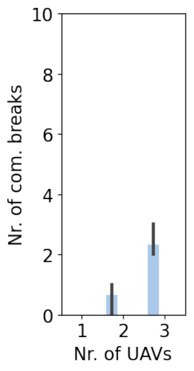
Results from the "High shapes" scenario are shown in Figure 5.17. Both TIG and IGE are in most cases similar when compared to the "Low shapes" cases, with the SFR planner having the greatest relative performance drop. This performance drop is likely due to the sub-optimal behaviour commented above, where a narrow formation is kept even when a wider formation is possible. A similar amount of CCBs are recorded for the Collaborative exploration planner as before, and are similarly caused by the UGV entering unmapped space.



(a) TIG.



(b) IGE.

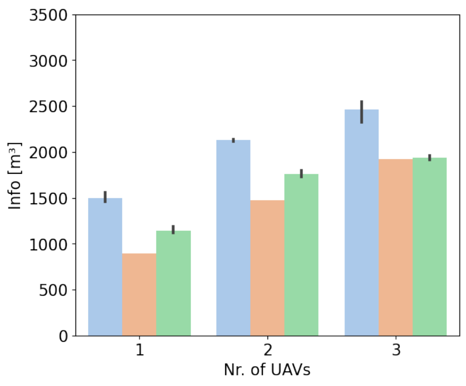


(c) CCBs.

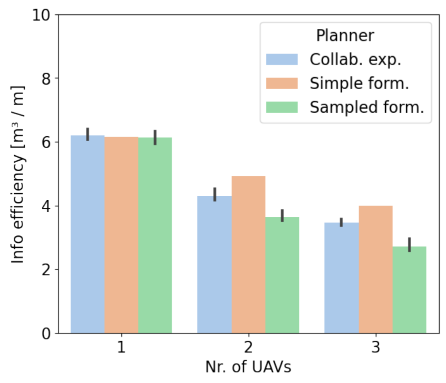
Fig. 5.17 Results from running the planners in the path scenario on the "High shapes" map.

Mixed shapes

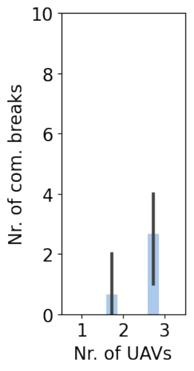
The results from the "Mixed shapes" scenario shown in Figure 5.18 resemble a blend of the results from the two other environments. This is to be expected, as the environment is created as a blend of the two other environments.



(a) TIG.



(b) IGE.



(c) CCBs.

Fig. 5.18 Results from running the planners in the path scenario on the "Mixed shapes" map.

5.4.2 Increased UGV speed

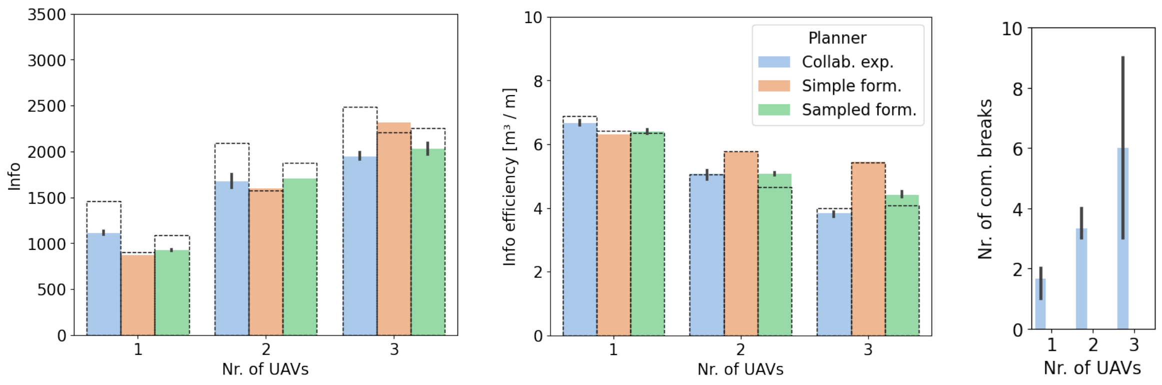
Scenario	Environment	Nr. of UAVs	Variation
Path	"Low shapes"	$1-3$	Increase UGV speed to see effects of speed on the planners.

In one variation of the path scenario, the UGV speed is increased from its default value of $v_{ugv}= 0.5$ m/s to $v_{ugv}= 1.0$ m/s. This effectively reduces the amount of time each UAV has to plan and act to maintain communication with the moving UGV. The increased UGV speed variation is tested in the "Low shapes" environment.

The Collaborative exploration planner is observed to suffer the most from the increased UGV speed. CCBs and subsequent recoveries occur multiple times, triggered by the UAVs exploring to the sides of the UGV and spending too much time planning. This allows the UGV to enter unmapped space, and the increased UGV speed results in the UAVs having to travel further to recover.

Increasing the UGV speed is not observed to affect the Formation-based planners' performance. The Simple formation-based planner is still able to follow the UGV as designed, and the lack of obstacles in the environment allows it to effectively sweep the unmapped area as before. The SFR planner has a similar performance, however, it is not able to perform the "side to side" sweeping motions as effectively as before.

The results are shown in Figure 5.19, with the results from Figure 5.16 overlayed in dashed lines for comparison. The decrease in performance is most apparent in the TIG for the Collaborative exploration planner. However, the SFR planner's performance is also seen to decrease, both for TIG and IGE. The greatest difference between the default and increased UGV speed results is in the recorded number of CCBs, where the Collaborative exploration planner in some cases has three times the amount of recorded breakages.



(a) TIG. (b) IGE. (c) CCBs.

Fig. 5.19 Results from running the planners in the path scenario on the "Low shapes" map, with an increased UGV speed $v_{ugv}= 1.0$ m/s. Results with default value $v_{ugv}= 0.5$ m/s are plotted as dashed bars.

5.4.3 Adjusted cost parameters

Scenario	Environment	Nr. of UAVs	Variation
Path	"Mixed shapes"	$1-3$	Adjust utility cost parameters to weight information gain lower relative to distance travelled.

The cost parameters c_{info}, c_{euc} and c_{ang} for information gain, euclidean distance and angular distance respectively can also be tuned to affect planner performance. This is the case for the Collaborative exploration planner, which utilizes the utility function shown in Equation (4.4), and the SFR planner, which utilizes Equation (4.5). For both functions, information gain is weighted positively while both distance metrics are weighted negatively.

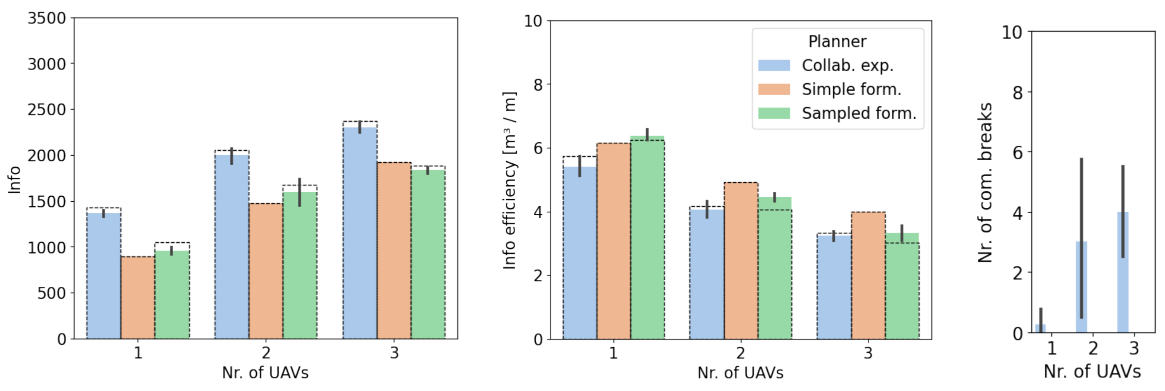
The default cost parameter value for information gain c_{info} is maintained at 100.0, while both the cost parameters for euclidean distance c_{euc} and angular distance c_{ang} are increased in magnitude from -1.0 to -10.0 for both parameters. This adjustment affects the choice of exploration plan or formation change, favoring actions which don't require as much distance to be covered to gather information. The planners running with the adjusted cost parameters are tested in the "Mixed shapes" scenario.

For the Collaborative exploration planner, the adjustment does not noticeably affect the performance of the planner during normal operations. Some behavioural changes are however observed in situations where a UAV gets stuck in a "corridor" and has to backtrack through largely mapped space. In such cases, the frequency of the UAV rejecting plans is increased, leading to the UAV being idle until it is forced by communication constraints to return to the UGV through recovery behaviour.

The adjustment is observed to mitigate the oscillation that could be observed for the SFR planner. This is likely due to the adjusted cost parameters favoring incremental formation changes rather than the abrupt formation changes which could lead to oscillations. However, this also had the perceived effect of prolonging the time the SFR planner spent in a "contracted" formation to avoid an obstacle, before assuming a wider formation in an open area.

The results for running the planners with the adjusted cost parameters are presented in Figure 5.20, with the default results from Figure 5.18 overlayed as dashed lines. A slight decrease on the TIG metric can be observed for both the affected planners. However, IGE is affected positively for the SFR planner while the Collaborative exploration planner has a reduced efficiency. These changes are however relatively small.

One noticeable difference is the increased rate at which CCBs are recorded for the Collaborative exploration planner. This corresponds to the increased occurrences of the UAVs getting "stuck" that was observed. A possible explanation for this behaviour and result is that the Collaborative exploration planner penalizes all movement, even movement which is necessary to maintain the communication constraints. Some of this movement should be discounted by the "free distance" discussed in Section 4.2.3. However, several factors could cause the movement to not be discounted sufficiently. This, along with an added cost for any remaining movement, discourages the UAV from moving from these sub-optimal positions.



(a) TIG. (b) IGE. (c) CCBs.

Fig. 5.20 Results from running the planners in the path scenario on the "Mixed shapes" map, with a increased euclidean distance cost $c_{euc} = -10.0$ and angular distance cost $c_{ang} = -10.0$. Results with default values $c_{euc} = -1.0$ and $c_{ang} = -1.0$ are plotted as dashed bars.

5.4.4 UGV waypoint utility

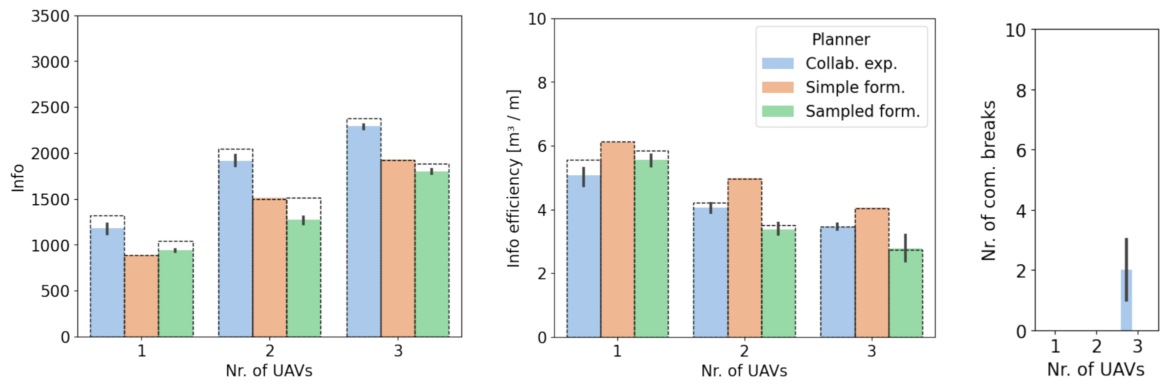
Scenario	Environment	Nr. of UAVs	Variation
Path	"Mixed shapes"	$1-3$	Use a modified utility function to encourage mapping of area in front of the UGV.

To address the issue observed with the Collaborative exploration planner losing communication due to the UGV entering unmapped space, a modified utility function was implemented which positively weights the information gathered in areas directly in front of the UGV. This is done using the distance from a UAV's potential sensor coverage to the closest local UGV navigation waypoint and multiplying the information gain utility with a multiplier that is inversely proportional to this distance. This new utility function is then used by the Collaborative exploration planner and the SFR planner to determine the next plan or formation change. A more detailed explanation of the modified utility function is presented in Chapter C.

For the Collaborative exploration planner, the modification has noticeable effects on the behaviour. The planner often chooses plans which lead to the UAVs exploring the area directly in front of the UGV, as intended. For multiple UAVs, one UAV is in most cases observed exploring the area in front of the UGV while the other UAVs explore to the sides. This reduces the amount of recoveries that have to be called due to the UGV entering unmapped space.

The modified utility function is also applied to the SFR planner. The formations the planner selects are observed to include UAVs with coverages that cover the local UGV navigation waypoint, as expected. However, the tolerance for overlapping coverages seems to increase. This is observed as formations are chosen which have overlapping coverages close to the local UGV navigation waypoints, leading to less explored area to the sides of the path.

The results shown in Figure 5.21 support the observed behaviour. The Collaborative exploration and SFR planners both have reduced TIG and IGE. However, the number of recorded CCBs for the Collaborative exploration planner is significantly reduced.



(a) TIG. (b) IGE. (c) CCBs.

Fig. 5.21 Results from running the planners in the path scenario on the "Mixed shapes" map, with an added local waypoint multiplier function to modify planner estimated information gain. Results without the local waypoint multiplier function are plotted as dashed bars.

5.4.5 Conclusion

In this section, each planner was tested in a moving UGV scenario on several different environments and with several different variations.

The Collaborative exploration planner is in general observed to have the highest TIG, meaning it is able to explore more of the environment than the other planners. The planner also performs well on the IGE metric, usually performing as well as or slightly better than the SFR planner. However, the Collaborative exploration planner is also plagued by the need for recovery behaviour due to CCBs. The amount of CCBs increases for higher UGV speeds and when the utility cost parameter for travelled distance is raised relative to information gain.

This problem is alleviated by the modified utility function which takes the local UGV navigation waypoints into account, but at some cost to the TIG and IGE. For the remainder of the results, the Collaborative exploration planner will utilize the modified utility function to reduce the number of CCBs. Results comparing the two variants further can be found in Chapter C.

The Simple formation-bank planner performs consistently, albeit poorly relative to the other planners, in the TIG metric. However, the planner performs well in the IGE metric, specifically when planning for multiple UAVs. Despite the planner's simplicity, it is able to map the area around a moving UGV efficiently while avoiding obstacles and maintaining communication constraints.

The SFR planner is able to map more area than the Simple formation-bank planner, but less than the Collaborative exploration planner. However, the planner's IGE is the lowest when default parameter values are considered. Changing the distance cost parameters is observed to lead to better efficiency, but at the cost of TIG.

5.5 Tour scenario

The third scenario simulates a longer UGV "tour", containing multiple tour waypoints where the UGV stops and path segments connecting these. This scenario includes elements of the two previous scenarios, with the addition of transitions between a stationary and a moving UGV.

A map of the tour scenario is presented in Figure 5.22. The tour contains three path segments and two tour waypoints. The first path segment is relatively straight and contains a mixture of low and high shapes, resembling a shortened path scenario on the "Mixed shapes" map. In contrast, the second path segment contains two sharp turns and an increased density of obstacles. This is intended to present a challenging area for the planners to coordinate UAV movement through. Finally, the third path segment first contains a U-turn for the UGV, followed by an area containing high obstacles on one side and low obstacles on the other. Both tour waypoints are located in corner areas of the map and are relatively open in that the surrounding obstacles have a low height.

For each tour waypoint, the UGV is stopped for 20 s before moving on the next path segment. The recorded mean completion time of the tour is 358.5 s with a standard deviation of 0.7 s.

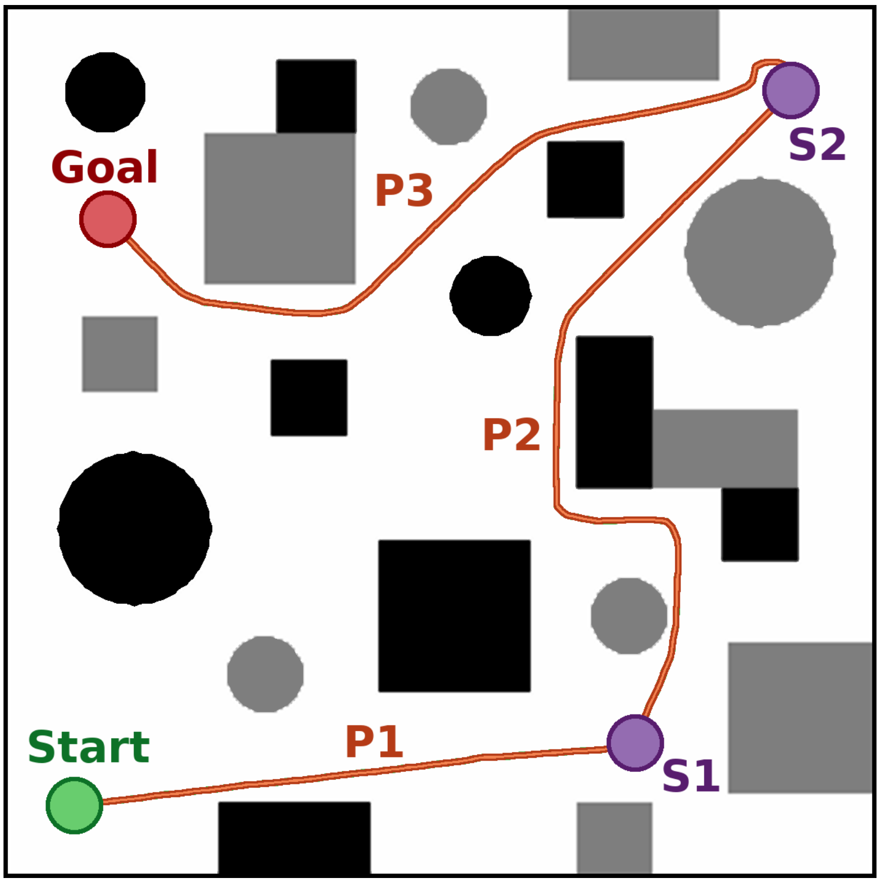


Fig. 5.22 Map of the Tour scenario. Path segments are illustrated and annotated in orange. Stationary waypoints are illustrated and annotated in purple.

Two additional planners were implemented for testing in the tour scenario: the Hybrid planner and the Split planner. These were developed following the results and observed behaviours of the planners in the two previous scenarios. All five planners were run in the tour scenario 5 times, with 2 UAVs and default parameter values[1].

Scenario	Environment	Nr. of UAVs	Variation
Tour	"Mixed shapes"	2	None.

Planner behaviour

The behaviour for the Collaborative exploration and Formation-based planners is similar to the behaviour observed in the previous scenarios. For the path segments, the Formation-based planners is able to map the area around the moving UGV, though not as effectively as the Collaborative exploration planner. The modified utility function used by the Collaborative exploration planner also mitigates the number of needed recovery behaviours, with the exception of some CCBs observed in the second path segment. Behaviour in the tour waypoint areas resembles the behaviour observed in the stationary UGV scenario, with the Collaborative exploration planner exploring more area than the Formation-based planners.

Some issues with the Formation-based planners are observed during sharp UGV turning maneuvers. As the formation is held relative to the UGV, the rotation of the UGV during a turn translates to rapid movement for the UAVs in order to keep their formation poses. The needed movement increases for formation poses which are further away from the UGV. When such a turn is performed close to obstacles, the planners are sometimes not reactive enough to choose more narrow formations to avoid collision with obstacles.

For the Hybrid planner, the planner is designed to utilize different planning methods for the different areas encountered on the tour. For the path segments, planning using the Simple formation-bank planner ensures a consistent mapping behaviour and no loss of communication or collision with obstacles. The planners switches to the Collaborative exploration planning method as the UGV approaches a tour waypoint, and the UAVs collaboratively explore the area while the UGV is stationary.

Some issues are observed when the planner switches back to the Formation-based planner. One issue arises from the UAVs having to travel from their exploration positions to their respective formation poses once the planner switch occurs. As no explicit collision avoidance

[1]The Collaborative exploration planner will make use of the modified utility function which takes into account the local UGV navigation waypoints. An explanation of the modified utility function, along with results comparing the performance of both planner variants on the tour scenario, are found in Chapter C.

is used during this movement, collision with obstacles is sometimes observed. Other issues similar to the turning-related issue for the Formation-based planner are also present.

In the path scenario, the Split planner coordinates one UAV using the Collaborative exploration planner and the other using the SFR planner. The planner is observed to behave largely as intended, with the SFR planned UAV planning ahead of the UGV while the Collaborative exploration planned UAV maps the areas to the side of the UGV. Having the area in front of the UGV mapped by one UAV largely eliminate the occurrences of LoS communication loss which affected the Collaborative exploration planner. Recovery behaviour and CCBs were still observed however, largely tied to the exploring UAV navigating to an area on the edge of the communication range and subsequently not being able to plan a route back to the UGV before losing communication.

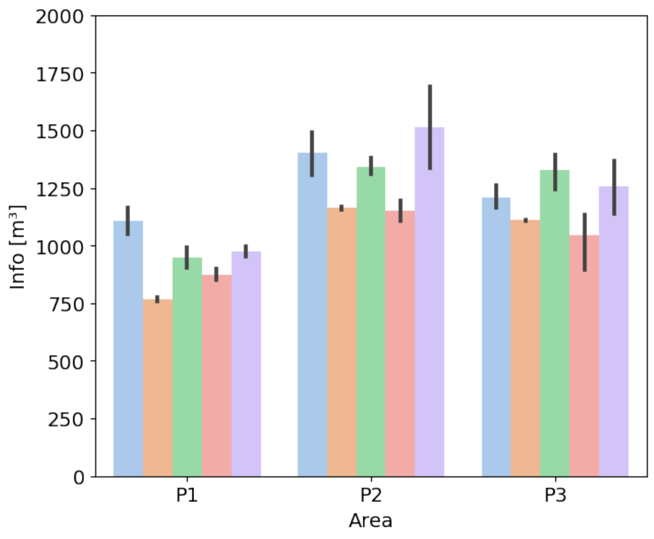
Results

TIG results from the path segments and tour waypoints are shown in Figure 5.23. The TIG from each area, defined by when the UGV transitions from moving to stationary and vice-versa, was recorded separately.

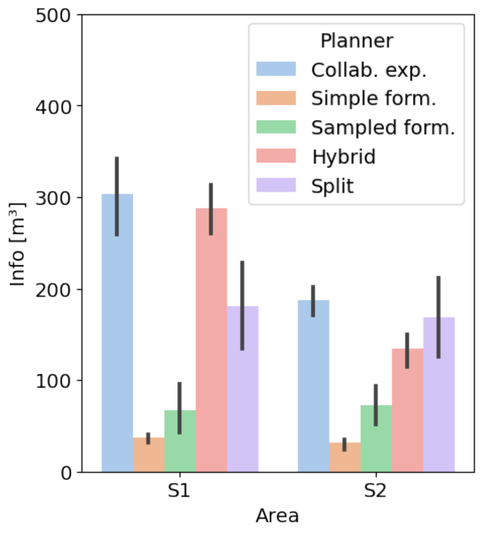
For the path segments the TIG of the Collaborative exploration and Formation-based planners bear resemblance to the previously presented path scenario results, apart from in the third path segment where the SFR planner gets a higher TIG than the Collaborative exploration planner. Tour waypoints results however are skewed more in favor of the Collaborative exploration planner than the results for the similar Stationary UGV scenario. The SFR planner has a very low TIG in these areas.

The newly introduced Combined planners have performance similar to the planners they combine features from. Since the Hybrid planner uses the Simple formation-bank planner in the path segments and switches to the Collaborative exploration planner for tour waypoints, its performance in each of these areas is similar to the results of the respective planner.

The Split planner performs very well in the TIG metric, matching and in some cases surpassing the Collaborative exploration planner. This indicates that having one UAV exploring ahead of the UGV and preventing LoS communication loss allows the other UAV to explore effectively. The TIG recorded for the Split planner around tour waypoints is however lower than for the Collaborative exploration planner, reflecting that it only has half the amount of exploring UAVs.



(a) Path areas.

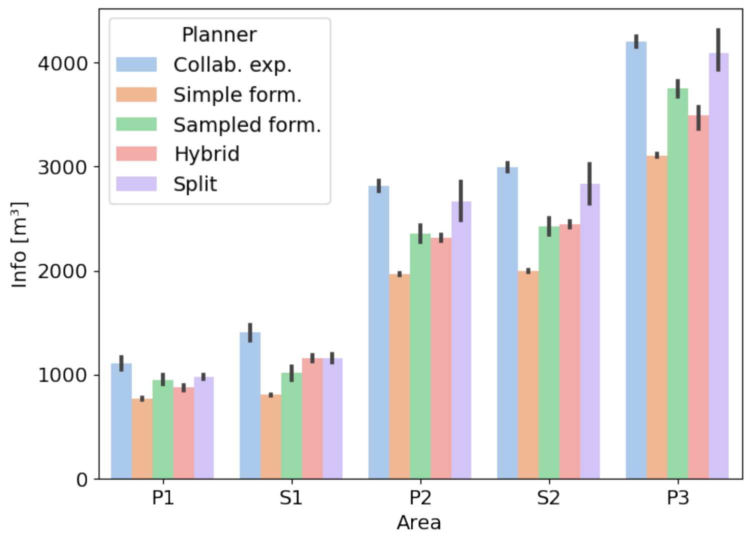


(b) Stationary areas.

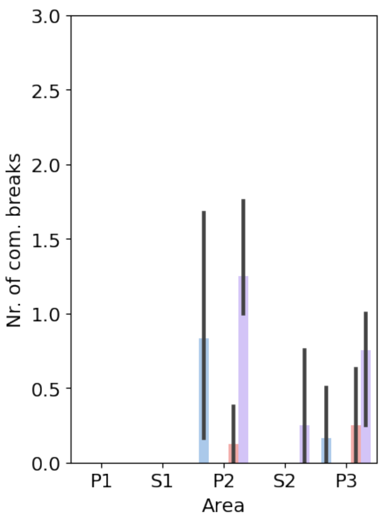
Fig. 5.23 TIG results from running the planners in the tour scenario on the "Mixed shapes" map, with TIG recorded individually for each area. (a) TIG from path segments, where the UGV is in movement (b) TIG from tour waypoints, where the UGV is stationary.

The cumulative TIG is also plotted in Figure 5.24, along with the recorded CCBs. Both the Collaborative exploration and Split planners are able to accumulate the highest TIG, with a higher variance in the Split planner results stemming from variable performance on the final two path segments. The SFR planner achieves the third highest cumulative TIG, performing better than the two other planners that make use of the Simple formation-bank planner. Though the Split planner is able to achieve a higher TIG than the pure Simple formation-bank planner, the added information gained from using the Collaborative exploration planner around the tour waypoints is not significant.

CCBs are also recorded for each area. The planners which make use of the Collaborative exploration planner are the only planners for which CCBs are observed. For the pure Collaborative exploration planner, the use of the modified utility function contributes to reducing the amount of CCBs. The Split planner is observed to have a higher and more consistent amount of CCBs, despite only having one exploring UAV. The CCBs are most prevalent in the second path segment, which was previously identified as a difficult area due to the concentration of obstacles and sharp turns. Some CCBs are also recorded for the Hybrid planner, though these are observed to occur when the planner switches modes.



(a) Cumulative information gain.



(b) CCBs.

Fig. 5.24 Cumulative TIG and recorded CCBs results for the tour scenario on the "Mixed shapes" map.

The IGE was recorded for each area and is shown in Figure 5.25. For the path segments, IGE performance is relatively equivalent for all planners, with the exception of the Simple formation-bank planner having good IGE performance in the last two path segments. This is similar to the behaviour observed for the path scenario, where the Simple formation-bank planner's low TIG was accompanied by a high IGE. The Hybrid planner is observed to have a lower IGE than the Simple formation-bank planner in the path areas of the tour, which is an interesting result as the Hybrid planner makes use of the Simple formation-bank planner in these areas and hsould have equivalent performance. However, due to the switching distance l_{hybrid}, the planner actually switches to the Collaborative exploration planner while the UGV is still moving, which explains the reduced efficiency. The Split planner is also observed to have a highly variable IGE in the stationary areas, though no explanation for this can be provieded.

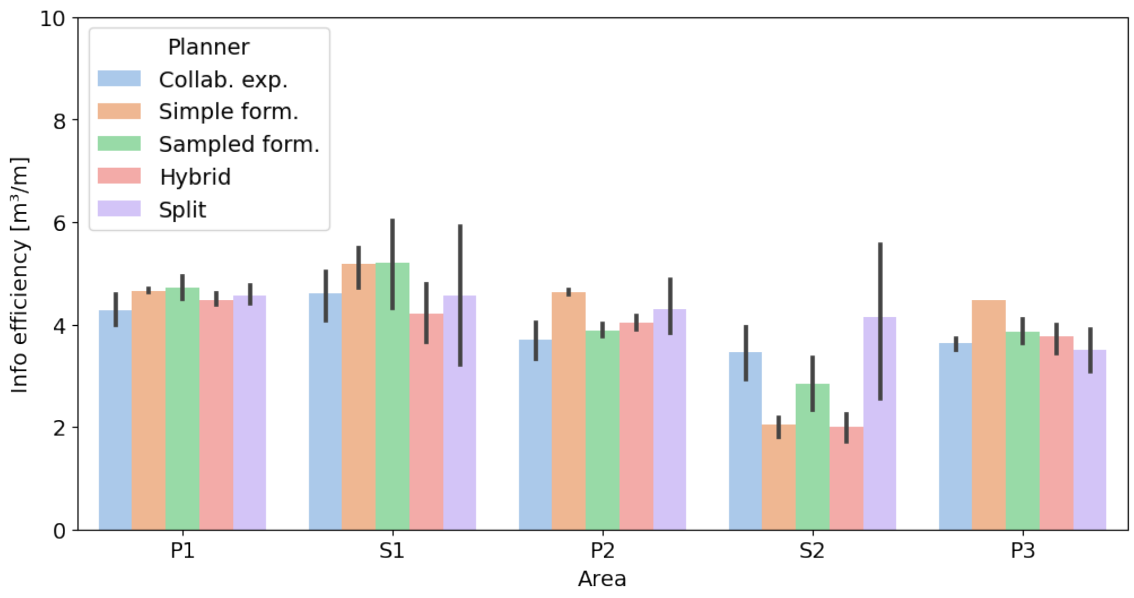


Fig. 5.25 IGE results from running the planners in the tour scenario on the "Mixed shapes" map, with IGE recorded individually for each area.

5.5.1 Conclusion

In this section, all five planners implemented for this project were tested in an extensive tour scenario. This scenario blended the two previous scenarios by including tour waypoints where the UGV would remain stationary and path segments for the UGV to move along. Transitions between the UGV being stationary and moving were also included in this scenario.

The Collaborative exploration planner and the two variants of the Formation-based planner has similar performance as had been observed in the previous scenarios. Some added issues for the Formation-based planners were observed due to the sharp turns included in the tour scenario, which were not present in the stationary or path scenarios. The inclusion of the modified utility function for the Collaborative exploration planner also significantly reduced the recorded number for communication constraints breakages.

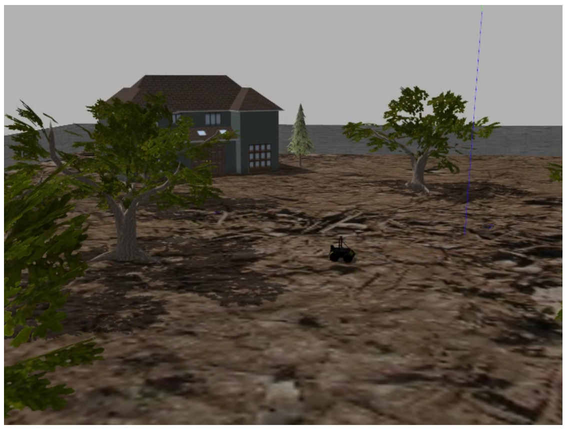
The Hybrid planner functioned as intended and switched between two planners during operation in the tour scenario. The conservative Simple formation-bank planner was utilized for the path segment areas of the tour, and provided a consistent mapping of the area around the moving UGV. For the tour waypoints, a switch tot eh Collaborative exploration planner allowed the area around the waypoint to be mapped extensively. Some issues were observed when the planner would switch from exploration to formation mode, as collision avoidance was not applied for the UAVs getting back into formation.

The Split planner has good performance on the TIG metric, with comparable performance with the Collaborative exploration planner. It does however have highly variable performance in some areas, as is observable in the IGE metric for tour waypoints. Though the SFR planner should prevent a higher number of CCBs, it still records more communication losses than the Collaborative exploration planner with the modified utility function.

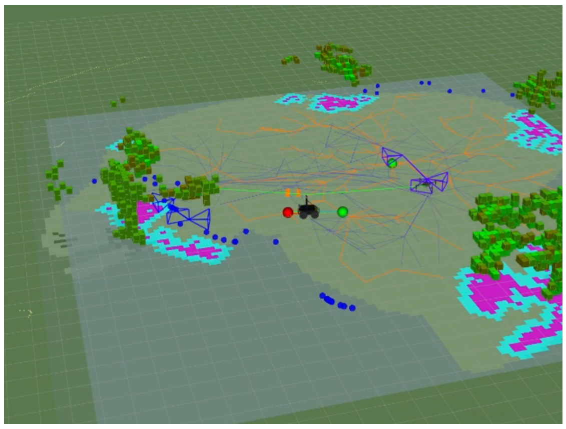
5.6 Exploration scenario

The final scenario makes use of the Frontier-exploration planner presented in Section 3.2.2 to drive the UGV in an exploration scenario. Unlike the previous scenarios, the sampling-based nature of the Frontier-exploration planner leads highly to variable performance and exploration behaviour. As such, no results were recorded for the exploration scenario.

The exploration behaviour is showcased here to demonstrate a potential use-case for the developed UAV coordination systems and planners.

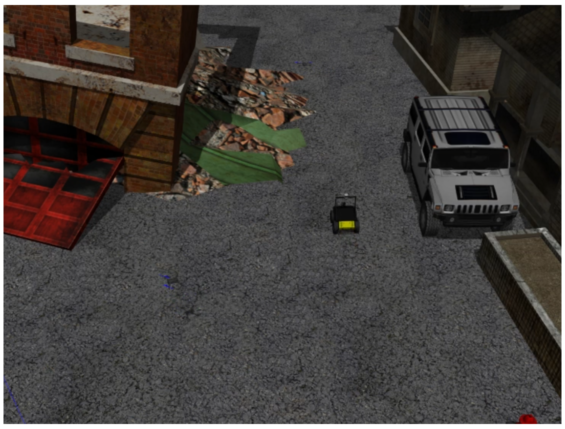


(a) Gazebo world.



(b) `Rviz` visualization.

Fig. 5.26 Simulation of the exploration scenario on the "Outdoor" map. (a) The Gazebo GUI allows the viewing of the robot system during in simulation. (b) The UGV exploring frontiers points, which are visualized as blue spheres. Frontier points are found using a modified RRT* algorithm, with the tree visualized in orange.


(a) Gazebo world.

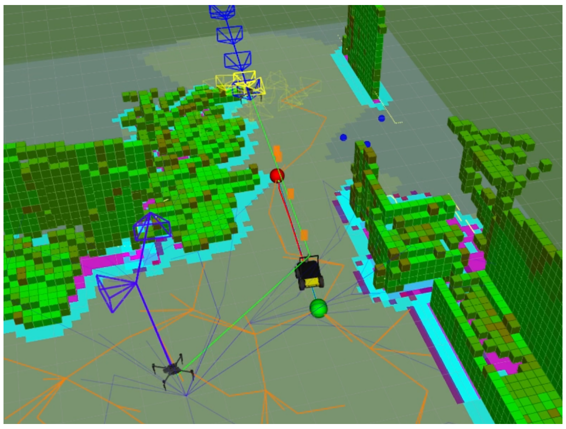
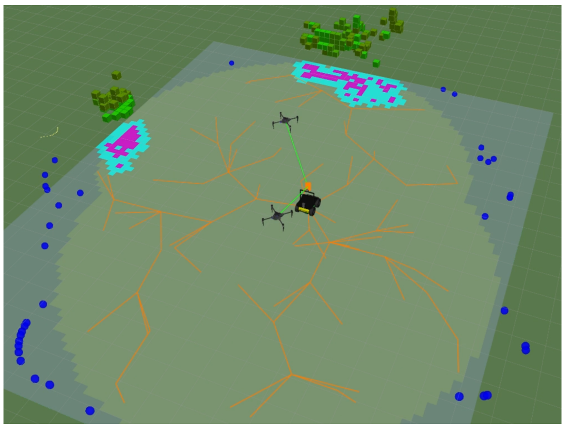

(b) Rviz visualization.

Fig. 5.27 Simulation of the exploration scenario on the "Disaster" map. (a) The Gazebo GUI allows the viewing of the robot system during in simulation. (b) The UGV exploring frontiers points, which are visualized as blue spheres. Frontier points are found using a modified RRT* algorithm, with the tree visualized in orange.


(a) A frontier tree is grown.

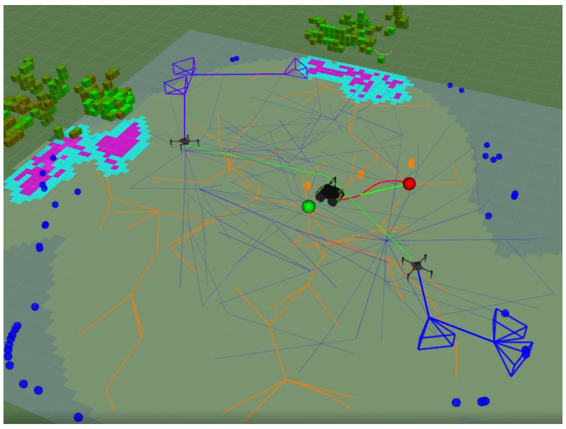
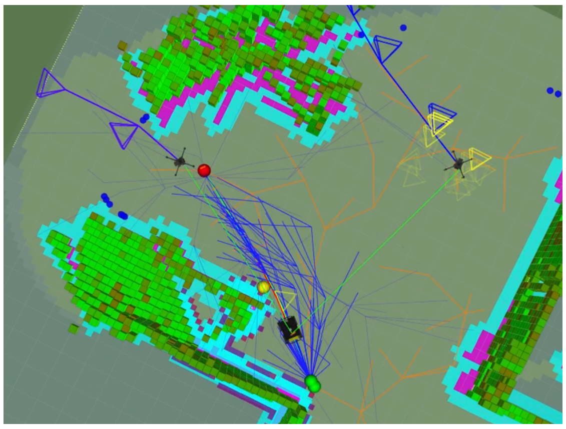

(b) A frontier is selected.

Fig. 5.28 Simulation of the exploration scenario on the "Outdoor" map, showing the growth of the initial frontier tree. (a) A frontier tree, visualized in orange, is grown from the current UGV position. Frontier points, visualized as blue spheres, are detected where tree edges cross from mapped to unmapped space. (b) A frontier point is selected and its parent vertex is chosen as the navigation goal, visualized as a red sphere. The UAVs make plans taking the new UAV path into account.


(a) Obstacle detected and replanning initiated.

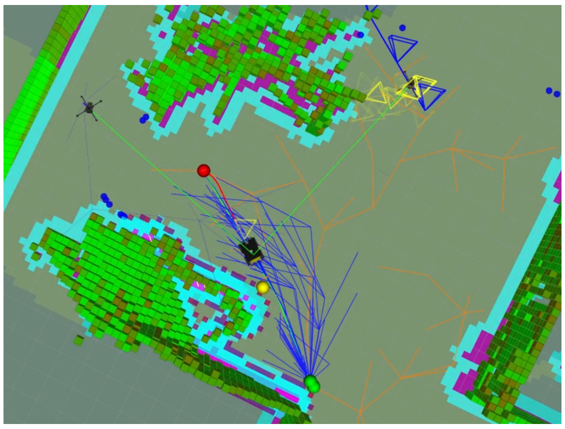
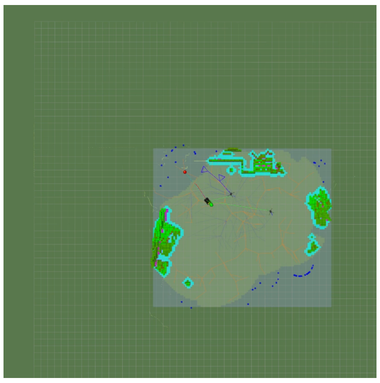
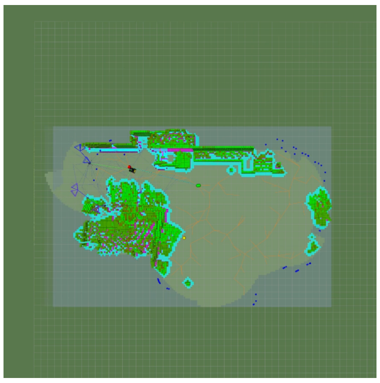

(b) UGV moves around obstacle.

Fig. 5.29 Simulation of the exploration scenario on the "Disaster" map, showing the global RRT planner navigating around obstacles. (a) The RRT global path planner is used to create a collision free path for the UGV. Planning is done using the Informed-RRT* algorithm, visualized in blue. (b) The UGV moves around the obstacle.


(a) Exploration 1.


(b) Exploration 2.

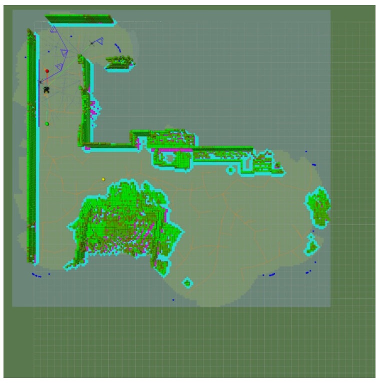

(c) Exploration 3.

Fig. 5.30 Simulation of the exploration scenario on the "Disaster" map, showing the gradual exploration of the environment by the UGV-UAV collaborative system.

Chapter 6

Conclusion

This chapter concludes the the project work presented in this thesis. First, a short summary of the thesis and project work will be given. Following this, a discussion is presented which addresses the results and how the results relate to the original problem formulation presented in Section 1.2. Possible improvements to the implemented systems will then be discussed. Finally, this chapter is concluded with a presentation of potential future work beyond the improvements presented in the preceding section.

6.1 Summary

The first chapter of this thesis introduced the problem formulation that was tackled in this thesis. This problem formulation was rooted in a potential scenario and use-case for a UGV-UAV cooperative team that was identified in the Specialization project [1]. Further motivation for tackling this problem was presented through a review of the existing systems within the field of UGV-UAV collaborative path planning and the identification of a gap in the literature.

Chapter 2 presented background theory and material that motivated the design choices taken when implementing the project systems. The chapter first presented general concepts within the field of robotic systems, such as motion control, planning and perception. Path planning concepts and algorithms for single-robot use cases were then presented, with a focus on Informative Path Planning (IPP). These concepts were then extended to the multi-robot case, with some added material within formation control and communication constrained planning also being presented.

Background material related to the software frameworks used for this project were presented in Chapter 3. This includes the Robot Operating System (ROS) framework and Gazebo simulator, both of which are central in this work. Following this, the systems for the UGV and UAV vehicle platforms were presented. An emphasis was placed on the motion- and path-planning capabilities of these systems, as these would be directly utilized by the implemented project systems. The chapter also summarized the most important systems implemented for the Specialization project, which includes single-robot exploration path planning systems.

The design and implementation of the project systems was presented in Chapter 4. The overall system architecture to coordinate a multi-UAV team around a UGV was first presented, with the presentation centering around the Finite-State Machine (FSM)s controlling UAV movement. The implemented Communication Constrained Multi-robot Informative Path Planning (CCMIPP)-systems were then presented.

First, the Collaborative exploration planner was presented as an extension to the View-based exploration planner implemented in the Specialization project. Communication constraint satisfaction and sensor coverage overlap had to be taken into account to enable collaborative exploration. Two variants of Formation-based planners were then presented; the Simple formation-bank planner and the Sampling-based Formation Reshaping (SFR) planner. Both planned UAV motion relative to the UGV, with the latter also making use of the systems implemented for the Collaborative exploration planner. Finally, two Combined planners were introduced which combined the functionality of the previously presented planners.

In the fifth chapter, simulation results from testing the implemented systems and planners in various scenarios were presented. The Total Information Gain (TIG), Information Gain Efficiency (IGE) and Communication Constraint Breakage (CCB) performance metrics were used to compare the implemented planners. Environments consisting of simple shapes were used to test the planners and gather results in three different scenarios involving a stationary UGV, a UGV moving along a path and a UGV tour consisting of path segments and waypoint stops. A final scenario was presented in two realistic environments to showcase the planners in an exploration use-case.

6.2 Discussion

The systems implemented in this project work were designed to coordinate UAV movement around a UGV, with the goal of the UAVs acting as mobile sensors for the UGV. This problem formulation was initially conceived prior to initiating the Specialization project work. During the course of the Specialization project, the focus shifted to implementing the simulation- and system framework which would eventually be used to implement and test the systems that could solve the initial problem formulation. The initial problem formulation was also modified to focus more on collaborative exploration rather than UAV coordination. In the conclusion of the Specialization project, several potential improvements to the systems and possible future works were presented. These will be addressed here.

For the UGV systems, the path planning and exploration algorithms used in the Specialization project were deemed to be robust, and have been utilized in this project work without major changes. The modifications to the planning algorithms to allow for path replanning and more intelligent usage of the exploration results were implemented. However, the focus in this project shifted away from collaborative exploration between the UGV and UAVs and back to the original goal. As a result of this, the planning and exploration systems of the UGV were not extended further beyond the aforementioned improvements.

As the project focus was shifted to UAV coordination, the potential improvements presented in the Specialization project regarding UAV behaviour had to be addressed. The UAV obstacle avoidance system was identified as a primary cause of issues for operations in the Specialization project. For this project it was replaced by a modified variant of the UGV path planner, as suggested in the Specialization project. Modifications to the View-based exploration planner were also made since the planner would play a central role in one of the implemented systems. This includes the inclusion of distance costs and sensor coverage overlap.

Beyond this, the framework itself was not extended far beyond its original functionality. An additional UAV was added, such that a total of three UAVs could be simulated alongside one UGV. Some modifications were made to improve general system and simulator performance, though these modifications were minor and will therefore not be elaborated further. The framework was also not extended to include other vehicle platforms beyond the UGV and UAV platforms.

The Specialization project presented two directions for future work. One direction was the extension of the framework into a higher level mission planner for extended and complex scenarios. Though some of the scenarios and monitoring systems implemented in this work bear resemblance to such a mission planner, any functionality beyond this would need to be addressed in future work. The other direction for future work was the implementation of a communication constrained planner to coordinate the UGV-UGV team to perform collaborative exploration, which has been addressed in this project work.

The problem formulation presented in Section 1.2 addresses the way in which this project work extends the work done in the Specialization project. Two research questions were posed, the first of which directly addresses the extension of the Specialization project systems into a system to solve the UGV-UAV CCMIPP problem.

The Collaborative exploration planner, which extends the functionality of the IPP systems made for the Specialization project, was in many cases shown to be superior to the other planning methods developed in this project work. Extending the system for this project's use-case involved changing both how the planner functioned and how the results were processed. To achieve the former, the planner was provided with details about the local UGV plans to inform sampling and planning. The results were then processed by giving the resulting paths new utility values, taking collaborative and communication constraint elements into account.

This method was given a lot of attention and tuning, with several modifications made to increase the planner's effectiveness. This includes the modified utility function to address the problem of UAVs operating under the Collaborative exploration planner losing LoS communication with the UGV. The effectiveness of the resulting Collaborative exploration planner is also possible due to the effectiveness and robustness of the original exploration algorithm it extended. As such, the answer to the first research question is positive. The Collaborative exploration planner, being an extension of a single-robot exploration planner, is able to solve the UGV-UAV CCMIPP problem in an effective manner.

The second research question was addressed by analyzing the performance of the Collaborative exploration planner against the other planners developed for this project. The Formation-based planners, consisting of a Simple formation-bank planner variant and a Sampling-based Formation Reshaping (SFR) planner variant, were developed explicitly to coordinate UAV movement around a stationary or moving UGV for the purpose of acting as mobile sensors.

The Simple formation-bank planner was shown to be surprisingly effective in the moving UGV case relative to it's simple design. Compared to the design and implementation of the Collaborative exploration planner, little effort went into the design and implementation of the

Simple Formation-bank planner. For the stationary UGV case, the pure Simple formation-bank planner was not effective. However, combining it with other planners, as was done in the Hybrid planner, alleviated this issue.

The SFR planner can in some ways be seen as combining principles from both the Collaborative exploration planner and the Simple formation-bank planner. Introducing sampling removed the need for the designer to specify the poses in the formation-bank, and allowed the planner itself to select and tune the chosen poses given the current situation and utility function. More effort can however be given to improving the performance of the planner, as multiple instances of erroneous behaviour could be observed.

Compared to the result of the Collaborative exploration planner, the Formation-based planners are shown to be moderately effective in the scenarios presented in Chapter 5. They do not behave as greedily as the Collaborative exploration planner, usually resulting in a higher IGE performance but a lower TIG performance. They are however much more robust to communication failure and have a lower amount of recorded CCBs. Answering the second research question is however made difficult by the fact that these planners were not given as much attention as the Collaborative exploration planner, making a direct comparison unfair in most cases.

It can also be discussed whether the presented scenarios are adequate in simulating the UGV-UAV CCMIPP problem, and whether the chosen performance metrics are good metrics to judge the systems by. The stationary, path and tour scenarios were used to test the planners and gather results. In these scenarios, the chosen metrics make sense. However, the planners were also tuned and modified to maximise the chosen metrics in these scenarios, which could affect system performance negatively if they were applied to other scenarios or with other performance metrics.

Though the case of collaborative UGV-UAV exploration was presented, it was not as heavily analyzed as the other scenarios. It was observed that some problems which had plagued the other scenarios were not present in the exploration scenario. This includes the issue of the UGV entering unmapped space and losing LoS communication with the UAVs, which had been a motivating factor in modifying some planner's behaviour.

Though several planning systems for solving the UGV-UAV CCMIPP problem were investigated in this thesis, these systems do not span the entire field of possible solutions to the presented problem. Sampling-based methods are central to all the solutions investigated in this project work. This is obvious for the Collaborative exploration and SFR planners, but the Simple-formation planner also makes use of sampling when checking whether poses in the formation bank are viable. The planners that make use of information gain also make use of the same method to determine the estimated information gain of potential poses.

As presented in both section 1.1 and chapter 2, similar problems can be solved using other methods. Many of the existing solutions involve optimization-based methods. These methods can in most cases make stricter claims on optimality and robustness compared to the sampling-based methods presented here. This direction was however not chosen for this project work, as the sampling-based methods presented were simpler to implement and test, and had already shown great promise in the previous work. Though optimization-based methods and other methods could be investigated, this is left as future work.

6.3 Improvements

Several avenues for further improvement were identified during the conclusion of this project work. This includes improvements to the framework for performance reasons and modifications of the implemented planning systems. Improvements or implementations of a larger scale are covered in Section 6.4.

Some shortcomings of the simulation framework implemented in the Specialization project were addressed and commented on in Section 6.2. However, the framework should still be improved further to optimize performance and increase usability. The Gazebo simulator is still the most computationally heavy component of the current system. Offloading this computation to another computer and implementing networked communication between the simulator and robot systems should be investigated to alleviate this issue. Usability can be improved by providing a better user interface, potentially by using the `rqt`-based tools identified in the Specialization project.

A central part of the problem revolves around planning under communication constraints. In this project work, communication constraint satisfaction was determined by systems operating in the same software framework as the planners. While the range communication constraint could be checked using ground truth positions of the vehicles, the LoS communication constraint was checked within the OctoMap framework, leading to the issue of determining whether or not unmapped space should be considered occupied. Moving the

LoS communication constraint checking to the simulator framework would provide a more correct way to determine whether the vehicles have lost LoS communication.

For the planning systems, a central feature of the UAV platform is its ability to operate in all three dimensions. In the current planning systems, UAVs are not allowed to move outside of their assigned operational altitudes. This was due to collisions between the UAVs that were observed to occur when full three dimensional planning was allowed, though this only happened in rare cases. Implementing collision avoidance between the UAVs, which can take into account communication of positions and plans between the UAVs, would allow the planners to plan freely in all three dimensions and potentially yield better plans.

Several issues and bugs were observed with the implemented planning systems which highlight areas of potential improvement. Some of these were covered in Chapter 5, while others will be mentioned here.

For the Collaborative exploration planner, one main issue was the rejection of plans that would prevent a loss of communication and subsequent recovery. This issue could be fixed by including communication constraint satisfaction into the utility of plans, such that longer plans that maintain communication are favored over shorter paths that would eventually break communication. Increasing the maximum path length for UAV plans could also enable the UAV to sample good view configurations on the other side of the mapped area which it could navigate to rather than maintaining the same position. This variation could also be extended to a split between a local and a global level of planning for information gain and communication constraint satisfaction respectively.

For the Formation-based planners, some issues were identified when maintaining a formation with a turning UGV. Removing the rotation of the UGV centered formation frame would fix this issue, though this would change the nature of the formation holding and could introduce other issues. If rotation should still be considered, a maximum rotation rate for the formation frame separate from the UGV orientation could be introduced.

The Simple Formation-bank planner would benefit from some additional attention and tuning with regards to the formations in the bank. Apart from adding additional poses, the individual UAVs could benefit from taking the actions of other UAVs into account when choosing a formation pose. This could come in the form of checking sensor coverage overlap, as is done in the other planning systems. Additional degrees of freedom, such as allowing the formation pose orientations to be chosen freely, could also be investigated.

The SFR planner, along with the Combined planners, were the last to be implemented and could see significant improvement with better selection of parameters and some general bug fixing. For the SFR planner, the design decision to sample and select entire formations rather than allowing individual UAVs to select formation poses lead to some suboptimal behaviour. One way to fix this would be to allow the UAVs to select their preferred formation pose in a sequence determined by priority or other means.

In general, this project work has focused on analyzing many different methods of solving the identified problem, rather than choosing one and improving one system. This has made it possible to compare the systems' performance with each other on the same scenarios, which has yielded insights into good directions to investigate in future works.

6.4 Future work

Beyond improving the implemented systems, several possible future works have been identified which could use the foundation built by this project work.

Perhaps the most obvious continuation of this project work would be to test the implemented systems on physical vehicle platforms. The Specialization project framework was designed with the intention of deploying the systems on physical vehicles, though no such experiments have been conducted. Such an extension would require systems for localization and motion control of the vehicle, which have been assumed to be functioning for this project work. Physical tests would also place higher requirements on the robustness and reliability of the systems for practical and safety reasons.

As a central part of this project work revolves around the communication constraints imposed by having UAVs act as mobile sensors for a UGV, more realistic communication bandwidth constraints should be investigated if the systems are applied to real robots. Some alternatives were presented in Section 2.3.4, which include probabilistic models that can be included in the utility functions of the planners. Other communication related factors, such as the propagation of information through robotic networks, would also have to be analyzed and tackled should the system be tested in real-world scenarios.

Several works within communication constrained planning have also demonstrated the usage of robots as communication relays to extend the range of operation. This would be especially relevant for scenarios with a stationary UGV and in obstructed environments where the LoS constraint severely limits the operational area. The increased bandwidth requirements on potential relay UAVs would also have to be accounted for. Additionally, online switching of robots to act as relays or explorers is an active field of research that could be investigated using the systems made in this project.

Within the space of sampling-based methods, other methods for solving the CCMIPP problem could be investigated. The PRM algorithm presented in Section 2.2 could be used to construct a communication constrained roadmap above the UGV which the UAVs could traverse. This would separate the problem into two parts: the constructions and maintenance of the roadmap for communication constraints and obstacle avoidance, and graph traversal of the roadmap by the UAVs. This could potentially simplify the problem, as potential UAV plans and states would be confined to the discrete edges and vertices of the roadmap graph.

More intelligent methods for negotiating plans between robots in multi-robot systems have also been presented. This work utilized priority-based sequential planning in the cases where multiple UAV plans had to be determined. Methods such as network consensus were presented in Section 2.3.1, while systems using optimization based methods such as integer linear programming are present in the literature. These methods could potentially increase the overall performance of the implemented systems.

Beyond the high-level path planning that this work has focused on, future works could investigate ways to integrate perception or motion control more actively in the planning. Though sensor characteristics play a role in the planning of informative paths, operation in GNSS-denied environments could make the role of active perception planning more important if range sensors are used to localize the robots. Metrics such as the amount of trackable visual features on a potential path could then be used to influence the plans of the UAVs. Including motion control of the UAVs beyond the simple navigation goals used in this project should also be investigated.

Finally, for this system to be viable for real applications, the implementation of a higher level mission planner should be investigated. This would give potential operators the ability to send goals to the system and monitor progress and actions made by the system. A mission planner would also need to account for auxiliary states and situations that the system could encounter in a realistic scenario, such as sensor- or communication failure, UAV battery recharge or shifting environmental conditions.

The potential future works that can be based on the work done in this project are not limited to the cases presented here. The author hopes that the foundation created by this project work, and which has been presented in this thesis, can inspire others to investigate and contribute to this exciting field of research.

References

[1] Kevin Meyer. Development of a Simulation Framework for Heterogeneous Robotic Systems. (June), 2020.

[2] Keiji Nagatani, Seiga Kiribayashi, Yoshito Okada, Kazuki Otake, Kazuya Yoshida, Satoshi Tadokoro, Takeshi Nishimura, Tomoaki Yoshida, Eiji Koyanagi, Mineo Fukushima, and Shinji Kawatsuma. Emergency response to the nuclear accident at the Fukushima Daiichi Nuclear Power Plants using mobile rescue robots. *Journal of Field Robotics*, 30(1):44–63, 1 2013.

[3] Elisabeth Menendez, Juan G. Victores, Roberto Montero, Santiago Martínez, and Carlos Balaguer. Tunnel structural inspection and assessment using an autonomous robotic system. *Automation in Construction*, 87:117–126, 3 2018.

[4] S. W. Squyres, A. H. Knoll, R. E. Arvidson, J. W. Ashley, J. F. Bell, W. M. Calvin, P. R. Christensen, B. C. Clark, B. A. Cohen, P. A. De Souza, L. Edgar, W. H. Farrand, I. Fleischer, R. Gellert, M. P. Golombek, J. Grant, J. Grotzinger, A. Hayes, K. E. Herkenhoff, J. R. Johnson, B. Jolliff, G. Klingelhöfer, A. Knudson, R. Li, T. J. McCoy, S. M. McLennan, D. W. Ming, D. W. Mittlefehldt, R. V. Morris, J. W. Rice, C. Schröder, R. J. Sullivan, A. Yen, and R. A. Yingst. Exploration of Victoria crater by the mars rover opportunity. *Science*, 324(5930):1058–1061, 5 2009.

[5] Sebastian d'Oleire Oltmanns, Irene Marzolff, Klaus Peter, and Johannes Ries. Unmanned Aerial Vehicle (UAV) for Monitoring Soil Erosion in Morocco. *Remote Sensing*, 4(11):3390–3416, 11 2012.

[6] J. A. Gonçalves and R. Henriques. UAV photogrammetry for topographic monitoring of coastal areas. *ISPRS Journal of Photogrammetry and Remote Sensing*, 104:101–111, 6 2015.

[7] Najib Metni and Tarek Hamel. A UAV for bridge inspection: Visual servoing control law with orientation limits. *Automation in Construction*, 17(1):3–10, 11 2007.

[8] Janosch Nikolic, Michael Burri, Joern Rehder, Stefan Leutenegger, Christoph Huerzeler, and Roland Siegwart. A UAV system for inspection of industrial facilities. In *IEEE Aerospace Conference Proceedings*, 2013.

[9] Quyen Vu, Mirko Raković, Vlado Delic, and Andrey Ronzhin. Trends in development of UAV-UGV cooperation approaches in precision agriculture. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11097 LNAI, pages 213–221. Springer Verlag, 9 2018.

[10] L. Cantelli, M. Mangiameli, C. D. Melita, and G. Muscato. UAV/UGV cooperation for surveying operations in humanitarian demining. In *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2013*, 2013.

[11] Connie Phan and Hugh H.T. Liu. A cooperative UAV/UGV platform for wildfire detection and fighting. In *2008 Asia Simulation Conference - 7th International Conference on System Simulation and Scientific Computing, ICSC 2008*, pages 494–498, 2008.

[12] Nathan Michael, Shaojie Shen, Kartik Mohta, Yash Mulgaonkar, Vijay Kumar, Keiji Nagatani, Yoshito Okada, Seiga Kiribayashi, Kazuki Otake, Kazuya Yoshida, Kazunori Ohno, Eijiro Takeuchi, and Satoshi Tadokoro. Collaborative mapping of an earthquake-damaged building via ground and aerial robots. *Journal of Field Robotics*, 29(5):832–841, 9 2012.

[13] Steven L. Waslander. Unmanned Aerial and Ground Vehicle Teams: Recent Work and Open Problems. In *Intelligent Systems, Control and Automation: Science and Engineering*, volume 65, pages 21–36. Springer Netherlands, 2013.

[14] Donald K. Macarthur and Carl D. Crane. Unmanned ground vehicle state estimation using an unmanned air vehicle. In *Proceedings of the 2007 IEEE International Symposium on Computational Intelligence in Robotics and Automation, CIRA 2007*, pages 473–478, 2007.

[15] Tianguang Zhang, Wei Li, Markus Achtelik, Kolja Kühnlenz, and Martin Buss. Multi-Sensory Motion Estimation and Control of a Mini-Quadrotor in an Air-Ground Multi-Robot System. pages 45–50, 2010.

[16] El Houssein Chouaib Harik, François Guérin, Frederic Guinand, Jean-François Brethé, and Hervé Pelvillain. UAV-UGV Cooperation For Objects Transportation In An Industrial Area. In *Proceedings of the IEEE International Conference on Industrial Technology*, volume 2015, pages 547–552, 2015.

[17] Chenzui Li, Qinyuan Ren, Fei Chen, and Ping Li. Vision-Based Formation Control of a Heterogeneous Unmanned System. In *IECON Proceedings (Industrial Electronics Conference)*, volume 2019-Octob, pages 5299–5304. IEEE Computer Society, 10 2019.

[18] Boris Sofman, J Bagnell, Anthony Stentz, and Nicolas Vandapel. Terrain Classification from Aerial Data to Support Ground Vehicle Navigation. 1 2006.

[19] Jeffrey Delmerico, Alessandro Giusti, Elias Mueggler, Luca Maria Gambardella, and Davide Scaramuzza. “On-the-Spot Training” for Terrain Classification in Autonomous Air-Ground Collaborative Teams. pages 574–585. 2017.

[20] Herbert G. Tanner. Switched UAV-UGV cooperation scheme for target detection. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3457–3462, 2007.

[21] H. G. Tanner and D. K. Christodoulakis. Cooperation between aerial and ground vehicle groups for reconnaissance missions. In *Proceedings of the IEEE Conference on Decision and Control*, pages 5918–5923. Institute of Electrical and Electronics Engineers Inc., 2006.

[22] Ben Grocholsky, James Keller, Vijay Kumar, and George Pappas. Cooperative air and ground surveillance. *IEEE Robotics and Automation Magazine*, 13(3):16–26, 9 2006.

[23] Martin Saska, Tomáš Krajník, and Libor Přeučil. Cooperative μuAV-UGV autonomous indoor surveillance. In *International Multi-Conference on Systems, Signals and Devices, SSD 2012 - Summary Proceedings*, 2012.

[24] Ayush Dewan, Aravindh Mahendran, Nikhil Soni, and K. Madhava Krishna. Heterogeneous UGV-MAV exploration using integer programming. In *IEEE International Conference on Intelligent Robots and Systems*, pages 5742–5749, 2013.

[25] Elias Mueggler, Matthias Faessler, Flavio Fontana, and Davide Scaramuzza. Aerial-guided navigation of a ground robot among movable obstacles. In *12th IEEE International Symposium on Safety, Security and Rescue Robotics, SSRR 2014 - Symposium Proceedings*. Institute of Electrical and Electronics Engineers Inc., 1 2014.

[26] Howie Choset. Coverage of Known Spaces: The Boustrophedon Cellular Decomposition. *Autonomous Robots*, 9(3):247–253, 2000.

[27] Jeffrey Delmerico, Elias Mueggler, Julia Nitsch, and Davide Scaramuzza. Active autonomous aerial exploration for ground robot path planning. *IEEE Robotics and Automation Letters*, 2(2):664–671, 4 2017.

[28] Luqi Wang, Fei Gao, Fengyu Cai, and Shaojie Shen. CRASH: A Collaborative Aerial-Ground Exploration System Using Hybrid-Frontier Method. In *2018 IEEE International Conference on Robotics and Biomimetics, ROBIO 2018*, pages 2259–2266. Institute of Electrical and Electronics Engineers Inc., 7 2018.

[29] Hailong Qin, Zehui Meng, Wei Meng, Xudong Chen, Hao Sun, Feng Lin, and Marcelo H. Ang. Autonomous Exploration and Mapping System Using Heterogeneous UAVs and UGVs in GPS-Denied Environments. *IEEE Transactions on Vehicular Technology*, 68(2):1339–1350, 2 2019.

[30] Luciano Cantelli, M Presti, Michele Mangiameli, Carmelo Melita, and Giovanni Muscato. Autonomous Cooperation between UAV and UGV to improve navigation and environmental monitoring in rough environments. 2013.

[31] Shannon Hood, Kelly Benson, Patrick Hamod, Daniel Madison, Jason M. O'Kane, and Ioannis Rekleitis. Bird's eye view: Cooperative exploration by UGV and UAV. In *2017 International Conference on Unmanned Aircraft Systems, ICUAS 2017*, pages 247–255. Institute of Electrical and Electronics Engineers Inc., 7 2017.

[32] Lukas Klodt, Saman Khodaverdian, and Volker Willert. Motion control for UAV-UGV cooperation with visibility constraint. In *2015 IEEE Conference on Control and Applications, CCA 2015 - Proceedings*, pages 1379–1385. Institute of Electrical and Electronics Engineers Inc., 11 2015.

[33] Barry Gilhuly and Stephen L. Smith. Robotic Coverage for Continuous Mapping Ahead of a Moving Vehicle. *Proceedings of the IEEE Conference on Decision and Control*, 2019-Decem:8224–8229, 9 2019.

[34] Seiga Kiribayashi, Jun Ashizawa, and Keiji Nagatani. Modeling and design of tether powered multicopter. In *SSRR 2015 - 2015 IEEE International Symposium on Safety, Security, and Rescue Robotics*. Institute of Electrical and Electronics Engineers Inc., 3 2016.

[35] Takahiro Miki, Petr Khrapchenkov, and Koichi Hori. UAV/UGV Autonomous Cooperation: UAV Assists UGV to Climb a Cliff by Attaching a Tether. *Proceedings - IEEE International Conference on Robotics and Automation*, 2019-May:8041–8047, 3 2019.

[36] Christos Papachristos and Anthony Tzes. The power-tethered UAV-UGV team: A collaborative strategy for navigation in partially-mapped environments. In *2014 22nd Mediterranean Conference on Control and Automation, MED 2014*, pages 1153–1158. Institute of Electrical and Electronics Engineers Inc., 11 2014.

[37] Xuesu Xiao, Jan Dufek, and Robin R. Murphy. Autonomous Visual Assistance for Robot Operations Using a Tethered UAV. *arXiv*, 3 2019.

[38] Kurt Konolige and Andreas Nüchter. Range sensing. In *Springer Handbook of Robotics*, pages 783–810. Springer International Publishing, 1 2016.

[39] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering, Transactions of the ASME*, 82(1):35–45, 1960.

[40] Cyrill Stachniss, John J. Leonard, and Sebastian Thrun. Simultaneous localization and mapping. In *Springer Handbook of Robotics*, pages 1153–1175. Springer International Publishing, 1 2016.

[41] Alberto Elfes. Using Occupancy Grids for Mobile Robot Perception and Navigation. *Computer*, 22(6):46–57, 1989.

[42] Wolfram Burgard, Martial Hebert, and Maren Bennewitz. World modeling. In *Springer Handbook of Robotics*, pages 1135–1152. Springer International Publishing, 1 2016.

[43] Stuart Russell and Peter Norvig. *AI a modern approach*, volume 2. 2009.

[44] Richard Balogh and David Obdržálek. Using Finite State Machines in Introductory Robotics. In *Advances in Intelligent Systems and Computing*, volume 829, pages 85–91. Springer Verlag, 2019.

[45] H Goldstein, C Poole, and J Safko. *Classical Mechanics [Textbook]*. Pearson, 3rd editio edition, 2002.

[46] L. E. Dubins. On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. *American Journal of Mathematics*, 79(3):497, 7 1957.

[47] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

[48] Robert Mahony, Randal W Beard, and Vijay Kumar. Modeling and Control of Aerial Robots. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 1307–1334. Springer International Publishing, Cham, 2016.

[49] Lydia E. Kavraki and Steven M. LaValle. Motion planning. In *Springer Handbook of Robotics*, pages 139–161. Springer International Publishing, 1 2016.

[50] Steven M. LaValle. *Planning algorithms*, volume 9780521862. Cambridge University Press, 1 2006.

[51] John H. Reif. Complexity of the mover's problem and generalizations. In *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 421–427. IEEE Computer Society, 1979.

[52] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 12 1959.

[53] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[54] Steven M. Lavalle. Rapidly-Exploring Random Trees: A New Tool for Path Planning. 1998.

[55] Sertac Karaman, Matthew R. Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the RRT. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1478–1483, 2011.

[56] Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. Informed RRT*: Optimal Sampling-based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic. *IEEE International Conference on Intelligent Robots and Systems*, pages 2997–3004, 4 2014.

[57] Geoffrey A. Hollinger and Gaurav S. Sukhatme. Sampling-based robotic information gathering algorithms. *The International Journal of Robotics Research*, 33(9):1271–1287, 8 2014.

[58] Amarjeet Singh, Andreas Krause, Carlos Guestrin, and William J. Kaiser. Efficient Informative Sensing using Multiple Robots. *Journal of Artificial Intelligence Research*, 34:707–755, 4 2009.

[59] Brian Yamauchi. Frontier-based approach for autonomous exploration. In *Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation, CIRA*, pages 146–151. IEEE, 1997.

[60] Sebastian Thrun. Robotic Mapping: A Survey. *Exploring Artificial Intelligence in the New Millennium*, 2002.

[61] C. I. Connolly. The determination of next best views. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 432–435. Institute of Electrical and Electronics Engineers Inc., 1985.

[62] Hartmut Surmann, Andreas Nüchter, and Joachim Hertzberg. An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments. *Robotics and Autonomous Systems*, 45(3-4):181–198, 12 2003.

[63] Stefan Isler, Reza Sabzevari, Jeffrey Delmerico, and Davide Scaramuzza. An Information Gain Formulation for Active Volumetric 3D Reconstruction. *Proceedings - IEEE International Conference on Robotics and Automation*, 6 2016.

[64] Hassan Umari and Shayok Mukhopadhyay. Autonomous robotic exploration based on multiple rapidly-exploring randomized trees. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2017-September, pages 1396–1402. IEEE, 9 2017.

[65] Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. Receding horizon next-best-view planner for 3D exploration. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2016-June, pages 1462–1468. Institute of Electrical and Electronics Engineers Inc., 6 2016.

[66] Giuseppe Oriolo, Marilena Vendittelli, Luigi Freda, and Giulio Troso. The SRT method: Randomized strategies for exploration. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2004, pages 4688–4694. Institute of Electrical and Electronics Engineers Inc., 2004.

[67] Benjamín Tovar, Luis Guilamo, and Steven M. LaValle. Gap navigation trees: Minimal representation for visibility-based tasks. *Springer Tracts in Advanced Robotics*, 17:425–440, 2005.

[68] Michael Erdmann and Tomás Lozano-Pérez. On multiple moving objects. *Algorithmica*, 2(1-4):477–521, 11 1987.

[69] Maren Bennewitz, Wolfram Burgard, and Sebastian Thrun. Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots. In *Robotics and Autonomous Systems*, volume 41, pages 89–99. North-Holland, 11 2002.

[70] Jur P. Van Den Berg and Mark H. Overmars. Prioritized motion planning for multiple robots. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 2217–2222, 2005.

[71] Lynne E. Parker, Daniela Rus, and Gaurav S. Sukhatme. Multiple mobile robot systems. In *Springer Handbook of Robotics*, pages 1335–1379. Springer International Publishing, 1 2016.

[72] Ke Cheng and Prithviraj Dasgupta. Weighted voting game based multi-robot team formation for distributed area coverage. In *ACM International Conference Proceeding Series*, pages 9–15, 2010.

[73] Heng Wei, Qiang Lv, Nanxun Duo, GuoSheng Wang, and Bing Liang. Consensus Algorithms Based Multi-Robot Formation Control under Noise and Time Delay Conditions. *Applied Sciences*, 9(5):1004, 3 2019.

[74] Brian Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the International Conference on Autonomous Agents*, pages 47–53, 1998.

[75] Wolfram Burgard, Mark Moors, Dieter Fox, Reid Simmons, and Sebastian Thrun. Collaborative multi-robot exploration. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 1, pages 476–481, 2000.

[76] Robert Zlot, Anthony Tony Stentz, M. Bernardine Dias, and Scott Thayer. Multi-robot exploration controlled by a market economy. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 3, pages 3016–3023, 2002.

[77] Wolfram Burgard, Mark Moors, Cyrill Stachniss, and Frank E. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386, 6 2005.

[78] J A Hartigan and M A Wong. Algorithm AS 136: A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1 1979.

[79] Agusti Solanas and Miguel Angel Garcia. Coordinated multi-robot exploration through unsupervised clustering of unknown space. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 717–721, 2004.

[80] Kai M. Wurm, Cyrill Stachniss, and Wolfram Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 1160–1165, 2008.

[81] Antonio Franchi, Luigi Freda, Giuseppe Oriolo, and Marilena Vendittelli. A randomized strategy for cooperative robot exploration. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 768–774, 2007.

[82] Antonio Franchi, Luigi Freda, Giuseppe Oriolo, and Marilena Vendittelli. The sensor-based random graph method for cooperative robot exploration. *IEEE/ASME Transactions on Mechatronics*, 14(2):163–175, 2009.

[83] Sotiris Papatheodorou and Anthony Tzes. Theoretical and Experimental Collaborative Area Coverage Schemes Using Mobile Agents. In *Applications of Mobile Robots*. IntechOpen, 3 2019.

[84] Mazda Ahmadi and Peter Stone. Continuous area sweeping: A task definition and initial approach. In *2005 International Conference on Advanced Robotics, ICAR '05, Proceedings*, volume 2005, pages 316–323, 2005.

[85] Stephen L. Smith, Mac Schwager, and Daniela Rus. Persistent robotic tasks: Monitoring and sweeping in changing environments. *IEEE Transactions on Robotics*, 28(2):410–426, 4 2012.

[86] Mac Schwager, Brian J. Julian, Michael Angermann, and Daniela Rus. Eyes in the sky: Decentralized control for the deployment of robotic camera networks. In *Proceedings of the IEEE*, volume 99, pages 1541–1561. Institute of Electrical and Electronics Engineers Inc., 2011.

[87] Sotiris Papatheodorou, Anthony Tzes, and Yiannis Stergiopoulos. Collaborative visual area coverage. *Robotics and Autonomous Systems*, 92:126–138, 6 2017.

[88] Mac Schwager, Brian J. Julian, and Daniela Rus. Optimal coverage for multiple hovering robots with downward facing cameras. pages 3515–3522. Institute of Electrical and Electronics Engineers (IEEE), 8 2009.

[89] Nikolaos Bousias, Sotiris Papatheodorou, Mariliza Tzes, and Anthony Tzes. Distributed surveillance by a swarm of UAVs operating under positional uncertainty. 10 2019.

[90] T. D. Barfoot and C. M. Clark. Motion planning for formations of mobile robots. *Robotics and Autonomous Systems*, 46(2):65–78, 2 2004.

[91] Kazuo Sugihara and Ichiro Suzuki. Distributed motion coordination of multiple mobile robots. pages 138–143. Publ by IEEE, 1990.

[92] Ivar André F. Ihle, Murat Arcak, and Thor I. Fossen. Passivity-based designs for synchronized path-following. *Automatica*, 43(9):1508–1518, 9 2007.

[93] Athanasios Krontiris, Sushil Louis, and Kostas E. Bekris. Multi-level formation roadmaps for collision-free dynamic shape changes with non-holonomic teams. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1570–1575. Institute of Electrical and Electronics Engineers Inc., 2012.

[94] Daniel Mellinger, Alex Kushleyev, and Vijay Kumar. Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 477–483. Institute of Electrical and Electronics Engineers Inc., 2012.

[95] Fredrik Baberg and Petter Ogren. Formation obstacle avoidance using RRT and constraint based programming. In *SSRR 2017 - 15th IEEE International Symposium on Safety, Security and Rescue Robotics, Conference*, pages 1–6. Institute of Electrical and Electronics Engineers Inc., 10 2017.

[96] Hou Sheng Su, Jin Xin Zhang, and Zhi Gang Zeng. Formation-containment control of multi-robot systems under a stochastic sampling mechanism. *Science China Technological Sciences*, 63(6):1025–1034, 6 2020.

[97] Robin Deits and Russ Tedrake. Computing large convex regions of obstacle-free space through semidefinite programming. In *Springer Tracts in Advanced Robotics*, volume 107, pages 109–124. Springer Verlag, 2015.

[98] Javier Alonso-Mora, Eduardo Montijano, Tobias Nägeli, Otmar Hilliges, Mac Schwager, and Daniela Rus. Distributed multi-robot formation control in dynamic environments. *Autonomous Robots*, 43(5):1079–1100, 6 2019.

[99] Javier Alonso-Mora, Stuart Baker, and Daniela Rus. Multi-robot formation control and object transport in dynamic environments via constrained optimization. *International Journal of Robotics Research*, 36(9):1000–1021, 8 2017.

[100] Miguel Juliá, Arturo Gil, and Oscar Reinoso. A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots*, 33(4):427–444, 11 2012.

[101] Francesco Amigoni, Jacopo Banfi, and Nicola Basilico. Multirobot Exploration of Communication-Restricted Environments: A Survey. *IEEE Intelligent Systems*, 32(6):48–57, 11 2017.

[102] Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proceedings - IEEE INFOCOM*, volume 2, pages 775–784. IEEE, 2000.

[103] Arnoud Visser and Bayu A. Slamet. Including communication success in the estimation of information gain for multi-robot exploration. In *Proceedings of the 6th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, WiOpt 2008*, pages 680–687, 2008.

[104] Victor Spirin and Stephen Cameron. Rendezvous through obstacles in multi-agent exploration. In *12th IEEE International Symposium on Safety, Security and Rescue Robotics, SSRR 2014 - Symposium Proceedings*. Institute of Electrical and Electronics Engineers Inc., 1 2014.

[105] Jacopo Banfi, Alberto Quattrini Li, Nicola Basilico, Ioannis Rekleitis, and Francesco Amigoni. Asynchronous multirobot exploration under recurrent connectivity constraints. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2016-June, pages 5491–5498. Institute of Electrical and Electronics Engineers Inc., 6 2016.

[106] Geoffrey A. Hollinger and Sanjiv Singh. Multirobot coordination with periodic connectivity: Theory and experiments. *IEEE Transactions on Robotics*, 28(4):967–973, 2012.

[107] Ronald C. Arkin and Jonathan Diaz. Line-of-sight constrained exploration for reactive multiagent robotic teams. In *International Workshop on Advanced Motion Control, AMC*, pages 455–461, 2002.

[108] Martijn N. Rooker and Andreas Birk. Multi-robot exploration under the constraints of wireless networking. *Control Engineering Practice*, 15(4):435–445, 3 2007.

[109] Piyoosh Mukhija, K. Madhava Krishna, and Vamshi Krishna. A two phase recursive tree propagation based multi-robotic exploration framework with fixed base station constraint. In *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, pages 4806–4811, 2010.

[110] Ethan Stump, Nathan Michael, Vijay Kumar, and Volkan Isler. Visibility-based deployment of robot formations for communication maintenance. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4498–4505, 2011.

[111] Francesco Amigoni, Jacopo Banfi, Alessandro Longoni, and Matteo Luperto. Online switch of communication modalities for efficient multirobot exploration. In *2017 European Conference on Mobile Robots, ECMR 2017*. Institute of Electrical and Electronics Engineers Inc., 11 2017.

[112] Tully Foote. tf: The transform library. In *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*, pages 1–6. IEEE, 4 2013.

[113] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: an efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 4 2013.

[114] Thomas Moore and Daniel Stouch. A generalized extended Kalman filter implementation for the robot operating system. In *Advances in Intelligent Systems and Computing*, volume 302, pages 335–348. Springer Verlag, 2016.

[115] Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian Gerkey, and Kurt Konolige. The office marathon: Robust navigation in an indoor office environment. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 300–307, 2010.

[116] Christoph Rösmann, Frank Hoffmann, and Torsten Bertram. Kinodynamic trajectory optimization and control for car-like robots. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2017-Septe, pages 5681–5686. Institute of Electrical and Electronics Engineers Inc., 12 2017.

[117] Diego Assencio. The intersection area of two circles (Published 2017.07.12). https://diego.assencio.com/?index=8d6ca3d82151bad815f78addf9b5c1c6, 2017.

[118] Zhi Yan, Luc Fabresse, Jannik Laval, and Noury Bouraqadi. Metrics for performance benchmarking of multi-robot exploration. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2015-Decem, pages 3407–3414. Institute of Electrical and Electronics Engineers Inc., 12 2015.

Appendix A

Algorithms

Algorithm 1: Sampling-Based Roadmap

```
Data:
N: Number of nodes to include in roadmap
Result:
G: Roadmap
1  G.init();
2  $i \leftarrow 0$;
3  while $i < N$ do
4      $\alpha(i) \leftarrow$ RANDOM_CONFIGURATION();
5      if $\alpha(i) \in \mathcal{C}_{free}$ then
6          G.ADD_VERTEX($\alpha(i)$);
7          $i \leftarrow i+1$;
8          for $q \in$ NEIGHBORHOOD$(\alpha(i),$G$)$ do
9              if CONNECT$(\alpha(i),q)$ then
10                 G.ADD_EDGE$(\alpha(i),q)$;
```

Algorithm 2: Rapidly-exploring Random Tree

```
Data:
$q_{init}$: Initial/root configuration
k: Number of exploration steps
$\Delta t$: Fixed time interval for integration
Result:
T: Tree
1 T.init($q_{init}$);
2 for $i \leftarrow 0$ to $k$ do
3     $\alpha(i) \leftarrow$ RANDOM_CONFIGURATION();
4     T $\leftarrow$ EXTEND$_{RRT}$(T, $\alpha(i)$, $\Delta t$);
```

Algorithm 3: EXTEND$_{RRT}$

```
Data:
T: Initial tree
$\alpha(i)$: Sampled configuration
$\Delta t$: Fixed time interval for integration
Result:
T': Extended tree
1 T' $\leftarrow$ T;
2 $q_{nearest} \leftarrow$ NEAREST_NEIGHBOR($\alpha(i)$, T');
3 $u \leftarrow$ SELECT_INPUT($q_{nearest}$, $\alpha(i)$);
4 $q_{new} \leftarrow$ NEW_STATE($q_{nearest}$, $u$, $\Delta t$);
5 if $q_{new} \in \mathcal{C}_{free}$ AND PATH($q_{nearest}$, $u$, $\Delta t$) $\in \mathcal{C}_{free}$ then
6     T'.ADD_VERTEX($q_{new}$);
7     T'.ADD_EDGE($q_{nearest}$, $q_{new}$, $u$);
8 return T';
```

Algorithm 4: EXTEND$_{RRT*}$

```
Data:
T: Initial tree
α(i): Sampled configuration
Δt: Fixed time interval for integration
Result:
T': Extended tree
1  T' ← T;
2  $q_{nearest}$ ← NEAREST_NEIGHBOR($\alpha(i)$, T');
3  $q_{new}$ ← STEER($q_{nearest}, \alpha(i)$);
4  if $q_{new} \in \mathcal{C}_{free}$ AND PATH($q_{nearest}, q_{new}$) $\in \mathcal{C}_{free}$ then
5      T'.ADD_VERTEX($q_{new}$);
6      $q_{min} \leftarrow q_{nearest}$;
7      $Q_{near}$ ← NEAR($T, q_{new}$);
8      for $q_{near} \in Q_{near}$ do
9          if PATH($q_{near}, q_{new}$) $\in \mathcal{C}_{free}$ then
10             $c'$ ← COST($q_{near}$) + DISTANCE($q_{near}, q_{new}$);
11             if $c'$ < COST($q_{new}$) then
12                 $q_{min} \leftarrow q_{near}$;
13     T'.ADD_EDGE($q_{min}, q_{new}$);
14     for $q_{near} \in Q_{near} \setminus \{q_{min}\}$ do
15         if PATH($q_{near}, q_{new}$) $\in$
           $\mathcal{C}_{free}$ AND COST($q_{near}$) >
           COST($q_{new}$) + DISTANCE($q_{near}, q_{new}$) then
16             $q_{parent}$ ← PARENT($q_{near}$);
17             T'.REMOVE_EDGE($q_{parent}, q_{near}$);
18             T'.ADD_EDGE($q_{new}, q_{near}$);
19 return T';
```

Algorithm 5: RANDOM_CONFIGURATION$_{INFORMED}$

```
Data:
$q_{init}$: Initial configuration
$q_{goal}$: Goal configuration
Result:
$\alpha(i)$': Informed random configuration
if COST($q_{goal}$) $< \infty$ then
    $c_{min} \leftarrow ||q_{goal} - q_{init}||_2$;
    $q_{center} \leftarrow (q_{goal} + q_{init})/2$;
    $\mathbf{C} \leftarrow$ ROTATION_TO_WORLD_FRAME$(q_{init}, q_{goal})$;
    $r_1 \leftarrow$ COST$(q_{goal})/2$;
    $\{r_i\}_{i=2,\ldots,n} \leftarrow \left(\sqrt{c_{max}^2 - c_{min}^2}\right)/2$;
    $\mathbf{L} \leftarrow$ DIAG$(r_1, r_2, \ldots, r_n)$;
    $q_{ball} \leftarrow$ SAMPLE_UNIT_BALL();
    $\alpha(i) \leftarrow \mathbf{CL}q_{ball} + q_{center}$;
else
    $\alpha(i) \leftarrow$ RANDOM_CONFIGURATION();
return $\alpha(i)$;
```

Algorithm 6: EXTEND$_{ViewBasedRRT}$

```
Data:
T: Initial tree
$\alpha(i)$: Sampled configuration
$d_{max}$: Maximum line-path distance
$\theta_{max}$: Maximum yaw-angle change
Result:
T': Extended tree
1  T' $\leftarrow$ T;
2  n$_{nearest}$ $\leftarrow$ NEAREST_NEIGHBOR$(\alpha(i)$, T'$)$;
3  $q_{new}$ $\leftarrow$ LINE_PATH(n$_{nearest}$, $\alpha(i), d_{max}, \theta_{max})$;
4  if COLLISION_FREE(q$_{new}$) AND CAST_RAY(n$_{nearest}$, $q_{new}) \in \mathcal{C}_{free}$
    then
5      n$_{min}$ $\leftarrow$ n$_{nearest}$;
6      $N_{near}$ $\leftarrow$ NEAR(T, $q_{new}$);
7      for $n_{near} \in N_{near}$ do
8          if PATH(n$_{near}$, $q_{new}) \in \mathcal{C}_{free}$ then
9              $c'$ $\leftarrow$ COST(n$(_{near}) +$ DISTANCE(n$_{near}$, $q_{new}$);
10             if $c' <$ COST$(q_{new})$ then
11                 n$_{min}$ $\leftarrow$ n$_{near}$;
12     n$_{parent}$ $\leftarrow$ n$_{min}$;
13     $c_{new}$ $\leftarrow$ COST(n$_{parent}$) + DISTANCE(n$_{parent}$, $q_{new}$);
14     $g_{new}$ $\leftarrow$ GAIN(n$_{parent}$) + $\frac{1}{\text{RANK}(\text{n}_{parent})+1}$GAIN($q_{new}$);
15     n$_{new}$ $\leftarrow$ CREATE_NODE$_{RRT}$(T',n$_{parent}$, $q_{new}, c_{new}, g_{new}$);
16     T'.ADD_NODE(n$_{new}$);
17 return T';
```

Appendix B

Parameter values

Default parameter values that are in use unless where stated otherwise. Parameter values are chosen by the author. Certain parameters have a relationship to other parameters which influence the choice of the parameter value. Other less explicit motivations for the parameter values are described in either Chapter 4 or Chapter 5. Some values are approximated since their corresponding parameters are not directly accessible to the author.

Symbol	Description	Relationship	Value	Unit
v_{ugv}	Max. UGV velocity	-	0.5	m/s
n_{ugv_wp}	UGV local waypoint amount	-	5	-
l_{ugv_wp}	UGV local waypoint spacing	$\geq v_{ugv} \cdot t_{ugv\_wp}$	2.0	m
t_{ugv_wp}	Min. UGV local waypoint time	-	4.0	s
v_{uav}	Max. UAV velocity	-	≈ 1.5	m/s
h_{uav_min}	Min. UAV operation altitude	-	1.5	m
h_{uav_int}	UAV operation altitude interval	-	0.5	m
r_{rrt_samp}	Sampling radius for UAV RRT* planning	$\leq l_{com}$	10.0	m
t_{rrt_samp}	Sampling time for UAV RRT* planning	-	1.0	s
l_{uav_rrt}	Max. edge distance for UAV RRT* planning	$\leq v_{uav} \cdot t_{ugv\_wp}$	4.0	m
rnk_{rrt_v}	Max. vertex rank for UAV RRT* planning	$< n_{ugv\_wp}$	4	-
l_{com}	Communication range	-	10.0	m
t_{com}	Communication timeout	-	1.0	s
f_{ugv_pl}	UGV planner loop frequency	-	5.0	Hz

f_{uav_pl}	UAV planner loop frequency	-	10.0	Hz
f_{uav_sfr}	Formation reshaper loop frequency	-	1.0	Hz
l_{uav_cov}	UAV circular coverage distance	-	6.0	m
r_{uav_cov}	UAV circular coverage radius	-	5.0	m
ψ_u	Standard angular unit	-	90.0	deg
d_{mult_max}	Max. distance threshold for way-point multiplier	-	4.0	m
d_{mult_min}	Min. distance threshold for way-point multiplier	-	1.0	m
c_{mult}	Max. multiplier value for waypoint multiplier	-	2.0	-
c_{info}	Information gain cost parameter	-	100.0	-
c_{euc}	Euclidean distance cost parameter	-	-1.0	-
c_{ang}	Angular distance cost parameter	-	-1.0	-
n_{form_bank}	Amount of formations in formation bank	-	7	-
n_{sfr_bank}	Amount of sampled formations in SFR bank	-	20	-
n_{sfr_keep}	Amount of sampled formations kept in SFR bank	-	5	-
r_{sfr_samp}	Sampling radius of SFR planner	-	1.0	m
ψ_{sfr_samp}	Max. sample yaw deviation of SFR planner	-	60.0	deg
t_{sfr_samp}	SFR planner sampling timeout	-	1.0	s
l_{hybrid}	Hybrid planner switch distance	-	10.0	m
$res_{octomap}$	OctoMap voxel resolution	-	0.25	m
$f_{octomap}$	OctoMap update frequency	-	1.0	Hz

Appendix C

Local navigation waypoint multiplier

In section 5.4.4, the observation is made that the Collaborative view-based exploration planner (presented in section 4.2) would frequently trigger recovery behaviour due to not mapping the area directly around the UGV. More precisely, when the UGV and its local navigation waypoints would enter unmapped space, LoS communication between the UGV and UAVs would be perceived as being lost. This is due to the conservative assumption that unmapped space is equivalent to occupied space when the Collaborative exploration planner is determining whether or not a plan is viable.

To attempt to fix this issue, a multiplier was added to the estimated information gain tied to the area around the UGV local navigation waypoints. The idea motivating this is that assigning more value to the information gained from unmapped areas around the UGV would drive the exploration planner to prefer exploring these area. Distance thresholds would apply, such that only unmapped areas within a certain radius of the UGV local navigation waypoints would add a multiplier. This multiplier function was then added to the normal information gain function presented in Equation (4.3). This function is repeated here, with the addition of the waypoint multiplier:

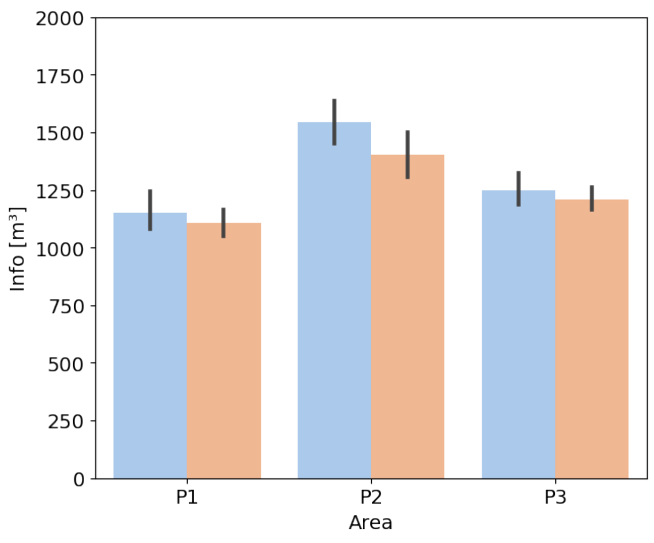
Given a view configuration q and corresponding sensor coverage circle $C(q)$, other sensor coverage circles (internal and external) C_{other}, the ratio rat_{unmp} of rays cast within the FOV which intersect with unmapped voxels to the total number of rays cast, the ratio of overlap $rat_{ovr}(C, C^*)$ between two circles C and C^* and a UGV local waypoint multiplier function $f_{ugv\_wp}(q)$, the information gain $Info(q)$ for a view configuration is recalculated as shown in Equation (C.1).

$$Info(q) = rat_{unmp} \cdot \left(1 - \min\left\{\sum_{C^* \in C_{other}} rat_{ovr}\left(C(q), C^*\right),\ 1\right\}\right) \cdot f_{ugv\_wp}(q). \tag{C.1}$$

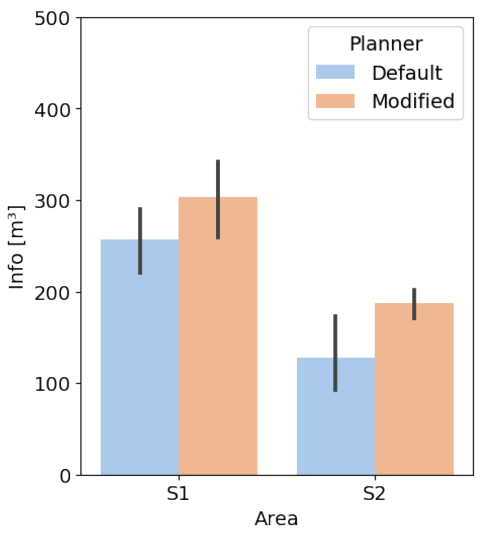
Here, $f_{ugv\_wp}(q)$ is the local navigation waypoints multiplier function. Given a view-configuration q, the distance $d_{cov\_wp}(q)$ between the corresponding coverage circle center and its closest local navigation waypoint is calculated. The function is presented in Equation (C.2), where the multiplier has a maximum value of c_{mult} and a minimum value of 1 dependant on the distance $d_{cov\_wp}(q)$ and the distance thresholds d_{mult_min} and d_{mult_max}. The function resembles a ramp that saturates at 1 for distances above d_{mult_max} and at c_{mult} for distances below d_{mult_min}.

$$f_{ugv\_wp}(q) = \begin{cases} 1, & \text{for } d_{cov\_wp}(q) \geq d_{mult\_max}, \\ c_{mult\_max}, & \text{for } d_{cov\_wp}(q) \leq d_{mult\_min}, \\ c_{mult\_max} - \frac{d_{cov\_wp}(q) - d_{mult\_min}}{d_{mult\_max} - d_{mult\_min}}, & \text{otherwise.} \end{cases} \tag{C.2}$$

A comparison of the Collaborative exploration planner with and without the modified utility function presented here was conducted in Section 5.4. Following this, results were gather for the Collaborative exploration planner using the modified utility function. Results of the Collaborative exploration planner using the default and modified utility functions on the tour scenario are presented here, without comments on the observed behaviour.

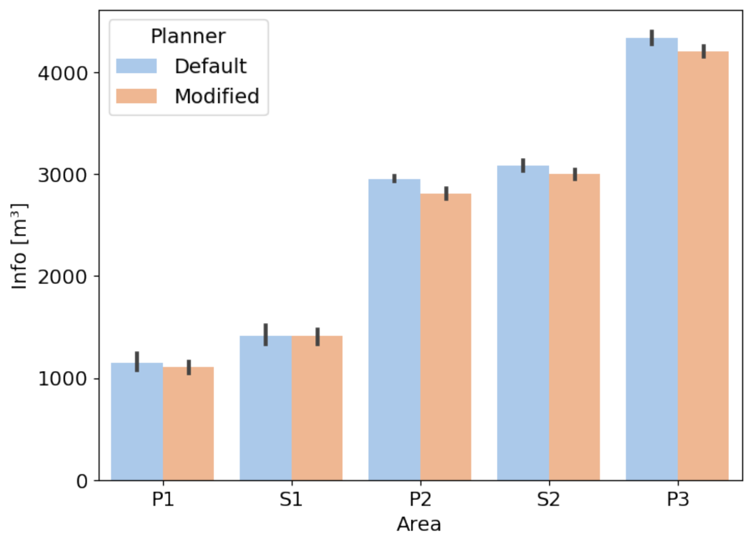


(a) Path areas.

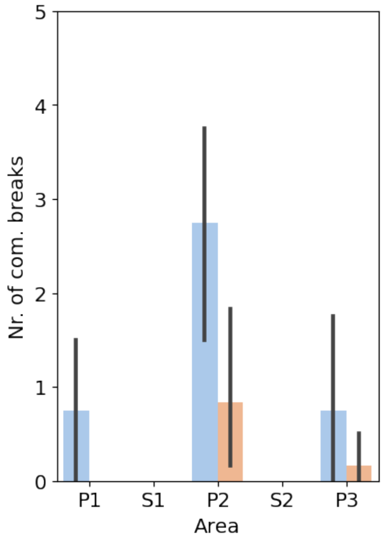


(b) Stationary areas.

Fig. C.1 Information gain in tour areas.



(a) Cumulative information gain.



(b) Com. breaks.

Fig. C.2 CUmulative Information gain and total Communication breakages pr. area.

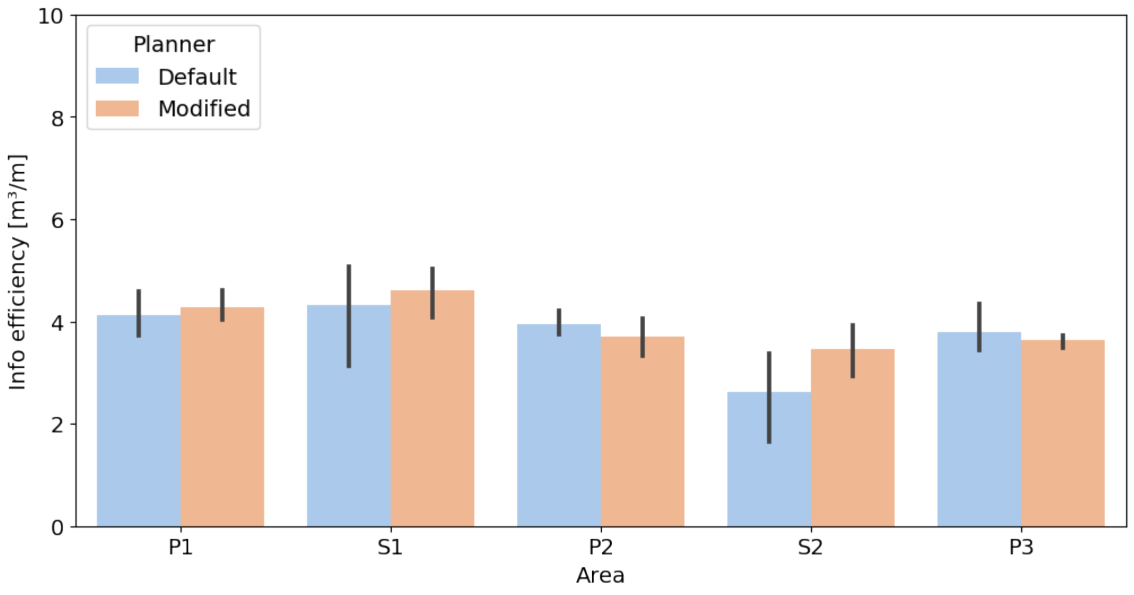


Fig. C.3 Information efficiency.

Appendix D

Formation bank

Number of UAVs = 1				
UAV id	Variable	x value [m]	y value [m]	yaw value [deg]
0	p_{close}	2.0	0.0	0.0
	p_{far}	2.0	0.0	0.0

Number of UAVs = 2				
UAV id	Variable	x value [m]	y value [m]	yaw value [deg]
0	p_{close}	1.0	0.0	-30.0
	p_{far}	0.0	-3.0	0.0
1	p_{close}	-1.0	0.0	30.0
	p_{far}	0.0	3.0	0.0

Number of UAVs = 3				
UAV id	Variable	x value [m]	y value [m]	yaw value [deg]
0	p_{close}	2.0	0.0	0.0
	p_{far}	2.0	0.0	0.0
1	p_{close}	0.0	0.0	-60.0
	p_{far}	0.0	-6.0	0.0
2	p_{close}	-2.0	0.0	60.0
	p_{far}	0.0	6.0	0.0