Fall 11-12-2017

Applying Heterogeneous Teams of Robotic Agents Using Hybrid Communications to Mapping and Education

Jonathan M. West

Jonathan M. West

Candidate

Electrical and Computer Engineering

Department

This dissertation is approved, and it is acceptable in quality and form for publication: *Approved by the Dissertation Committee:*

Prof. Rafael Fierro, Chair

Prof. Nader Vadiee

Prof. Jane Lehr

Dr. Asal Naseri

Applying Heterogeneous Teams of Robotic Agents Using Hybrid Communications to Mapping and Education

by

Jonathan M West

B.S., Electrical Engineering, University of New Mexico, 1995
M.S., Electrical Engineering, University of New Mexico, 2013

DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy
Engineering

The University of New Mexico

Albuquerque, New Mexico

December, 2017

Dedication

To my wife, Beryl and my sons Daniel and Joseph for all of their love and support

Acknowledgments

I would like to thank my advisor Prof. Rafael Fierro for the opportunity to work on the MAST project and for his leadership through my pursuit of this degree. Without your support and guidance, none of this would have been possible.

I am also grateful to Prof. Nader Vadiee and the faculty at the Southwestern Indian Polytechnic Institute for the opportunity to work on the Mars Yard program and teach courses over the last several years. The experience gained through working and teaching at SIPI has been wonderful and has introduced me to many great students and programs with which I hope to continue to work.

Also thank you to Prof. Jane Lehr for participating on my committee and allowing me to co-teach courses at UNM for the last few years. The teaching experience has has given me a great foundation on which to build my teaching career.

Also thanks to Dr. Asal Naseri for participating on my committee and for the helpful comments and feedback on my work.

And finally, thank you to the other members of the MARHES lab: Joseph Kloeppel, Shakeeb Ahmad, Christoph Hintz, Patricio Cruz, Rebecca Kreitinger and Carolina Gomez. Your help on the experimental testbed, demos and presentations was invaluable.

Applying Heterogeneous Teams of Robotic Agents Using Hybrid Communications to Mapping and Education

by

Jonathan M West

B.S., Electrical Engineering, University of New Mexico, 1995
M.S., Electrical Engineering, University of New Mexico, 2013

Ph.D, Engineering, University of New Mexico, 2017

Abstract

Robotic agents are being increasingly utilized to carry out tasks that are difficult or dangerous for humans. Many of these missions are best performed by heterogeneous teams of agents with various individual abilities. Communication among agents and with operators is a critical element in the performance and efficiency of these missions. Although radio frequency communications dominate the robotic networking field, they are limited in range and bandwidth due to spectrum congestion and subject to interference from noise or hostile jamming and can be intercepted. Optical communication has many advantages such as higher bandwidth and focused beam, however the line-of-sight requirement generally limits its range and application. Maintaining a continuously connected network between agents is also overly restrictive, dramatically limiting their freedom of motion and therefore efficiency.

In this work, we have developed a method to coordinate a heterogeneous team of agents using hybrid high frequency (HF), ultra high frequency (UHF) and optical wireless (OW) intermittent communications and cloud based computing resources to efficiently achieve a mission. This method is demonstrated by accomplishing an exploration and 3D mapping mission through a realistic simulation. The simulation includes accurate models of RF and optical noise and attenuations to reproduce real world scenarios. An experimental testbed was also developed to demonstrate the effectiveness of the system in real hardware.

Teams of robotic agents are also well suited to space exploration and the development of these agents and the algorithms to direct them are crucial elements of the education of engineering students. At the Southwestern Indian Polytechnic Institute (SIPI), we have developed a robotics-based educational program to teach engineering and programming through teleoperated robotic systems inspired by those used by NASA. The internet accessible Mars Yards provide a platform through which students in middle school, high school and college can learn programming, engineering, math and science. As part of the SIPI Mars Yard program we also developed an efficient visual localization system which is computationally light enough to operate on low power processors.

Contents

List of Figures

List of Figures

List of Tables

Glossary

HF	High Frequency radio (3-30MHz)
OW	Optical Wireless
ROI	Region of Interest
SIPI	Southwestern Indian Polytechnic Institute
SVD	Singular Value Decomposition
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
UHF	Ultra High Frequency radio (300MHz-3GHz)
VHF	Very High Frequency radio (30-300MHz)

Chapter 1

Introduction

As robotics continue to assume an increasing role in our world, many challenges must be overcome in order to continue to develop and apply them to various situations. In this paper we address several important areas in the robotics fields of communication and coordination, education and localization.

Communication and coordination among multiple robotic agents is also a critical component for any successful robotic team. In Chapter 2 we discuss an approach to coordinating a team of heterogeneous robotic agents and cloud computing resources to accomplish mapping of a complex, unknown environment. Our specific focus is on a situation where normal communication may be intermittent or impossible due to environmental characteristics or interferences. In these situations, robots with multiple channels of communication in both radio and optical media can accomplish tasks which robots with limited communications would be unable to complete.

In Chapter 3 we describe a real world testbed developed in thei Multi-Agent Robotics and Heterogeneous Systems Lab at the University of New Mexico. This tesbed provides an environment in which the concepts developed in Chapter 2 can be demonstrated and evaluated.

Computer programming skills are a critical necessity for today's students, but maintaining student interest in programming and engineering courses is challenging unless the theory is accompanied by engaging, hands-on applications. Many schools, especially those in underprivileged areas, lack the resources and personnel to develop or implement such applications. The Southwestern Indian Polytechnic Institute (SIPI), through the support of a NASA grant, has developed an integrated teaching program where students from middle school through college can learn programming and robotic design from the introductory level to advanced embedded computing, hardware and web page design. The centerpiece of the program is the indoor "Mars Yard" which is a SIPI facility that allows remote operation of robots in an indoor environment to simulate remote space missions. Beginning with simple Arduino-based robot kits, students are introduced to programming and robotics using an easy to follow curriculum. As they advance, students remotely access the Mars Yard and perform missions on real or simulated rovers. At the advanced level, the students proceed to design, build, program and test their own robots and sensors and develop custom missions.

The problem of localization is one aspect that is critical to almost all applications of robotics. Chapter 5 describes a platform independent, visual localization algorithm which allows a rover to find its location on a map based on recognition of visual landmarks. Although it is a platform independent software algorithm, this localization system is specifically designed to address the needs of the SIPI indoor Mars Facility and it therefore has very specific requirements and restrictions placed on it. Some of these restrictions are due to the effort to simulate the actual conditions on Mars. These include operation with significant communication latency, limited power resources, and lack of external positioning systems (such as GPS satellites). While these restrictions are required for the Martian simulation, they are also useful in terrestrial applications. Specifically, the lack of GPS is true for indoor applications and the power restrictions are important for battery powered systems.

Additionally, the desire to make the system applicable and extendable in educational applications demands that the system be as simple, small and inexpensive as possible so that students in schools with limited budgets can participate in the hardware and software development. The simple rover and lack of any sophisticated or expensive sensory equipment means that this rover functionality can reproduced on any platform at very little cost, thus opening the doors for robotics projects to schools which otherwise could not afford them.

1.1 Summary of Publications and Presentations

- Heterogeneous Systems
  - Conference paper published and presented at *2016 International Symposium on Distributed and Autonomous Robotic Systems*
  - Special Papers Session will be presented in the *2017 International Symposium on Multi-Robot and Multi-Agent Systems* in December 2017
  - Presented and published in the i*MAST Consortium Capstone Briefing* August 2017
  - Work up to the optical wireless communication system being prepared for submission to the *Journal of Intelligent & Robotic Systems*
  - Complete work being prepared for submission to the *International Journal of Robotics Research*
- SIPI I-C-MARS Program
  - Presented and published in the *2017 IEEE Integrated STEM Education Conference (ISEC)*
  - Presented at the *2016 Space Exploration Educators Conference (SEEC)* at the NASA Johnson Space Center, Houston, TX.

- Presented at the *2017 Soar To Greater Heights STEM Educators Conference* in Las Cruces, NM.
- Submitted for publication in the *Tribal College Journal of Native American Education* in October 2017 and is under review.
- Being Prepared for submission for publication in the *American Indian College Fund Tribal College & University Research Journal (TCURJ) Volume III*

Chapter 2

Heterogeneous Robotic Teams Using Intermittent Hybrid Optical and Radio Communications

2.1 Introduction

In this paper we discuss an approach to coordinating a team of heterogeneous robotic agents and cloud computing resources to accomplish mapping of a complex, unknown environment. Coordinating the actions of a heterogeneous team of robotic agents is a difficult problem especially since the agents may all possess different abilities and limitations. The diversity of capabilities of heterogeneous agents greatly increases their flexibility and ability to accomplish complex tasks, however it also makes coordinating their efforts challenging. In [1] the authors describe methods of distributing tasks among heterogeneous agents based on matching each agent's capabilities to specific parts of the task.

In the event of an emergency, whether due to hostilities, natural disaster or other-

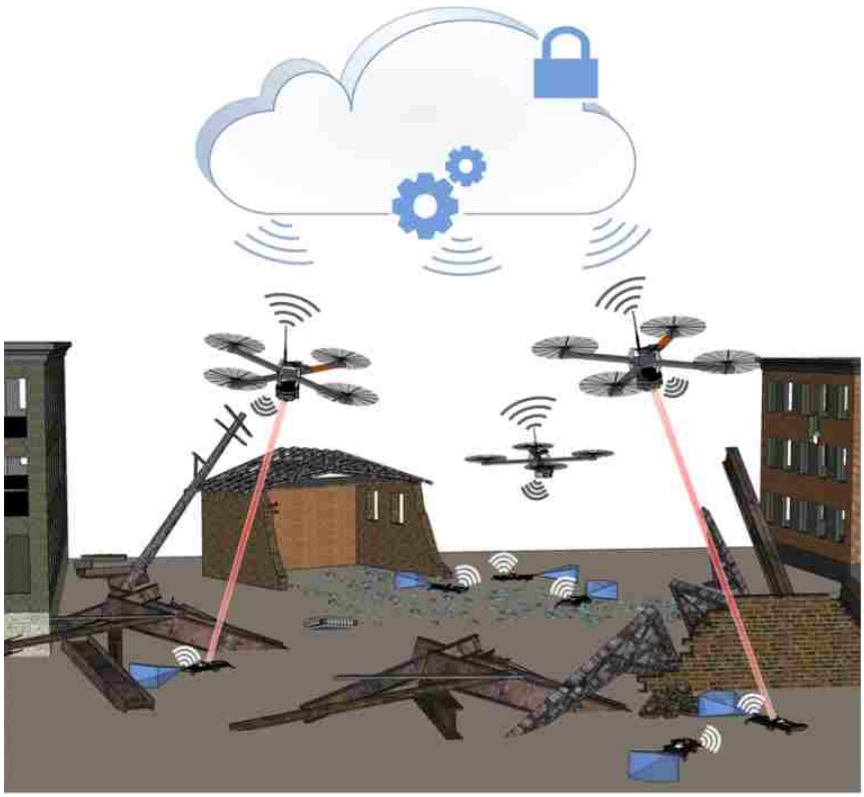


Figure 2.1: Scenario of mapping a collapsed building using a heterogeneous team of robotic agents.

wise, it is often necessary to enter and map an unknown environment where human lives could be endangered due to structural instabilities, dangerous environmental elements or hostile actors. In these cases, it would be advantageous to deploy a system of robotic agents to map the environment and indicate the locations of people in distress or dangerous elements.

For the purposes of this discussion we consider the environment of a building that has been partially collapsed as shown in Figure 2.1 due to an event such as an earthquake or hostile action. A team is deployed to the area, but cannot enter the building. They deploy a heterogeneous team of robotic agents to inspect the building and map its interior. The team consists of UAVs which can enter the building quickly,

and record aerial images of the interior. However the range of the UAVs is limited by their battery capacity and the possibility that there are features such as narrow passages or blocked hallways which prevent the UAV from flying into certain areas. The areas which are inaccessible to the UAVs may be accessed by small, agile UGV agents which can go under or around the obstacles and explore the areas beyond. Both the UAV and UGV are capable of quick movement and data collection, but lack the storage and processing power to combine the images and other data into a useable map. They also require coordination in order to adequately cover the region of interest. This requires a more powerful and therefore less mobile server to be available. The server can combine the data from all sources, plan the deployment and movements of the agents and communicate the results to operators.

However, as in most multi-agent situations, communication becomes the limiting factor. If the agents must maintain constant contact with the server or each other, then they are severely limited in their flexibility. The communication will be degraded by the distance between agents, the communication medium separating them and possibly by noise or hostile jamming.

We therefore are working to develop a deployment and communication scheme which can maximize the effectiveness of each agent and combine the results in the most efficient manner possible. The novel aspects of this work include the use and coordination of heterogeneous agents and the use of three different communication channels to effectively execute a mission.

This work was supported in part by the Army Research Lab MicroAutonomous Systems and Technology Collaborative Alliance ARLMAST-CTA #W911NF-08-2-0004.

2.2 Related Work

2.2.1 Heterogeneous Agents

Many mapping or searching operations are too complex to be carried out by a single type of robotic agent. Many studies including [2] have been done to map agent capabilities to mission requirements and to find the minimal set of agents required for a task. In [3] and [4] the authors discuss the effectiveness of heterogeneous teams of ground agents in accomplishing a mapping operation, and in [1] the effects of diversity on the completion of tasks is explored. These and many similar projects have demonstrated the necessity and effectiveness of using many different agents with various capabilities.

2.2.2 Communications

The communication among the agents and between the agents and the base or cloud is a critical element of the operation. An important decision regarding the operation of the agents is whether or not the communication network must be continuously connected. Most planning algorithms such as [5] expend great effort to ensure that no agent moves out of communication range of the others. While this is certainly the safest and simplest approach to avoid agents becoming disconnected or lost, it is highly restrictive and may even render the mission impossible since it restricts the separation of each rover and leads to serious bandwidth congestion, especially when using RF signals. Approaches such as described in [6], [7] and [8] allow for periodic connectivity where agents must check in with a base or one another at regular intervals. This type of approach is also necessary if there are regions of the environment which can only be reached by breaking communication links. In [9] the authors describe use of a UAV to carry data between unconnected agents acting as

a data mule and in [10] the authors address communication issues in multi-robot systems.

In this paper, we consider three types of communication channels as shown in Table 2.1. Each mode represents a trade-off between range and bandwidth.

Table 2.1: Communication Channels

Type	Range	Bandwidth	Use
HF RF	Long	Low	Commands and Status
UHF/VHF RF	Medium	Medium	Map Sharing
Optical	Short	High	Video / Sensor Data

Radio Frequency (RF) Most robotic agents communicate via radio frequency channels such as WiFi, bluetooth, Zigbee and others. These RF communications are highly effective, simple, cheap and consume reasonable amounts of power. They are also thoroughly developed and tested and have extensive hardware support. Much research has been done regarding establishing and maintaining RF communication networks among robotic agents [6]. However RF communication is limited by several factors. UHF communication range is limited due to its poor penetration of structural elements. HF communications can have very long range due to minimal attenuation, however it requires large antennas and has low bandwidth. Both forms of HF suffer from congestion and interference. For many applications, RF interference may not be a dominant concern since most environments present few sources of such interference. However for military or police actions, the possibility of a hostile jamming agent can potentially render RF communications useless which requires an alternative communication channel to guarantee mission success. Another limitation of RF for large teams of agents is the shared channel bandwidth where all agents and any other RF systems in range must share the frequency spectrum and therefore may degrade the available bandwidth of the communications. An additional concern

with RF communication may also be that of security. Since RF is typically broadcast omnidirectionally, it may be readily intercepted by hostile agents. Although encryption can be employed to ensure security, it comes at a high computational and bandwidth cost. The broadcast of RF energy also may serve as a beacon for hostile parties seeking to locate the agent.

Optical Wireless (OW) An optical wireless communication system has been proposed in [11] which allows high bandwidth communication over distances that are reasonable for indoor environments. That paper describes an OW system which allows a unmanned ground vehicle (UGV) to communicate with an unmanned aerial vehicle (UAV) and presents a control algorithm to maintain that communication channel over a reasonable period of time.

The primary limitation of OW is the line-of-sight requirement. If high bandwidth and long range communication is required, then a narrow beam laser is the best communication channel. However, a laser requires precise pointing and tracking hardware and software and may not be practical in dynamic environments. Shorter range, lower bandwidth communication is possible with spread laser or LED beams which greatly relax the requirements of the pointing system. All OW systems are limited by the quality of the air between the agents with smoke, fog or dust effectively jamming the signals and dramatically limiting the useable bandwidth. However, the directional nature of the OW beam means that the communication can only be detected and intercepted in a small area which enhances the security of the system. It also allows many agents to communicate simultaneously without interference as in the RF case.

2.2.3 Mission Planning

In [12] Wettergren and Bays describe a solution for planning coordinated deployments of agents. This solution model is used as a starting point for our model since it accounts for similar mobility and fuel constraints.

Methods of agent deployment and path planning for environment exploration are presented in [13], [14], [15] and [16]. Approaches to implement the control and coordination of teams of agents have been developed in [17], [18] and [19]. In [20] and [21] different approaches to allocating tasks among heterogeneous agents are presented.

2.2.4 Coordinated Localization and Mapping

In [22] an approach to coordinate localization between UAVs and UGVs. In [23] an algorithm is presented to map an unknown environment using a single robot. This algorithm is used as the basis for the individual agents map building operations. In [24], [25] and [4] the authors describe approaches to combining maps collected by individual agents into a single global map. In [19] the authors present a cooperative mapping algorithm for distributed and possibly disconnected agents. It includes independent frontier exploration and map merging.

2.2.5 Cloud Computing

Our approach requires the use of cloud computing resources for the computational and storage capacity require for the image processing and mapping. In [26] the authors present an approach for coordinating data collection to cloud storage and processing resources. In [27], [28], [29] and [30] cloud based robot software architectures are developed. And in [31] a system is presented which augments the capabilities of

simple robotic agents with cloud computing resources and in [32] a cloud engine is presented to provide general computing services to robotic agents.

Although each of these works addresses a particular aspect of agent performance, none address the complete solution of heterogeneous agents with various communication methods. They also typically focus on relatively simple mapping resulting in occupancy grid type maps, but typically do not consider the fact that the agents may also need to collect and deliver large volumes of critical data such as images or senor readings. Our research aims to utilize all of the best aspects of these and other approaches in order to provide an improved approach to coordinating the various elements into a cohesive operation to achieve the mission of mapping and exploring an unknown environment.

2.3 Model Formulation

2.3.1 Mission Environment Definition

Let there exist an environment which is to be searched and mapped. This environment is described by a set of maps indicating various features. For purposes of this discussion, we consider only a 2 dimensional planar environment such as a single floor of a building, but the concepts are extensible to multiple floors. This building environment shown in Figure 2.2 is modelled in the ROS Gazebo simulator which provides a realistic physics model for the agents and the building elements such as floors and walls. This allows us to use standard ROS sensors such as cameras and laser range finders to simulate the rover data collection.

The communication characteristics of the environment such as RF and OW attenuation and noise cannot be modelled directly in Gazebo. These characteristics of the environment are represented by digital maps, the cell values of which represent

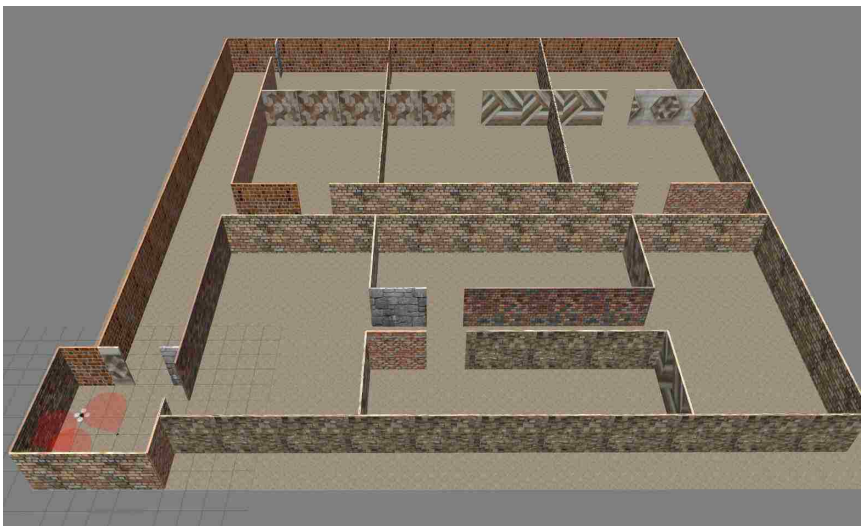
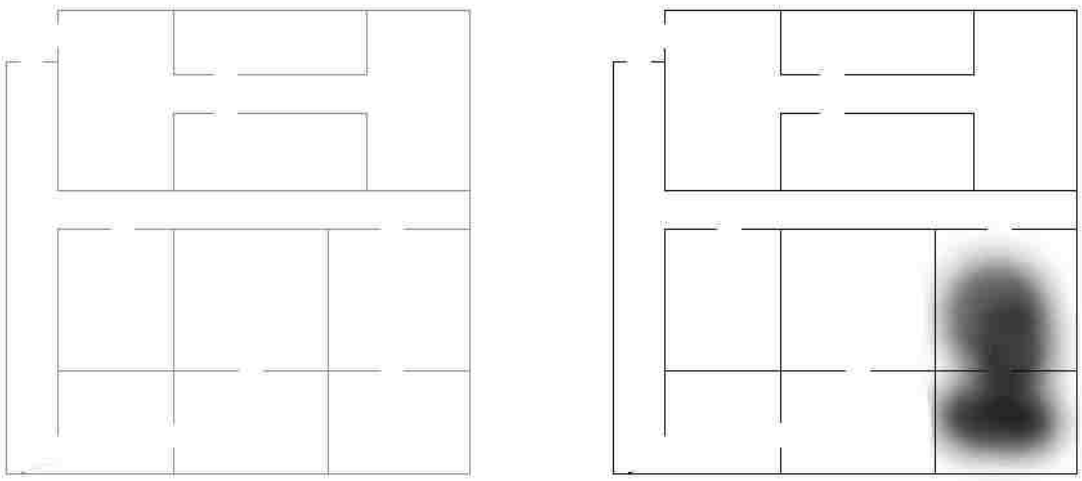


Figure 2.2: Gazebo World for exploration and 3D mapping mission.

obstacles or other environmental characteristics.

Transmissivity Maps

The ability of a particular communication channel to travel through the environment is modelled using transmissivity maps. The cell value of this map indicates the transmissivity of the medium for communication. A value of zero indicates complete attenuation while one indicates no impedance to transmission. Obstacles such as walls will have values of zero for optical communication and a value less than one for RF signals. For optical signals, areas in which smoke or dust are present will have values between zero and one to indicate the density of the obstruction. Figure 2.3(a) shows the RF transmissivity map used in this simulation. Note that in this case, the walls are not fully black since they attenuate but do not block the signal. Figure 2.3(b) shows the OW map used in this simulation which includes an area which is partially opaque to indicate an area in which smoke or dust are present.



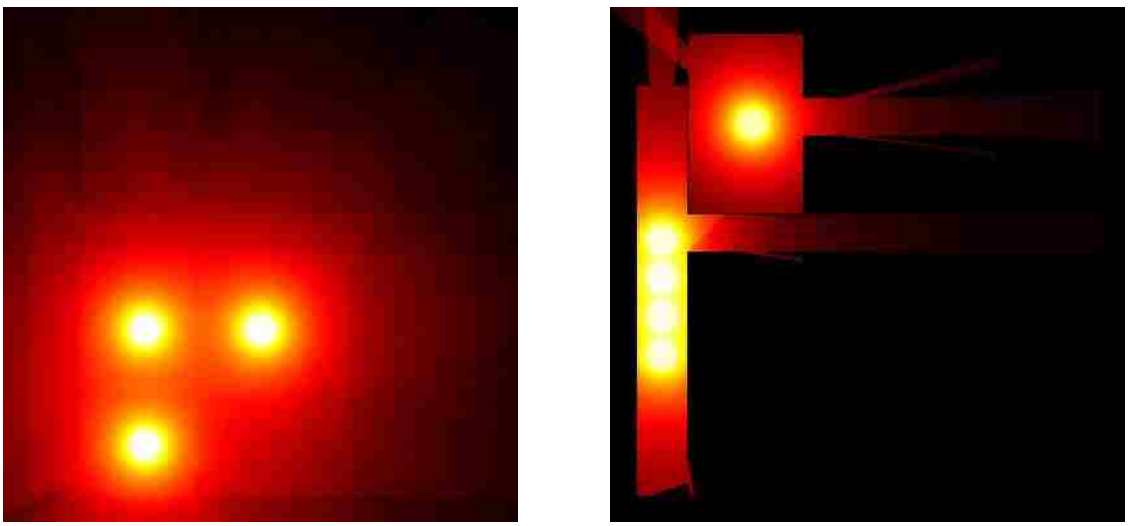
(a) RF Transmissivity (b) OW Transmissivity

Figure 2.3: Transmissivity maps indicating attenuations to transmissions. Note that the RF walls are grey, since they allow some transmission, but the OW walls are black.

Noise Sources

Another obstacle to communication is the presence of noise or jamming signals. The noise signals are indicated as the locations of sources of radio or optical energy and their strength. For this simulation it is assumed that the noise sources are of the same spectrum as the communication signals. To calculate the noise levels at each point in the environment, the noise from the sources is passed through the Transmissivity Map for that channel and then added to the noise from all other sources. The resulting noise value is stored in a map, the pixels of which are the received noise at each location. This received noise is used in the signal to noise ratio calculation at the receiver. The noise is simulated for RF and optical signals as shown in Figure 2.4.

It must be noted that these maps define the environment, but are unknown to



(a) RF Noise (b) OW Noise

Figure 2.4: Noise maps showing the received noise at each location

the agents until they are discovered and mapped. These serve as inputs to the simulations.

2.3.2 Graph of Agents and Connections

The system of agents is represented by a connected graph $G = (A, C)$ with $C \subseteq [A]^2$ where $A = \{1, ..., A\}$ is the set of agents and $C = \{1, ..., C\}$ is the set of communication paths connecting the agents. Each agent in the system is described by a class which contains all of the information about the agents capabilities and limitations. The edges C of the graph represent the communication channels between agents and are influenced by many different factors. Each edge can, at various times represent an RF or OW link and each has a weighting which is the available bitrate capacity of the channel. The bitrate of each edge will vary as the relative physical locations and environment between the agents changes. If the bitrate goes below the minimum for that channel, the edge will be removed from the graph. It is likely that

the graph will not always be bidirectional since the receiver noise may be different for the connected agents. In this case, one agent will hear the other but not be able to reply.

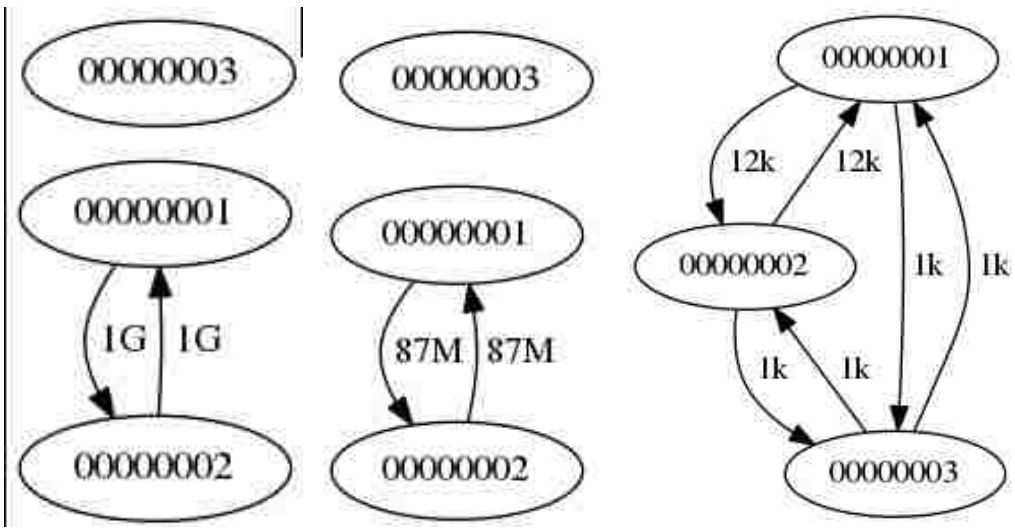
Each communication channel has its own graph which is dynamically updated as the mission progresses. The edges will appear or be removed as communication on that channel is possible.

2.3.3 Agents

Each agent is modelled as a vertex on the graph and is represented in simulation as a class. The *Agent* class contains lists of sub classes which an agent can contain including communication channels, sensors, data storage, locomotion and batteries. The details of each of these classes include all of the necessary parameters to correctly represent their behaviour or limitations. For example, the *Locomotion* class includes the energy required to hover and to move, and the *Battery* class contains the energy available. As the simulation steps through time, each of the classes calculates how much of each resource it consumes (i.e. energy or data storage) and how much it contributes to the mission (i.e. area mapped or sensor data collected.) The status of each class is reported so that the agent and the base can determine the appropriate actions.

2.3.4 Communication Channels

Each communication channel between nodes is modelled as an edge on the graph. The characteristics of the communication channel is modelled as a class including the bandwidth, transmission power and minimum bitrate required to maintain a connection. The available bitrate is calculated based on the distance between the



(a) OW Com (b) UHF Com (c) HF Com

Figure 2.5: Communication Graphs showing connections and bitrates

agents, the transmission characteristics of the medium, and the in-band noise at the receiver. For example, if the air is smoky or cloudy, or if there is an opaque obstacle between the agents, then the OW bitrate will go to zero. The RF bandwidth will similarly be effected by distance and obstacles or interfering signals. Example graphs for HF, VHF and OW connections are shown in Figure 2.5. In this case, Agent 3 has moved out of range or UHF and OW, but still maintains HF communications at a low bitrate.

Power vs. Distance and Attenuation

In the absence of obstructions, the power level will vary inversely with the distance squared. The attenuation of the signal due to the environment is calculated as the integral of the attenuation along the path between the agents. In the simulation this integral is calculated as the product of values in the cells of the transmissivity map M through which the signal passes. The set of cells C through which the signal

passes are found using the *Supercover Line* algorithm described in [33]. Combining these factors gives us the power at the receiver according to (2.1), where P_{tx} and P_{rx} are the signal powers at the transmitter and receiver, d is the distance between the points, C is the set of map cells between the points and M_{C_i} is the attenuation of a particular cell.

$$P_{rx} = P_{tx} \frac{1}{d^2} \prod_{i=1}^{C} M_{C_i} \tag{2.1}$$

Noise

The noise maps provide the amount of noise that is present in each cell of the grid from all noise sources present in the simulation. This noise at each map cell is calculated by using (2.1) to propagate the noise from the source to that cell. The input to the noise map creation is a list of locations and transmission power of the noise sources. The sources are propagated to every other point on the map and the sum of all sources at each point is recorded. The interference from simultaneous transmissions from multiple agents is handled in the simulation by coordinating the transmissions so that only one agent on each connected graph can transmit at a time. Disconnected graphs are assumed to not interfere with each other as in the case where many OW connections can be made in different locations without interference.

Channel Capacity

The bitrate of the channel is calculated from the bandwidth and signal to noise ratio according to the Shannon-Hartley theorem shown in (2.2), where BR is the channel's bitrate capacity, B is the bandwidth, P_{rx} and P_N are the signal power and noise power at the receiver.

$$BR = B * log_2(1 + \frac{P_{rx}}{P_N}). \tag{2.2}$$

This value is applied to the graph edge C corresponding to the connection between the agents. This value is the maximum channel capacity but does not imply that the agents can communicate at that rate. The Agents will communicate at the channel's bitrate which is determined by the specific protocol and must be less than or equal to the channel capacity. Since agents are capable of communicating using multiple channels, the capacity of each channel is calculated separately using the appropriate maps.

2.4 Methodology

2.4.1 Agent Subclasses

The purpose of the agents is to collect image and sensor data and deliver it to the Base Agent and then to the operators and possibly to cloud computing resources for analysis and mapping. As previously described in Section 2.3.2, the agents of our system are defined by the set A.

We now define n exclusive subsets of A as different types of agents within the system where $A = \bigcup_{i=1}^{n} a_n$ and $\forall A_n \subset A$.

Although, in general, n can be large, for convenience in our discussion we will define 3 types of agents A_A = UAVs, A_G = UGVs and A_B = Base stations each of which include one or more agents with a common class definition.

Once the set of agents is defined for a particular scenario, the next step is to determine how they will communicate and explore the environment.

2.4.2 Communication Structures

The communication between agents consists of the following datasets:

Status

The lowest bandwidth signals are the periodic status messages. These messages are kept to the absolute minimum size and frequency to permit sharing of the limited communication available on the HF channel. These messages include the pose of the agent and the revision numbers of the agent's Operation, Occupancy Grid and Image Progress Grids.

Command Pool

The Command Pool is the set of all of the commands that are active in the system. Typical commands would be to explore a frontier, go to a particular location to record sensor data, go to another agent to act as a data mule or return to base. The current command set is shown in Table 2.2.

When an agent needs to issue a command to another, it simply adds that command to its current revision of its Command Pool. When another agent is in range, that revision will be merged and therefore the command will propagate through the network and eventually reach the targeted agent. The status of each command is also propagated through the network in the same way so that the issuing agent will know if the command is being acted upon or is completed. A history of all commands is maintained in the Command Pool structure, with each command associated with a priority and a status. The available status states are shown in Table 2.3.

Each command is assigned a priority value from 0 to 100 and is acted upon in a preemptive manner. When an agent is actively executing a command, that command

Table 2.2: Agent Commands

Command	Description
Transmit	Transmit data to a mule
Receive	Receive data from a mule
Transfer	Instruct an agent to begin a data transfer
Goto	Go to to a given pose
Explore	Search frontiers for occupancy grid
Mule	Perform data Mule function for an agent
Image	Record images on nearest frontier

status will be *Active*. If another command is received with a higher priority, then the current command's status sill be changed to *Preempted*, and the higher priority task will become *Active*. When the current task is finished it will be marked as *Complete* and the next highest priority task that is not marked as *Complete* will become *Active*.

The priority is a relative indication of how important a task is. If an agent discovers extremely urgent information such as an image of an explosive device or person in need of rescue, this is assigned an priority of 100, meaning it must be completed immediately at all costs. In this case, the agent will return to base as quickly as possible while simultaneously requesting a data mule to relay the sensor information as quickly as possible. The data mule will check the priority of this mule request against its current command and if the priority is higher, it will preempt its current command and execute the data mule command. Even if the UGV passes the data to a mule, it will still proceed to the base to deliver the data until it receives status messages indicating the task has been completed. Although it is not efficient for the agent to do so if the data mule accomplishes its task, it is still necessary in case the data mule is unable to complete the mission due to a crash or failure. Once the urgent task is completed, the agents either go back to their original commands or to new ones which may preempt the original commands.

Table 2.3: Command Status

Command	Description
Pending	Command has been issued but no action taken
Active	An agent is currently executing the command
Preempted	Command was active but has been preempted by another
Complete	Command successfully executed
Cancelled	Command cancelled by issuer or acting agent

Occupancy Grid

The first priority of exploration is to develop the occupancy grid so that the base and operators will know the layout of the environment and where best to deploy agents. The fastest agents are initially dedicated to this task. Once the occupancy grid is known, the fast agents become data mules or collect images themselves (if possible) as needed.

The Occupancy Grid is stored as a standard ROS Occupancy Grid message. Each agent stores its own grid but also maintains a grid formed by the synchronization of the grids from other agents as they are available.

The occupancy grid is used for the agent navigation and path planning.

Imaging Progress Grid

Similar to the Occupancy Grid, the Imaging Progress Grid is a map of the area in which the pixels indicate if an area has been imaged. It is compared against the Occupancy Grid to determine if all of the area of interest have been imaged. The Imaging Progress Grid is stored and shared in the same manner as the Occupancy Grid.

Images

The ultimate objective is to collect images of the entire area and deliver them to the BASE agent and then to the cloud resources for processing. The slower UGV agents are dedicated to imaging operations, although they also contribute to the occupancy grid as they go.

The image data is too large to transmit via slow HF and UHF channels and therefore must rely on OW communication.

2.4.3 Intermittent Communication

Synchronization

The communication and synchronization of the commands and data through the intermittent channels is accomplished through a distributed version control system modelled after the popular *git* program [34]. Each agent maintains a snapshot, or revision, of the Status, Operations, Occupancy Grid and Imaging Progress Grids and for all agents. Each of those snapshots is stamped with a randomly generated, globally unique revision number. When an agent makes a new revision of any of these data sets, the revision number is changed to reflect this change. Each agent broadcasts its status into the communication channel periodically for all other agents in range to receive. The status message includes the agent's pose and the revision numbers of its commands, status, images and maps. When two agents are in communication range of each other, one will receive the status message of the other and then merge the two revisions into a new one.

In this way, each agent will always have the best possible copy of the status and map structures. Of course when the communication graph becomes disconnected, all agents will have different data revisions, but as they continue to connect to each

other over time, the versions will eventually converge and become consistent.

Merging of the commands and status structures can be done over the HF communication link which allows most agents to receive commands and provide pose updates either directly or through intermediately connected agents. Merging of the maps must be done through the higher bandwidth UHF channel. Although each agent will likely not have a direct path to the base over UHF, the map coordination will take place through relays between various connected agents or by data mule operation as described in [9].

High bandwidth sensor data such as video must be transmitted via OW connections. This is typically only done between an agent and the base or an agent and a data mule.

2.4.4 Deployment and Exploration

It is assumed that the base agent is responsible for the overall coordination of the mission and is the only agent capable of communication with the operators or the cloud.

When first deployed, the agents are all located within communication range of the base and are waiting for deployment commands. We will assume there are the communication channels mentioned in Table 2.1 available. If an agent is out of range of one or more of these channels, then it will be required to buffer its data internally until it is able to communicate again.

The Base agent scans the immediate area and identifies the known frontiers. It then looks at the set of agents available (known through the reception of Status messages on the network) and dispatches each agent to a frontier with a *Goto* command. The UAVs (if any) are given *Explore* commands and the UGVs are given *Image*

commands so that when they reach the frontiers, they will begin either exploring or imaging.

Once the UGVs have finished exploring (no more frontiers exist on the Occupancy Grid) the Base will issue *Mule* commands for each of the UGVs. The UAVs will then fly to each UGV, collect its data, return to the Base and repeat until all of the UGV data has been collected and returned.

The UGVs collect their Image data until there are no more frontiers in the Image Progress Map or their storage becomes full. When finished (or when recalled by the Base if it determines the imaging is complete) the UGVs return to base and transfer any remaining data to the Base.

As soon as image data is transferred to the Base agent (either by data mule or directly) it will be being the upload to the cloud and the Cloud will begin processing. The cloud processing works on subsets of the image data and reassembles the resulting 3D maps based on the agent's poses. The reassembled maps are then transferred back to the Base or to the operators for analysis.

For purposes of the discussion and to match our testbed and simulation environments we will make the following assumptions:

- UGVs have sensors to measure both the Occupancy Grid and Images.
- UAVs have sensors only to measure the Occupancy Grid.
- Only the Base agent can communicate with the Cloud.
- All agents can communicate with each other via HF, UHF and OW channels when the environment permits.

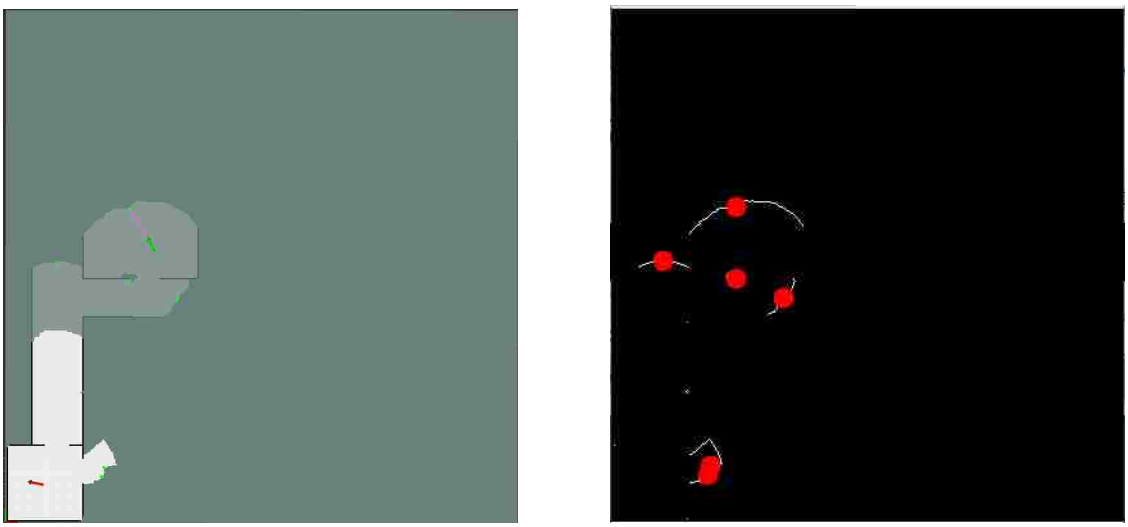
The mission consists of the following operations:

- Collection of Occupancy Grid data for all reachable areas
- Transfer of Occupancy Grid to base agent
- Transfer of Occupancy Grid from base to cloud
- Collection of Images of all reachable areas
- Transfer of image data to the base station
- Transfer of image data from base to cloud
- Processing of image data in cloud

Some of these are sequential and some can be performed in parallel. For example, the Occupancy Grid and imaging can be done in parallel, but the transfer of data from the base to the cloud and the processing in the cloud cannot begin until the data is delivered to the base agent. The critical parameters are how quickly the Occupancy Grid and images can be transferred to the cloud since that is the point at which the information becomes actionable.

2.4.5 Frontier Exploration

In order to develop the Occupancy Grid and Imaging Progress Grid, a robust and efficient frontier exploration algorithm is needed. A frontier goal is a point to which the agent should move in order to best explore the frontier. The frontiers are identified as areas in the known map where open space is adjacent to unknown space. This is calculated using OpenCV and results in a binary image in which the frontier pixels are 1 and all others are 0. This binary image is then used to calculate contours of connected pixels. Each contour is a vector of all of the pixels which have a value of 1 and are adjacent to each other. An example of an Imaging Progress Grid and the resulting frontier identifications are shown in Figure 2.6.



(a) Imaging Progress Grid (b) Frontier Identification

Figure 2.6: Imaging Progress Grid map and corresponding frontier identification

It is often the case, for example when an agent enters a room, that a contour will be very large and the nearest point will not be the best exploration point. In this case, if a contour is longer than a threshold, it will be segmented into several contours each of which will have a length less than the maximum. The list of frontier goals are then selected as the point along each contour that is nearest the agent's pose.

The list of frontier goals is then analyzed to choose the best one for this agent to pursue. The selection of the best goal is performed using a cost function. The cost of a frontier goal is calculated using (2.3), where d is the distance required to travel to that frontier, l is the length of the frontier and θ is the required change in heading to move toward the goal.

$$C = W_d d + W_l l + W_\theta \theta \tag{2.3}$$

The distance d is calculated using the ROS navFN algorithm computed against the currently known Occupancy Grid. Since the Occupancy Grid may be incomplete,

Table 2.4: Frontier Exploration Weights

Weight	Value	Description
W_d	1	Cost per meter to goal
W_l	0	Cost per meter of frontier length
W_θ	100	Cost per radian heading change

d and θ may not be accurate, but they will always represent the best estimate based on the current knowledge.

Each of those values are weighted by an appropriate cost factor W_d, W_l and W_θ. The most important factor is typically d since it is logical to pursue closest frontiers first. The second most significant factor is θ since it is most efficient for the agent to proceed mostly forward and avoid oscillating back and forth between goals. The l factor is given the least weight since in a building, small doors often lead to large frontiers. This weighting favors the agent quickly going through rooms to the farthest extents before carefully exploring each room. Different weightings will result in different behaviours. The weights used in our simulation are shown in Table 2.4.

It is also possible that a frontier will be too close to the agent for the agent to actually record data. For example if a point to be imaged is too close to be visible in the camera or the area is too close for the laser scanner to resolve it. In this case, the agent will move away from the frontier and then re-approach it.

The coordination of multiple agent exploration is difficult since it is decentralized and the agents are possibly disconnected and most likely do not have the same Occupancy Grid or Imaging Progress Grids. Since an agent cannot know the Grids of agents with which it is not connected, it will use the last known position and Grid from each agent to attempt to avoid duplication. Consider two agents (A and B) which are both exploring the Imaging Progress Grid. Agent A identifies all of the frontiers on its map and then calculates the cost of exploring each of them. Using

the same frontier list, agent A then calculates the cost of each frontier goal for agent B, using agent B's last known pose instead of its own. Agent A then compares its own cost for each goal with B's cost and if B's cost is lower, then A ignores that frontier, assuming that B will pursue it instead. Agent A will then do the same comparison with all other agents and eliminate the frontiers that they should be exploring. If after all of the eliminations, A has no more frontiers (as may be the case if both agents are travelling down the same hallway), then it pursues the one with the lowest cost and assumes that eventually the goals will diverge again.

2.4.6 Multi-Agent Performance Improvements

Estimation of multi-agent performance is difficult to generalize since it depends completely on the structure of the environment. Large open spaces will lend vastly different results from office building with many hallways and small rooms. We will discuss some general estimations and then apply them to our specific test case.

Frontier Exploration

The maximum rate at which an agent can explore a map is given by 2.4 where v is the velocity at which the agent can move while collecting data r is the radius of the range of the sensor and θ is the angle over which the sensor can collect data. These parameters will be different for the UAV and UGV agents as shown in 2.5

Using this formula, the time it will take to explore the area is given by (2.5), where A is the area to be searched.

$$R = v * r * \frac{\theta}{2\pi}[\mathrm{m}^2/\mathrm{sec}] \tag{2.4}$$

$$t_{explore} = \frac{A[\mathrm{m}^2]}{R[\mathrm{m}^2/\mathrm{sec}]} \tag{2.5}$$

Table 2.5: Frontier Exploration Parameters

Parameter	UGV	UAV	Use
v_{images}	0.3		Velocity [m/s]
r_{images}	1		Sensor Range [m]
θ_{images}	$\pi/4$		Sensor Angle [rad]
v_{range}	0.3	1	Velocity [m/s]
r_{range}	1	5	Sensor Range [m]
θ_{images}	$\pi/2$	$\pi/2$	Sensor Angle [rad]

This is the theoretical best case, however, this assumes the agent travels in straight line, never retraces the same area and the area is unobstructed. Characterizing the amount of time lost for retracing and avoiding obstacles is difficult since it depends directly on the environment. Retracing is a factor of the layout of rooms and hallways such that the agent must travel back through a previously explored area to reach an new frontier. Even in a completely open area, retracing will be required due to the need to return to base. We introduce a weighting parameter $d_{retrace}$[m] which is the distance the agent must travel over already explored areas due to the geometry of the environment. Distance driven to avoid obstacles and walls is also a large factor resulting in lost time. We introduce a factor $d_{obstacle}$[m] to capture the additional distance needed to travel to go around walls and obstacles. The $d_{retrace}$ and $d_{obstacle}$ parameters are not independent and will usually overlap. These parameters are impossible to predict with any certainty in an unknown environment, but they can be statistically determined for different types of known environments and can therefore be estimated from simulations in the hope that they will provide useful estimates for real environments that have characteristics similar to the simulated ones. The equation for the time to explore the environment with these additional factors is shown in (2.6).

$$t_{explore} = A[\mathrm{m}^2] * R[\mathrm{m}^2/\mathrm{sec}] + (d_{retrace} + d_{obstacle}) * v \tag{2.6}$$

Effect of Multiple UGV Agents

For a team of n similar agents, the area is divided among them and they can explore in parallel. Ideally, with similar agents, this will give the exploration time according to (2.7).

$$t_{explore} = \frac{A}{n}[\mathrm{m}^2] * R[\mathrm{m}^2/\mathrm{sec}] + (d_{retrace} + d_{obstacle}) * v \tag{2.7}$$

However the division of the area is not done with perfect efficiency because the agents may have to travel the same areas to get to the frontiers. For example, several agents may have to travel the same hallway to get to unexplored rooms, resulting in increasing $d_{retrace}$. Duplicate exploration also results from lack of coordination when one agent wastes time exploring an area that has already been explored by another agent, but lack of communication prevents the agent from being aware of it. Adding in the duplicate area A_d gives the final equation (2.8).

$$t_{explore} = (\frac{A}{n} + A_d)[\mathrm{m}^2] * R[\mathrm{m}^2/\mathrm{sec}] + (d_{retrace} + d_{obstacle}) * v \tag{2.8}$$

Effect of Adding UAV Agents

Adding a UAV agent decreases the mission time in several ways. First, the UAV has a higher exploration rate v and can therefore explore the map much faster than the UGVs. Second, the UAV can act as a data mule for the UGVs to bring the data back to the base quickly. Third, the UAV can carry updated maps and information among the agents to keep them better coordinated and therefore increase their efficiency.

2.4.7 Effect of Adding Hybrid Communications

The hybrid communication system provides the best of all communication possibilities. In order to understand the benefits of the hybrid communication system, we describe the benefits of each channel and the drawbacks if each channel is unavailable.

HF: Status and Commands

In order to relay commands to agents and to know their locations to send data mules, the agent's status (and therefore pose) must be known to the base and other agents. The HF channel is therefore the most important for efficiently achieving the mission.

Lack of the HF communication would require that the agents periodically check in with the base and the other agents over UHF or OW in order to update the Command Pool and know the poses of the other agents. This effectively decreases the W_r parameter since the agents will retrace the path to and from the base many times. It also requires the agents to operate autonomously between check-ins which will increase the likelihood of them duplicating each others efforts, decreasing W_d and it all but eliminates the use of the UAV as a data mule since it will not know the UGV's locations.

UHF: Map Sharing

The use of UHF to share the mapping progress allows agents to avoid overlapping each other's progress. Since UHF has a reasonable range and can penetrate walls, the agents can share these maps opportunistically without the need to stop and establish an OW channel.

Lack of UHF connection will result in the need to use OW to share maps. This will increase the time for map synchronization since the agents must coordinate to

be in OW range and can no longer share data opportunistically at a distance.

OW: Image Transfer

The OW is the most effective way to transfer video data. In a typical mission using our simulation or testbed, the UGVs can collect approximately 3.3MB/s. In the simulated mission this amounts to approximately 7GB. Given the max rate of the WiFi on our testbed UGV agents of 11Mbps the data transfer alone would take approximately 106 Minutes. Although the mission could still be accomplished under these conditions, it is highly impractical.

2.5 Simulation

The simulations to validate the proposed methodology have been carried out using the Robot Operating System [35] and the Gazebo [36] simulator.

All of the agents and their components were modelled as ROS nodes and C++ classes. This allows each module to be replaced in ROS with real hardware when available and paves the way for a smoother transition from the model to the real world implementation.

A Gazebo world shown in Figure 2.2 is generated from the occupancy grid maps previously discussed. The communication bitrates and connectivity for each channel are modelled in ROS nodes which calculate the communication channel qualities based on the agent's poses and the transmissivity and noise maps. These nodes produce connectivity graphs such as Figure 2.5 in real time so that the agents can only communicate over the channels that are connected at that time.

The Cloud service used in this simulation was an Amazon E2C c4.8xlarge instance

which utilized 36 cores and 60GB of RAM. The videos are segmented into 50MB files each of which is processed independently and can be run on parallel cores. This instance running the MVE toolchain could process a 50MB video block in approximately 300 seconds giving a 3D map processing rate of about 164kB/sec/instance. The typical simulation run produces 7GB of video. In order to process this video in a realistic time frame, a cluster of processors would be required. Although not implemented in our simulation, a cluster of 50 such processors would allow a complete map rendering in about 800 seconds or 14 minutes. So for purposes of comparison we will assume that the cloud processing takes 1000 seconds from when the data first arrives at the Cloud.

In order to compare the performance of various configurations, the simulation was run in three cases:

1. One BASE agent and one UGV
2. One Base agent and two UGVs
3. One Base agent, one UAV and two UGVs

As the mapping and imaging progresses, the results can be observed using ROS visulaization tools. The RViz tool displays the development of the Occupancy Grid and the Imaging Progrss Grid in real time. A snapshot of this progress for Case 3 is shown in Figure 2.7. The agents are shown as arrows, the frontier goals are shown as blue dots and the shades of grey are the map grids of the various agents. The agents each have their own maps which are different until they are merged over the UHF channel.

The three simulation cases were run 5 times each and the data was averaged across the runs. Figure 2.8 shows a sample of the discovery of the Occupancy Grid over time. The graph shows individual traces for each agent and the vertical steps

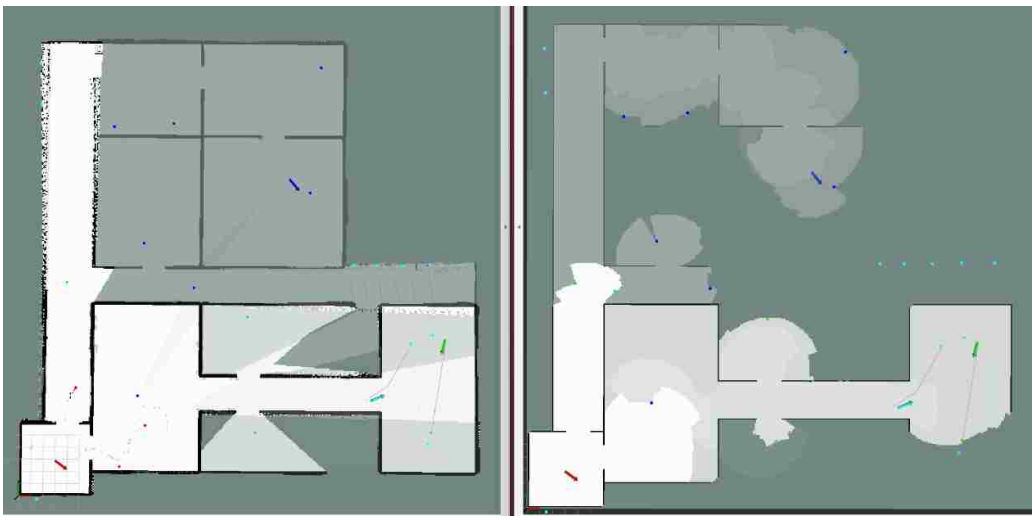


Figure 2.7: Navigation occupancy grid (left) and Imaging progress (right) maps during the process of exploration. Different shades of grey indicate maps of various agents.

indicate merges between agents. The important factor in this case is how quickly the Grid is transferred to the Base so that action can be taken based on it. The progress of the Occupancy Grid is summarized in Table 2.6.

Figure 2.9 shows the progress of the imaging of the environment. These plots show the percentage of the environment that has been imaged as it is known by each agent. Notice that although the overall progress between Cases 2 and 3 are similar, the knowledge of the progress by the Base is much more advanced in Case 3 due to the data mule operations of the UAV. Table 2.6 shows the summary of the time for each case to collect all of the images.

Figure 2.10 shows the progress of the transfer of image data to the base and therefore to the cloud for processing. The Cloud will start processing as soon as data is available. Since in Cases 1 and 2 the data is not transferred to the Base until the end, the cloud is idle until that point. However, in Case 3 the UAV brings the Base data much sooner and therefore the Cloud can begin its processing while the

Table 2.6: Simulation Results: Average of 5 Runs on each case (sec)

Case	Occupancy Grid	Imaging Complete	First Images To Base	All images to Base	3D Map Complete
1	2000	2400	2400	2400	3400
2	1400	1450	1500	1500	2500
3	600	1300	750	1400	1750

agents are still collecting images. The summary for the time to transfer to the Base and the Cloud computing is shown in Table 2.6.

2.6 Conclusions

We have developed and demonstrated an effective approach to mapping an unknown environment using a heterogeneous team of robotic agents implementing various intermittent communication channels. This approach has been validated in simulation and allows for the efficient and reliable exploration in the presence of environmental interference to communication channels.

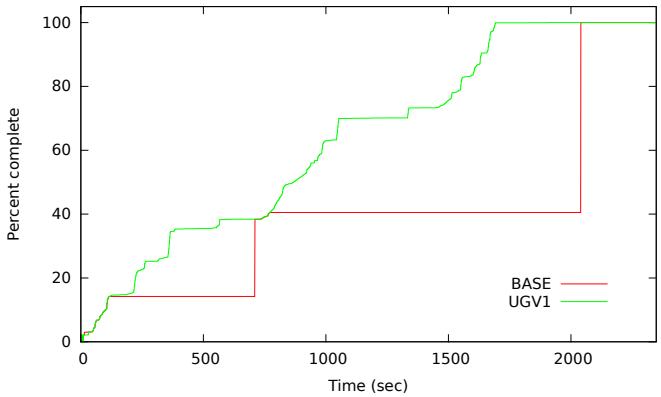
The advantages of incorporating this multi-channel system are numerous.

Speed The higher bandwidth of the OW channel allows the image merging to occur much faster than is possible over UHF RF.

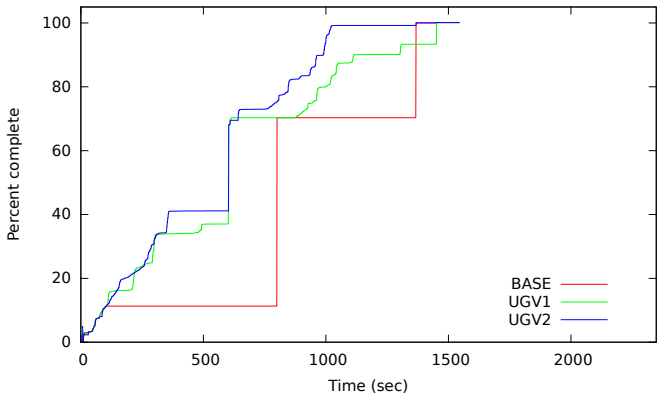
Reliability The OW allows for the mission to be complete in spite of radio interference or denial. The three separate channels provide the greatest possibility of completing the mission.

Robustness Since each agent operates autonomously between merges, the system can continue to function in spite of the loss of an agent. If an agent is lost or disabled, only the data it has collected since its last merge will be lost, and the other agents will ultimately explore its assigned areas.

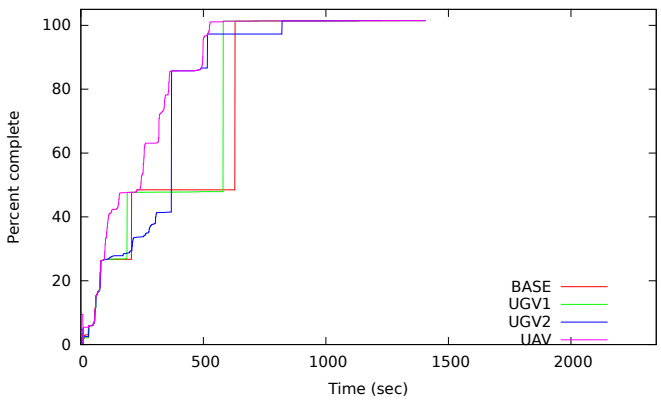
However, there will be several areas of challenge, especially with regards to accurate localization. Precise localization is required to perform the map merging and the simulation provides accurate poses for all of the agents. Motion tracing systems such as Vicon can provide fairly accurate localization, but only in a limited and controlled environment. Efforts within the MAST program are developing solutions to this localization problem which promise to allow us to operate in more realistic environments in the future.



(a) One UGV

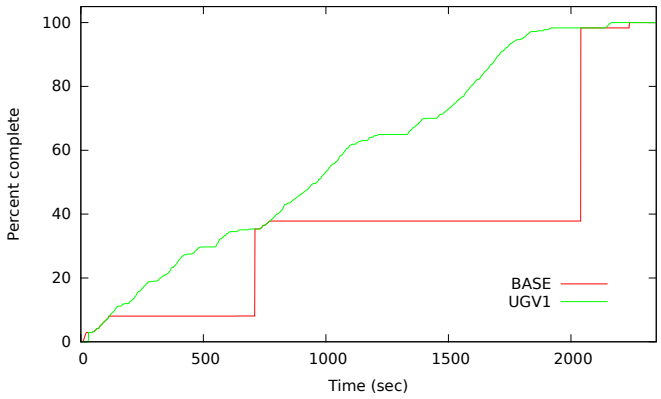


(b) Two UGVs

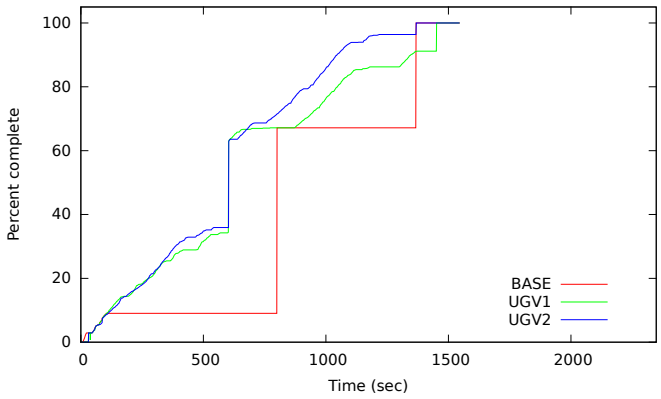


(c) Two UGVs and one UAV

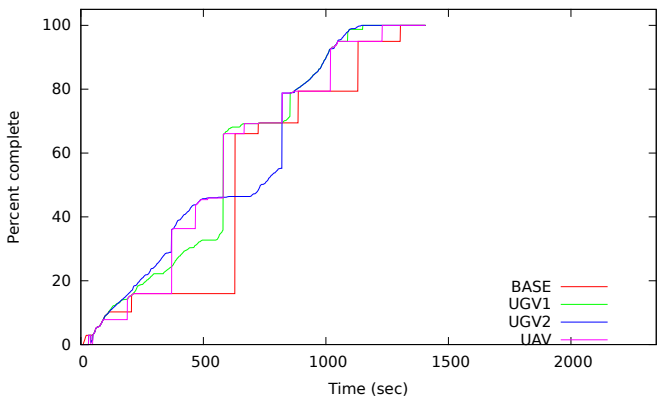
Figure 2.8: Exploration of the Occupancy grid over time



(a) One UGV

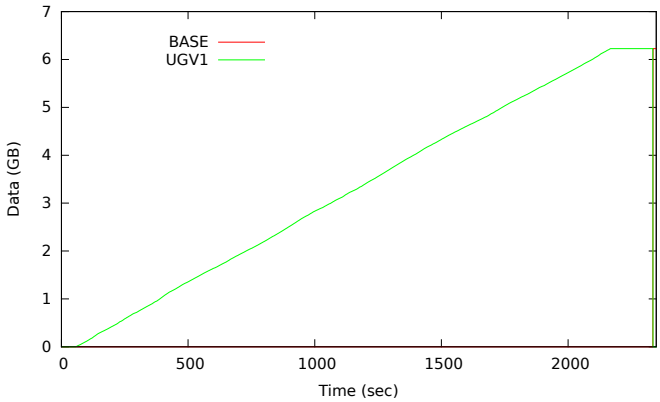


(b) Two UGVs

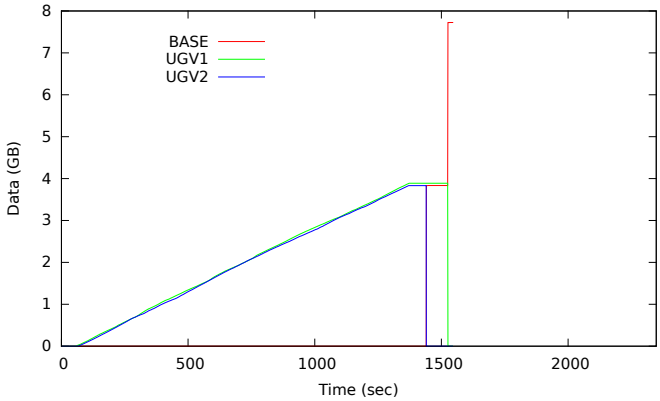


(c) Two UGVs and one UAV

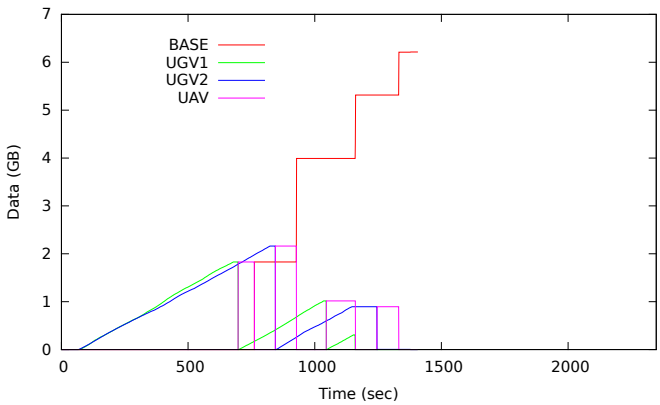
Figure 2.9: Image Capture Progress over time



(a) One UGV



(b) Two UGVs



(c) Two UGVs and one UAV

Figure 2.10: Video Data Transfer to Cloud over time

Chapter 3

Experimental Testbed for Cloud Based 3D Mapping Using Heterogeneous Robotic Teams

3.1 Experimental Testbed

We have developed a hardware testbed to reproduce portions of the simulations which were presented in Chapter 2. Since the simulation was done in ROS, the transfer of the major algorithms to hardware was straightforward. However, the very large physical scale of the simulated area in which to conduct the test makes a full realization impractical.

3.2 Agents

3.2.1 UGV Agents

For UGV agents we have developed the miniROaCH shown in Figure 3.1 which has a camera, WiFi and OW interfaces. In our lab facility we will develop a demonstration showing several UGVs imaging an area and transferring their data to a base station through a UAV operating as a data mule using Optical Wireless communication for the data transfers. The operation of the testbed will be similar to the simulation except that it will operate under the Vicon system to provide localization. This restricts the area of operation, but will provide a sufficient environment to demonstrate the effectiveness of the coordination algorithms and of the optical wireless communication channel.

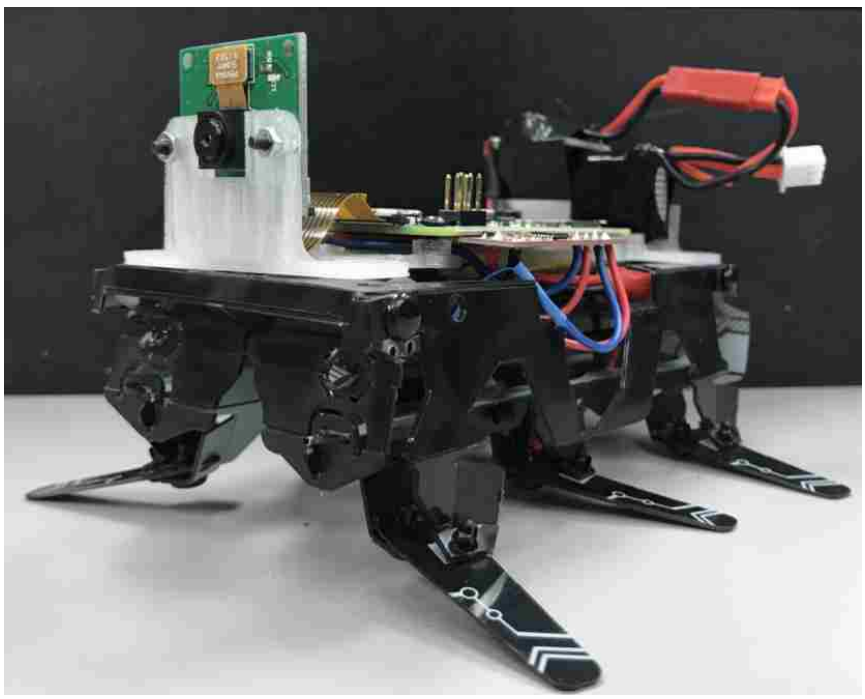


Figure 3.1: miniROaCH UGV test agent

3.2.2 UAV Agents

For the UAV agent in our testbed we will use the Astec Hummingbird Quad copter with the OW system attached to the bottom as shown in Figure 3.2. The UAV will be controlled using the Vicon system as its localization. It will function as a data mule, flying to the UGV agents, collecting their data and then flying to the base agent to deliver the data.

Figure 3.2: UAV agent with OW transceiver attached

3.2.3 Base Agent

The Base agent is a Turtlebot with the OW system installed on its top platform as shown in Figure 3.3. A laptop computer was used to provide the processing, communication and storage for the Turtlebot.

The Base agent is stationary receives the video data from the UGVs via the UAV

acting as a data mule. As the data is collected, the Base agent will dispatch the UAV to collect the images and data from the UGVs and will receive the image data from the UGVs via the UAV acting as a data mule. The Base station will then perform any necessary processing and then upload the data to the Cloud processing system.

Figure 3.3: Swarmie used as a base agent

3.3 Cloud computing resources

For small demos and diagnostic work, the cloud computing resources will be provided by a Dell T720 server in the lab connected to the base agent over an Ethernet connection. This allows us to transfer the data and view the results locally.

For larger jobs and to demonstrate the full cloud-based solution, the processing was deployed to Amazon Web Services EC2 computers. These computers allow scaling of the number and power of the machines and also allow many parallel machines to be run simultaneously.

There are many 3D modelling tools available. For our research we chose two open source packages.

The Multi-View Environment (MVE)[37] from the Technische Universitt Darmstadt Graphics, Capture and Massively Parallel Computing Lab creates 3D maps from a sequence of overlapping images. We found it to perform well for small sets of images, but it requires extensive overlap and unique visual features in order to determine the camera's location in the world.

The Open Drone Map project [38] uses many of the same processing packages from the MVE, but it allows the user to include the position of the camera in the data. This removes some of the burden from image matching algorithms since the camera pose is known. However, since the program is written for UAV video, the camera poses are in GPS coordinates so a script was written to convert the Vicon poses into relative GPS coordinates.

Both of these programs produced good results, but were unable to handle very large data sets. For large area maps, the image data must be broken into smaller areas and then the resulting maps stitched together to form the final result. Breaking the data into smaller sets is also desirable for data transmission and to facilitate parallel processing so the data files are segmented into 50MB blocks.

3.4 OW Transceiver

For the experiment, a commercial IRDA module (Figure 3.4) was used. Although it only operates at 4 Mbps, it interfaces over USB and uses standard Linux network drivers which allowed it to be easily integrated into the agents.

The IRDA transceivers transmit ping messages periodically and automatically detect when another transceiver is in range. Once in range, the receiving agent

Figure 3.4: Commercial IRDA USB transceiver used in the testbed

negotiates opening a network socket connection between the agents and then uses the Linux secure copy (scp) program to transfer the data files. The UAV maintains its position over the ground agent for the duration of the file transfers as shown in Figure 3.5.

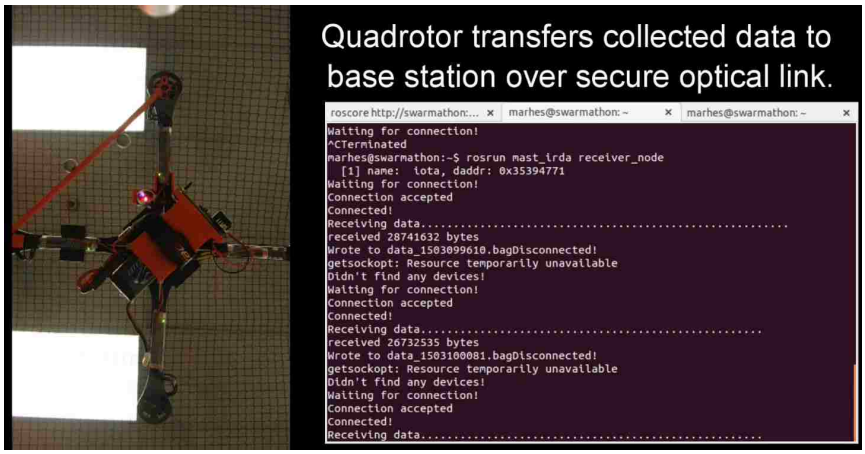


Figure 3.5: UAV agent transferring data to base over OW channel

3.5 Arena

The arena is in the MARHES lab at the Univeristy of New Mexico. It consists of a 4 by 4 meter area shown in Figure 3.6 under a Vicon motion capture system.

The Vicon system provides all of the agents with their poses through a Rosbridge websocket interface. The floor of the arena is made of corkboard in order to provide the UGVs with a smooth surface with suitable traction and is bounded by movable plastic walls. Visual markers such as colored stickers and Apriltags were placed around the area in order to aid in the visual 3D mapping.

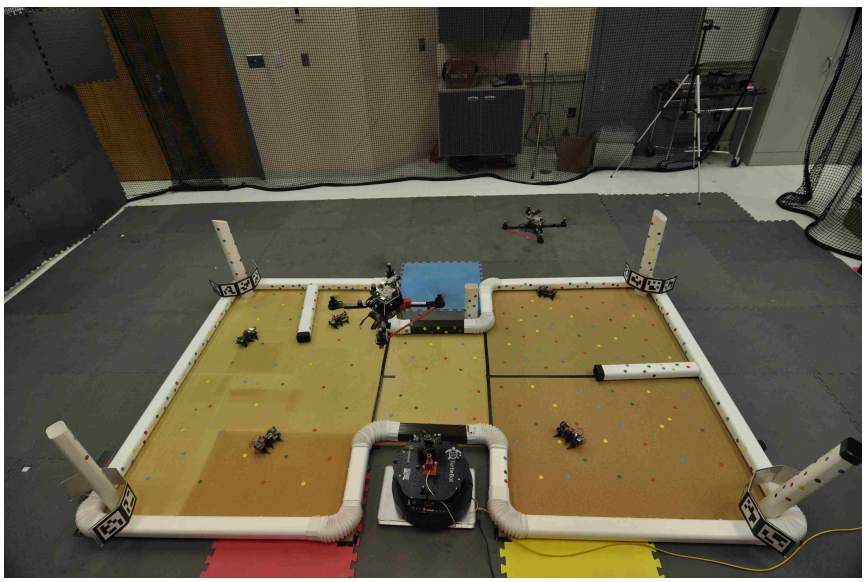


Figure 3.6: Testbed arena in the UNM MARHES lab

3.6 Operation

The operation of the testbed follows the same basic pattern as the simulation, only simplified due to the size and hardware constraints.

The UGVs are first dispatched to record video of the area. Since these particular UGVs move by flapping their legs, the video is highly blurred when they are in motion, so they are required to stop often in order to record usable images. Once they have collected enough images to cover the arena, they stop and wait for the

UAV to come and collect the data.

Since the miniROaCH platform is extremely lightweight, it was not able to perform with the additional weight of the IRDA transceiver. For the purposes of our demo, we simulated the transfer of the data by using WiFi. The UAV flies to position over one of the UGVs and then receives its data. It then flies into OW position over the Base agent. Once the base agent discovers the UAV in OW network range, it establishes the OW network connection and then receives the data. While the data is transferring, the UAV must maintain position over the base agent until the transfer is complete. If the UAV wanders enough to break the OW connection, it will re-establish and continue as long as the interruption is less than 5 seconds.

Once the UAV has finished transferring the data from the first UGV, it flies to the other UGV and repeats the process.

As soon as the Base agent has the video data from the first UGV, it begins to upload it to the cloud servers. The cloud server then begins the 3D reconstruction of the map. Once the 3D reconstruction is complete, the 3D map (Figure 3.7) can be downloaded for analysis.

3.6.1 Future Work

Our demo shows all of these elements in action, however the level of automation needs to be improved. The data transfer to the cloud, the cloud processing and the reconstruction of the entire map from segments is currently being done manually. Development of a higher speed, lightweight OW transceiver is also necessary for the UGVs to be able to transfer their data to the UAV over OW.

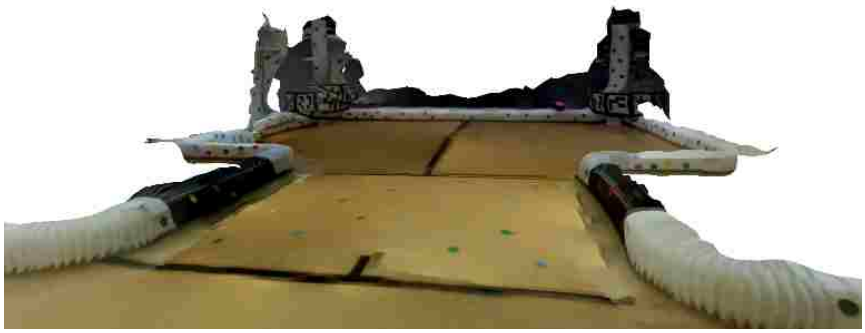


Figure 3.7: Resulting 3D model of testbed area

Chapter 4

The SIPI I-C-MARS Robotic Educational Platform

Computer programming skills are a critical necessity for today's students, but maintaining student interest in programming and engineering courses is challenging unless the theory is accompanied by engaging, hands-on applications. Additionally, many schools, especially those in underprivileged areas, lack the resources and personnel to develop or implement such applications. The Southwestern Indian Polytechnic Institute (SIPI), through the support of a NASA grant, has developed an integrated teaching program where students from middle school through the college levels can learn programming and robotic design from the most basic introductory level to advanced embedded computing, hardware and webpage design at little or no cost to the participating schools and with minimum burden to the teachers. The centerpiece of the program is the indoor "Mars Yard" which is a SIPI facility that allows remote operation of robots in an indoor environment to simulate remote space missions. Beginning with simple Arduino-based robot kits, students in the middle and high school levels are introduced to programming and robotics using an easy to follow curriculum. As they advance, students can remotely access the Mars Yard and

perform pre-determined missions on real or simulated rovers. At the advanced high-school and college level, the students proceed to design, build, program and test their own robots and sensors and develop custom missions. The educational platform described in this paper is being implemented at SIPI and affiliated local high schools with tremendous results.

This work was funded by the Southwestern Indian Polytechnic Institute, the NASA TCU-ELO (Grant NNX14AJ99A), the North Carolina Agricultural and Technical State University TECHLAV program (Award 210158A), and the Regents of New Mexico State University, under the New Mexico Space grant Consortium (NMSGC) Program (Award A01723).

4.1 Introduction

There is no question that STEM education is critical to the future of our students and workforce. As technology advances, computer programming skills are becoming a necessity in almost all fields. However, teaching programming and other advanced technologies is very difficult, especially in underprivileged areas and specifically among Native American students [39]. Teachers in community colleges such as the Southwestern Indian Polytechnic Institute (SIPI) are often faced with the dilemma of only having a few short courses to teach these subjects, often to students with no background in them at all. Additionally, teaching these subjects is often difficult due to the complex nature of the topic and the required technological resources required.

Recognizing this problem, the faculty at SIPI and the NASA Minority University Research and Education Project (MUREP) formulated a plan to provide an engaging STEM educational experience at the Community College level and also a program reaching down into the feeding middle and high schools in order to build the prerequisite foundation to better prepare the students before they enter the college level.

SIPI is a two year undergraduate institution serving Native American students from all over the nation. As with most technical education programs, we find it difficult to maintain student interest in complicated subjects such as calculus, physics and programming. These courses are traditionally textbook based lectures presenting all of the necessary theory but with little practical application. Without a clear, tangible objective, the courses quickly degrade into memorization exercises and the students become discouraged from pursuing further education in a field for which they do not see an immediate use. Many recent efforts in the education world have recognized and attempted to address this issue. The Next Generation Science Standards (NGSS) [40] directly addresses this problem by promoting "meaningful learning" experiences. However, these standards are difficult to achieve in a real classroom with limited physical and faculty resources. In order to provide an environment in which these experiences can be achieved, SIPI has developed the Intelligent Cooperative Multi-Agent Robotic System (IC-MARS) program which uses a unique NASA-inspired robotics facility to provide interactive educational experiences for college, middle and high school students.

SIPI's Advanced Technical Education Department offers core courses in math, science, engineering and programming. In order to ground these courses in practical experiences, the students all participate in team projects related to the Mars Yard. Additional courses, beyond the core requirements, are also offered which use the Mars yard to supplement the core courses. In the past two years courses such as ENGR290: Computer Programming using ROS have been taught which use the robots and sensors in the Mars Yard to give the students hands-on programming experience in which their programming assignments are to develop code that is actually deployed on the rovers. The utilization of the Mars yard in courses and team projects at SIPI has been a great success with more than 60 students participating over the last 2 years. The program has been externally evaluated by Dr. Janet Gordon who concluded:

> STEM by nature is a philosophy grounded in an inter-disciplinary perspective. SIPI's IC MARS Program has successfully created a STEM inter-disciplinary program that ignites interest, boosts retention and places a desire in young Native American students to pursuit a STEM-related career, especially with NASA.
>
> Based on qualitative data from student focus groups, the IC MARS Program created an environment that offered opportunities for students to safely work out of their comfort zone and engage in project-based learning that developed the knowledge, skills, attitudes, and proclivities needed to be successful in their academic and professional career. Student supports, such as mentor and mentee relationships, as well as caring professors and instructors, helped provide emotional support which assisted in alleviating stress from loneliness while students were away from family, thus boosting retention.
>
> Qualitative data strongly suggest the IC MARS Project is successfully meeting its goals and objectives to: 1) build a community of learners where students feel supported academically and emotionally leading to increased retention, 2) develop skills and Habits of Mind that promote academic and career success, 3) promote awareness and encourage the pursuit of educational disciplines critical to NASAs and the Nations future STEM workforce and 4) spark interest and drive in SIPI students to transfer to a four-year university to pursue a degree in STEM. [41]

However the effectiveness of such a program is often limited by the lack of prerequisite knowledge of the incoming students. In order to increase the impact of the college program we also need to reach down into the middle and high schools that prepare the students for their education at SIPI. The enrollment and success rate of students in STEM classes, especially math and physics is always a problem,

especially in underprivileged areas. SIPI has established a pilot program with several regional high schools in which the physics classes are renamed and reorganized into robotics classes where the principles of science and programming are taught through their application to robotics. Through these partnerships, SIPI provides the robot hardware, curriculum and training to the local schools. The results so far have been extraordinary with four schools and over 150 students already participating [42] and expansion of the program is underway.

4.2 Mars Yards

The centerpieces of the SIPI IC MARS program are the Mars Yards. We have developed these Mars Yards as tele-operated facilities where students can design, build test and operate robotic rovers locally or through the Internet from anywhere in the world. Most of our classes and team projects focus on developing some aspect of these yards. By broadening the scope of the program to development of these facilities, we have opened the reach far beyond the typically isolated computer or electronics class and instead provide students the opportunity to focus on the areas of their own interest and still contribute to the overall objectives. For example, students interested in construction can design, build and landscape the physical facility. Those studying computer aided drafting produce blueprints for the construction and design the models for the robot simulations. And the list goes on to include rocketry, 3D printing, smart lighting, web page design, documentation and even growing food for astronauts. By creating a flexible project with so many facets we can leverage many seemingly unrelated grants and projects into one larger objective to which everyone can contribute. There are two Mars Yards at SIPI: the Mini Mars Yard and the Main Mars Yard. Both are equipped with full coverage wireless networking (designed and installed by students) and are hosted by a Linux web server which provides Internet

based communication with the rovers. Both yards are similar in function, the only difference being the landscaping and scale.

4.2.1 Mini Mars Yard

The Mini Mars yard shown in Fig 4.1 is a classroom in the SIPI Science and Technology building which is used to develop rover technologies and software. It provides an environment where the students can interact directly with the rovers and can reprogram both the rovers and the web page interfaces as necessary. It is primarily used as a testing ground for technologies that are to be deployed in the Main Mars Yard.

4.2.2 Main Mars Yard

The Main Mars Yard shown in Figs 4.2-4.4 is a 3500 square foot free-standing building on the SIPI campus in Albuquerque, New Mexico. It houses a distance learning classroom which is soon to be outfitted with a complete audio and visual remote classroom system using the Zoom web conferencing system. The Mars Yard consists of a roughly 50x50 foot area which is landscaped to resemble a Martian surface. The Main Mars Yard's design was done entirely by student teams.

4.2.3 Cameras

The activities of the rovers in the Mars Yards are monitored by many cameras installed throughout the area. These cameras are actually Raspberry Pi based units which are running the the same software as the ones on the rovers. By using the same hardware for the cameras and the rovers, we maintain a standard interface for

the software development and the user interface. Most of the cameras are stationary, but some students have been developing a servo driven pan-tilt unit which will be used on selected cameras. The camera views can be selected on the web page so that any user can choose any camera view.

4.2.4 Webpage Interface

The users interact with the Mars Yard and the rovers through a web page which is hosted by a Linux Apache Webserver in each Yard. The servers are standard LAMP servers running Ubuntu 16.04. The custom web page and database design is also part of the educational curriculum as the students are encouraged to design and implement their own web pages to support custom sensors or missions.

The server hosts a fully interactive web page from which students and observers can log in, view and interact with the rovers from any Internet connected computer. Users are assigned a user name and password and can then log on at any time. Multiple users can access the system simultaneously allowing different schools to cooperate, observe or compete with each other. Users are assigned different levels of access according to their needs and each user can create their own robots and missions.

The user web page is shown in Fig 4.5. The pages can be accessed at sipi-i-c-mars.org.

4.3 Rovers

4.3.1 Roadrunners

The outreach program to the middle and high schools uses a simple Arduino based rover that allows students to learn basic programming skills and then advance to more sophisticated software and sensor designs. The roadrunners consist mostly of commercially available components but also include custom 3D printed components which were designed and built by SIPI students Brandon Ray and Tomczak Billie. The construction and programming curriculum was developed at SIPI by the same students and then expanded into a full course by Bernalillo High School teacher Katrina Lake. These course materials are freely available on the SIPI I-C-MARS webpage [43]. The Roadrunners are provided to the schools as the complete kit shown in Fig 4.6 with comprehensive assembly and programming instructions. SIPI and the affiliated High schools also provide training for the teachers of these courses. The courses teach many aspects of physics, mechanics and programming while the students build and program their own rovers and then perform educational experiments with them. The assembled roadrunner rover is shown in Fig 4.7.

These roadrunners have simple Arduino Uno processors shown in Fig. 4.8 which is a standard educational platform commonly used to introduce students to programming. Although limited in capabilities, the Arduinos are very inexpensive and the free software development environment they provide makes it easy for students to get started. The Arduinos also support a wide array of affordable sensors such as range detectors, accelerometers, gyroscopes, compasses and line following optical sensors. This wide array of sensors provides the teachers with great flexibility to choose lessons that are appropriate for their classes.

4.3.2 Mini Mars Yard Rovers

The simple Arduino on the roadrunner platform is easy to program and use, but very limited in capability. The Mars Yard rovers use the same Arduinos for sensor and motor interfaces so that the students who have completed the Roadrunner based courses can immediately begin work on the rovers. The only difference being that the rover Arduinos must now interface with the rest of the world. This interface is done using the Robot Operating System (ROS) environment [44].

ROS is a standard and well supported software environment which provides many features which would be otherwise impossible to develop on our own. Primarily it provides a communication network so that all of the rovers and cameras have a common interface so that the students can learn how to program one interface and then easily apply it to many more. The Arduinos on the rovers run student-written ROS programs to collect sensor data and control the motors of the drive wheels, gripper and other attachments. They communicate with the more powerful Raspberry Pi computers shown in Fig. 4.10 which run the Ubuntu operating system and can handle more complicated code.

The Raspberry Pi3 computer is a powerful but affordable platform for robotics. It supports Ubuntu Linux and ROS which makes software development easy. It communicates the the Mars Yard servers through a built in WiFi adapter and communicates with the on-board Arduino through a USB connection. It has a built in high resolution camera which is used for navigation and is also used as the Mars Yard stationary cameras. It also supports standard USB cameras which enables the students to add cameras to monitor important features such as the gripper or other attachments. The current model of rovers used in the Mini Mars Yard shown in Fig 4.9 are small and affordable, but contain powerful processors and provide the basic platform for the SIPI programming courses. They were also designed and built by

students and consist of a Raspberry Pi3 computer and camera, Arduino 101 and motor shield for interfacing the motors and sensors, custom designed gripper and a web camera. Since they are cheap and easy to build, the Mini Yard rovers provide the students with a flexible platform to experiment with new software and hardware designs without worrying about damaging expensive units or interfering with the Main Mars Yard operations.

4.3.3 Main Mars Yard Rovers

The rovers used in the Main Mars Yard shown in Fig 4.11 extend the same electrical design as the Mini Mars rovers, but utilize more advanced and rugged all-terrain vehicles in order to be able to navigate the Martian landscape of the Main Mars Yard. They have the same Arduino, Raspberry Pi and motor interfaces as the Mini Mars rovers so that the students can use the exact same hardware and software designs on both. This allows the education progression to continue from the first introduction to programming on the Roadrunners to the more advanced Mini Mars Yard Rovers to the final platforms in the Main Mars Yard with the design from each step leading directly into the next.

The Main Yard rovers are based on the Gears Educational Systems Surface Mobility Platform (SMP) shown in Fig 4.12. The SMP is an off-the-shelf platform which allows the students to focus on developing the payloads and software. They also use a Raspberry Pi3 processor and can be fitted with multiple attachments such as an arm, drill, scoop and spectrometer. They are capable of carrying considerable payloads which allows students to design custom hardware for specific applications.

As an example of their use in the classroom, the attachments for the first set of rovers were designed and built by a team of students at New Mexico State University. The students spent one semester designing the lift mechanism, drill attachment and

scoop attachments shown in Fig 4.13.

4.3.4 Swarmie Rovers

Also in the Main Mars Yard are the Swarmie rovers, shown in Fig 4.14, from the recent University of New Mexico and NASA Swarmathon program [45]. These rovers have powerful Intel NUC on-board processors and are capable of more sophisticated programming operations but are more limited in their mobility than the Main Mars Rovers. They are used to develop more computationally intensive navigation and vision processing algorithms.

4.3.5 Expandability

The interface between the rovers and the web server is a modular design using standard ROS interfaces. Therefore any ROS-programmed rover can be used in the Mars Yards with minimal modifications. In the future, the students will be adding additional rovers and continuing to increase the features and capabilities of the existing ones.

4.3.6 Virtual Mars Yards

Since the rover software is based on the Robot Operating System, we have also used the ROS Gazebo [36] simulator to develop a complete virtual edition of the Mini Mars yard. Thanks to the work of a student team in Fall 2016, the Mini Mars Yard and the Mini Mars Yard rovers have been modeled and can operate in an on-line simulation. The student team worked under support from the New Mexico's Experimental Program to Stimulate Competitive Research (EPSCoR) [46]

to develop 3D models of objects from video. The resulting photo-realistic simulation of the Mars Yards allows more students to interact with the rovers at the same time and provides a virtual testbed where they can try out new programs and algorithms without needing access to the physical rovers. Since the ROS code the students write will run on both on the hardware and the Gazbeo simulator, the same programs they develop in the virtual world will run on the physical hardware. The user on the web page will be able to choose to run a virtual or physical mission, but otherwise the behavior will be similar.

An example of a simulated Mini Mars Yard mission is shown in Fig 4.15. The simulation for the Main Mars Yard is still in progress.

4.4 Curriculum

The curriculum is broken into four phases:

4.4.1 Phase 1: Middle and High Schools

Phase 1 consists of a middle and high school curriculum, shown in Fig 4.16 and 4.17, provides a course that takes the students from assembling a rover from a kit to programming those rovers to do obstacle avoidance and line following. The curriculum includes slides and other classroom materials as well as quizzes and assignments. The result is a turn-key solution whereby, with a simple training session, any teacher can run the entire course at their school with support from the SIPI staff and other affiliated teachers. SIPI has hosted two such training sessions in 2016 with the participation of teachers from the Native American Community Academy, Zuni High Schools, Bernalillo High School, Wallatowa High Charter School and Twin Buttes High School.

4.4.2 Phase 2: Virtual Mars Missions

Phase 2 consists of simulated robotic environments in which the students can log into the IC MARS system and perform missions. This allows the students to learn robotics, ROS and basic programming without the need for expensive hardware.

4.4.3 Phase 3: Physical Mars Missions

In Phase 3, the students work with the actual Mars yards, either physically or remotely. They are given pre-specified hardware and mission requirements and they must operate the rovers to complete the missions.

4.4.4 Phase 4: Rover Development

Phase 4 is for advanced high school or college students and allows them to create their own hardware and missions.

4.4.5 College Courses

The power of the Mars Yard project is that it provides a platform on which almost any aspect of STEM education can be realized. The following is an sampling of some of the course work and projects that have been generated by the Mars Yard at SIPI:

ENGR290: Computer Programming with ROS (summer 2016)

The purpose of this course was to introduce students to a wide variety of computer programming languages and platforms with clear applications to the robotics field. The course began with the most basic C++ programming on the Arduinos where

the students each wrote a program to read data from a sensor. The course then advanced to teaching the basics of the Robot Operating System (ROS) where they learned to convert the sensor data into usable measurements and bring them into the ROS system. They then advanced to programming ROS under Linux where they used the Raspberry Pi3 to write a ROS program to read the Arduino sensor data and transmit it to the web server. The course completed by introducing the students to web server programming using HTML, PHP and JavaScript where the students wrote custom modules on the web server to display the data from their sensors. In a one trimester course, the students went from little or no programming experience to writing code in four different languages. Since they could physically see the results of each programming operation on the Mars system, the students stayed engaged and enthusiastic throughout the course despite the complicated material.

ENGR285 Engineering Projects

There have been too many team projects to list here, but some of the current activities include:

- Participation in the NASA Swarmathon where students use the ROS programming skills learned in the Mars Yards to program NASA rovers in a competition. The SIPI team won third place in the 2016 competition and is already writing the code for the 2017 edition.

- Design and construction of the yards themselves where the students have done everything from wireless network design and installation to landscaping and construction

- Designing and 3D printing of grippers and other robot components using SIPI's manufacturing and CADD classes and labs

- Developing virtual reality 3D tours of the yards
- Developing an augmented reality sandbox [47] to make 3D models of landscapes
- Participating in The Colorado Space Grant RockOn rocketry workshop in 2015 and 2016 [48] and now the The First Nations Rocket Launch Competition [49].

4.4.6 Missions

In addition to the courses involving construction and programming of the rovers, SIPI has also developed a set of Mars Missions that can be used by any school with Internet access to the Mars Yard. These missions consist of varying levels of complexity from simple remote control of the rovers to complicated path planning and autonomous missions. These missions can be integrated into existing courses or used as stand-alone workshops or exercises. Students are also encouraged to work with the SIPI students to develop their own missions.

SIPI student teams have developed a set of 10 simple missions which students can access either at SIPI or remotely through the Internet. These missions start by demonstrating basic abilities to login to the system and then progress through driving a rover to perform several simple tasks. As the rover capabilities are increased, the missions will be expanded to include autonomous navigation and operations more closely resembling actual Mars missions.

While there are already many on-line remote controlled robotics projects, what distinguishes the Mars Yards is our intent to simulate actual Space missions. Due to the vast distances to Mars, remote control of a rover is not possible since there is significant delay in communications and limited windows of communication due to the orbits and rotations of Mars and the Earth. Therefore the rovers must behave autonomously over short time intervals. The Mars Yard communication is deliber-

School	Fall 2014	Fall 2015	Fall 2016
SIPI	28	60	45
Bernalillo High	12	48	48

Table 4.1: Student Participation in STEM Classes

School	Enrollment
Bernalillo High	48
Native American Community Academy	70
Zuni High	40
Twin Buttes High	7
Wallatowa High	15

Table 4.2: Student Enrollment in NASA Technologies Course 2016-2017

ately delayed with an adjustable latency so that students can get an appreciation of the difficultly in controlling rovers at a large distances. In the Mars Yard missions, the operator can specify the number of seconds to delay communications in each direction. This latency can be anywhere from one second to hours depending on the particular scenario. With the introduction of even a several second delay, the students quickly discover that it is impossible to simply drive the rover with a joystick with video feedback. They therefore must write sophisticated programs to give the rovers a sequence of operations to perform autonomously. They write a script, upload it and then wait for it to complete or fail. Then they must analyze the results and write the next script to upload. This closely emulates the actual NASA Mars operations and teaches the students valuable lessons in planning, autonomous operation and fault tolerance and analysis.

4.4.7 Results

SIPI is seeing increased participation and persistence among students in our STEM classes associated with grant activities. At our first Partner High School, Bernalillo

High School (BHS), we saw 48 students enrolling in two new NASA Technologies classes in September 2015 (Table 4.1.) Of these students 36 completed the class in the Spring 2016. In the Fall of 2016, 48 students were enrolled in the NASA Technologies I and II courses at BHS, and 34 completed the classes in Spring 2017. Beyond this, both at SIPI and BHS, we are seeing increased interest in STEM classes on the part of students who, pre-grant, did not think that they would be interested in STEM courses or careers. Sixty-four SIPI students, including SIPI graduates at the University of New Mexico (UNM) were involved in our Summer, Fall and Spring 2016-17 VIP ROSE-STEMS Teams. Adding the 180 high school students who were enrolled in our Partner High School Classes (Table 4.2), we involved 244 students with NASA-related STEM projects and educational content over the final Year of the Grant. To this total are added students at the Middle Schools associated with our Partner High Schools over the course of the year. These student numbers, while impressive, are only one part of the story. The program's effectiveness at SIPI was externally evaluated by Dr Janet Gordon. Her report [42] found that The IC-MARS Program exposed students to authentic practices and project teams supported deeper learning, collaboration, problem-solving and real-life application of their new knowledge. Students recalled prior knowledge and experiences that supported their sense-making and how all this new knowledge fits into their own life, family and community.

Despite students arriving at SIPI with varying levels of academic preparation, and future post-SIPI plans, each student found meaning in the projects and collaboration with peers, faculty and mentors that provided the initial spark or provided further validation to the student that STEM field is within their reach academically and professionally. Nearly all of the students either expressed their intent to continue their education at four-year University in a STEM discipline or had already been accepted to a four-year university. Notwithstanding that most of the students interviewed had not initially perceived STEM in their future. The intentional com-

munity of learners, comprised of students and faculty, has undoubtedly bolstered students' individual investment at SIPI resulting in improved motivation to persist and know that a STEM professional career is within their reach. In conclusion, the IC-MARS Program has had overwhelming success in meeting all of their intended program goals and objectives.

Figure 4.1: Mini Mars Yard

Figure 4.2: Main Mars Yard Exterior East

Figure 4.3: Main Mars Yard Exterior South

Figure 4.4: Main Mars Yard Interior

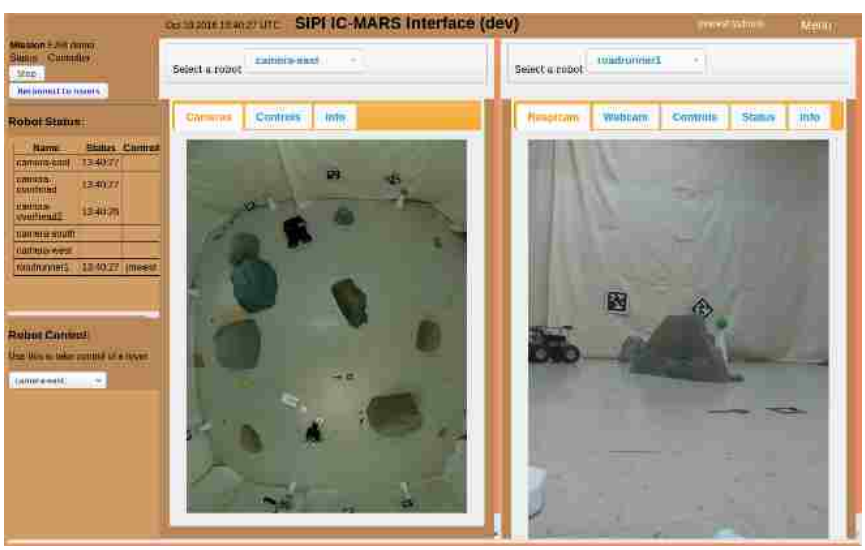


Figure 4.5: Web Page for Remote Access to the Mars Yards

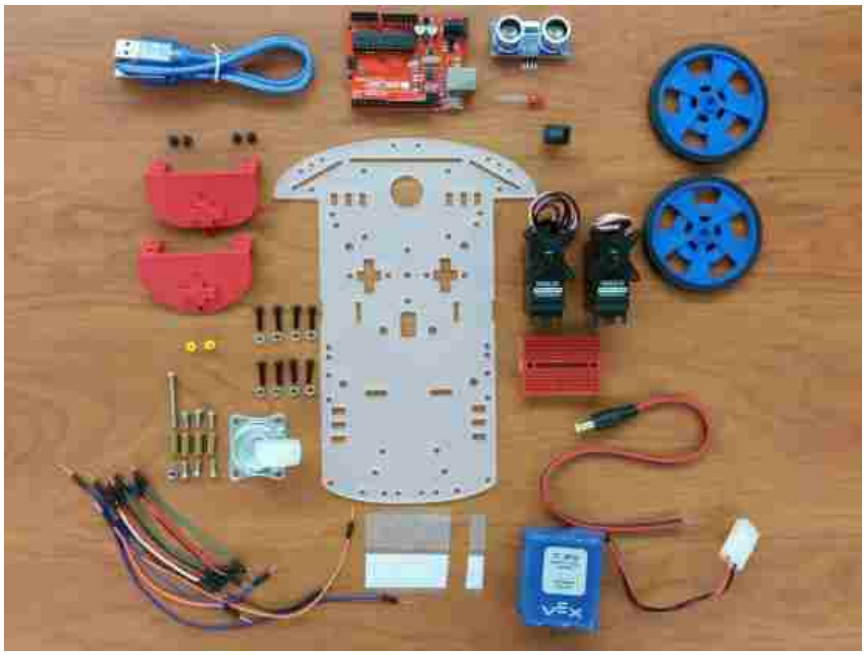


Figure 4.6: Roadrunner Kit

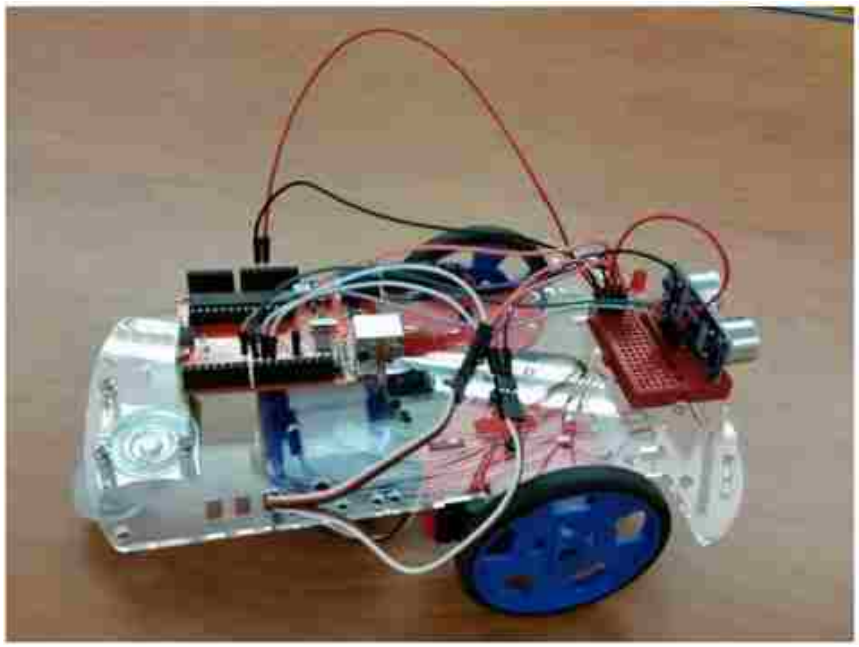


Figure 4.7: Roadrunner Rover Assembled

Figure 4.8: Arduino Processor For Roadrunners and Sensor Interfaces on Rovers

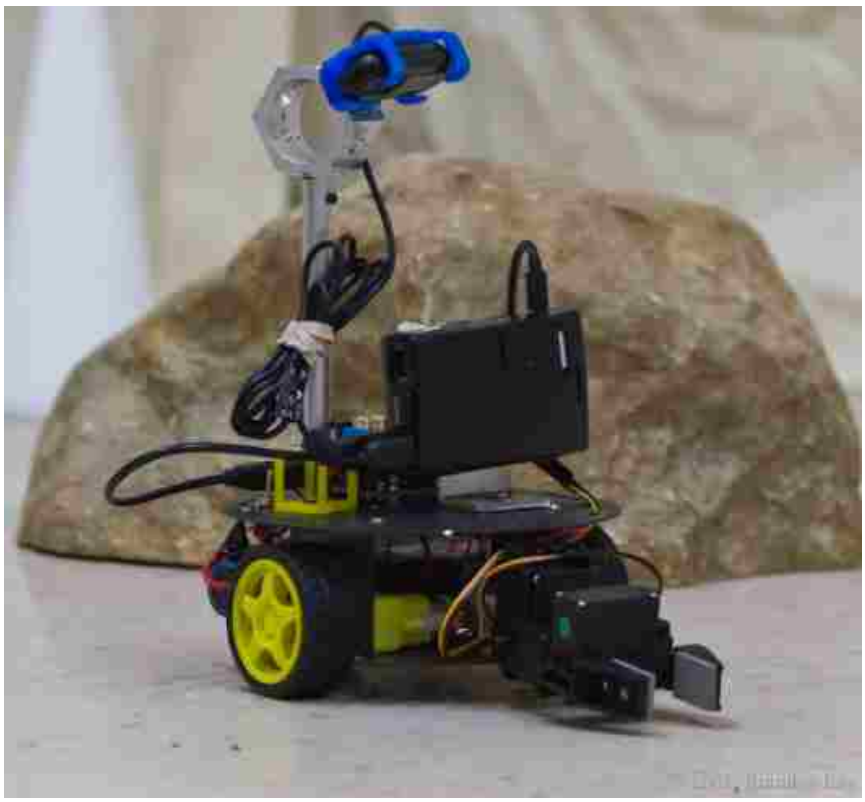


Figure 4.9: Mini Mars Yard Rover

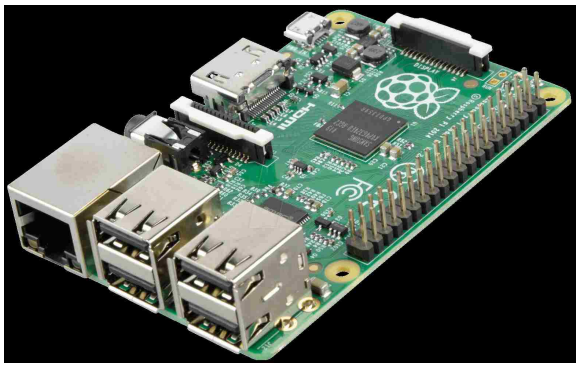


Figure 4.10: Raspberry Pi Processor Rovers and Cameras

Figure 4.11: Main Mars Yard Rover With Arm and Drill Attachments

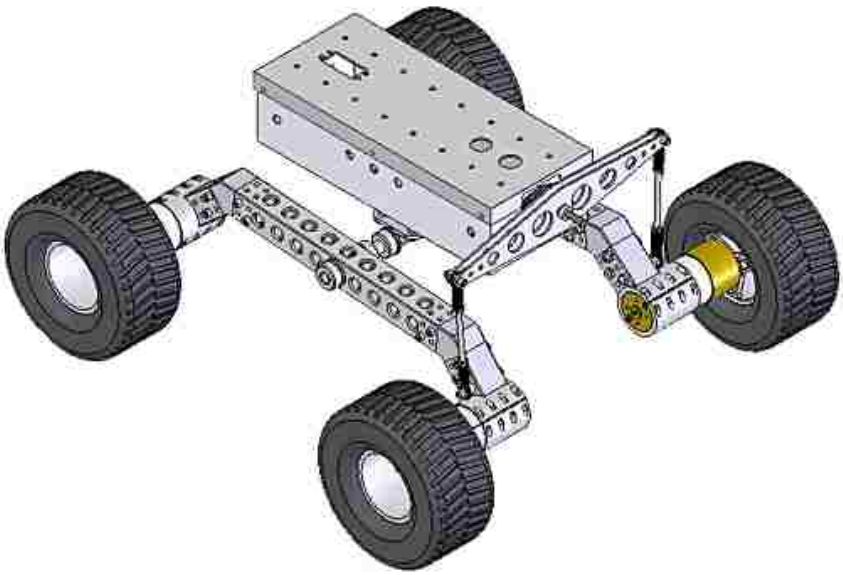


Figure 4.12: Gears Surface Mobility Platform

Figure 4.13: Scoop and Drill Attachments With Lift Mechanisms

Figure 4.14: Swarmie With Gripper

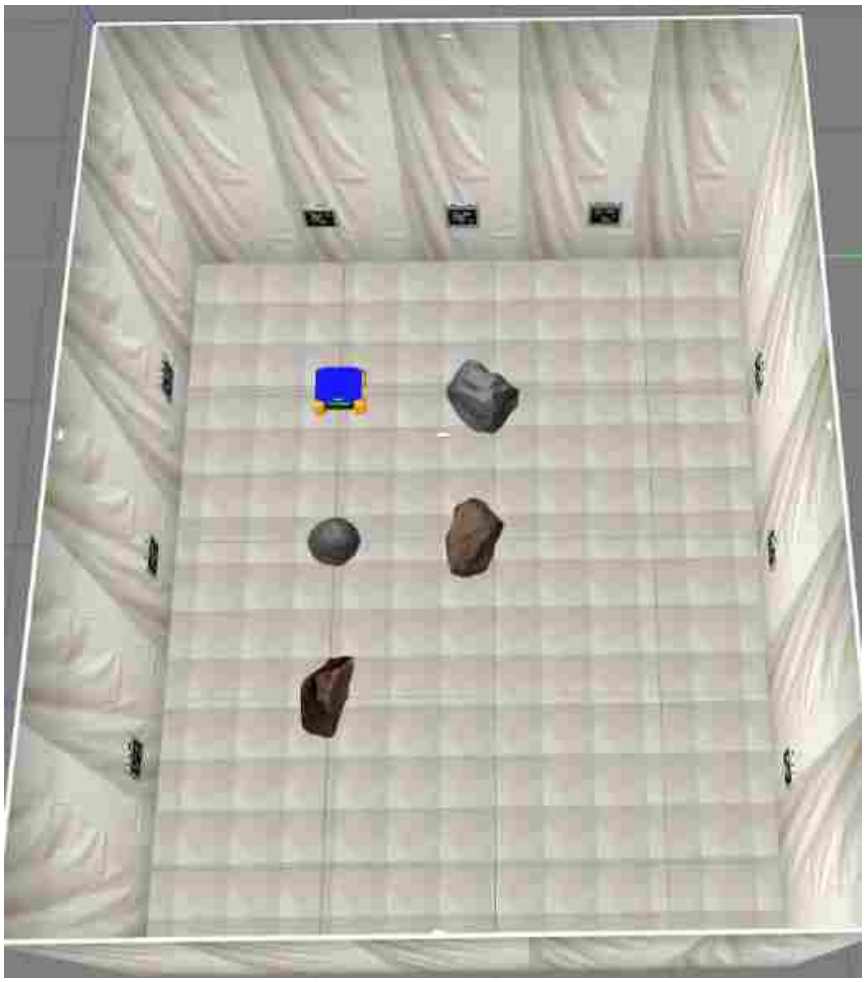


Figure 4.15: Mini Mars Yard mission simulated in Gazebo

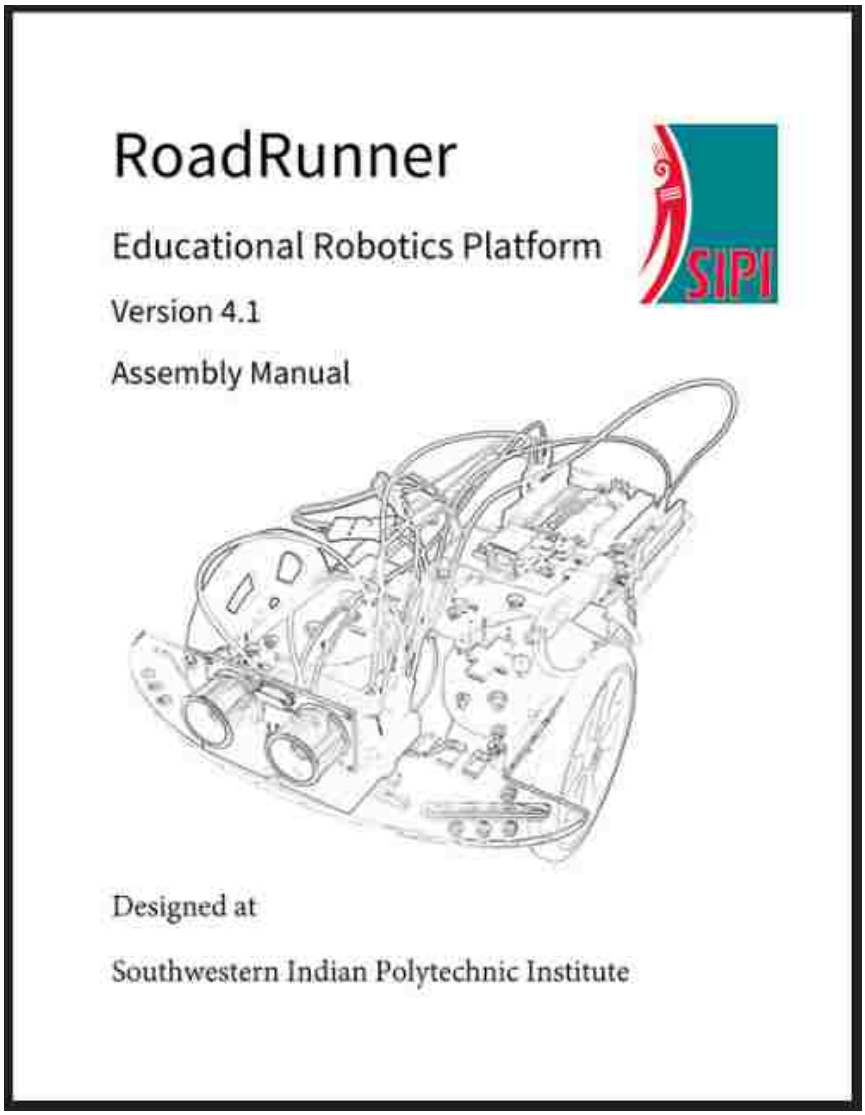


Figure 4.16: Assembly Manual

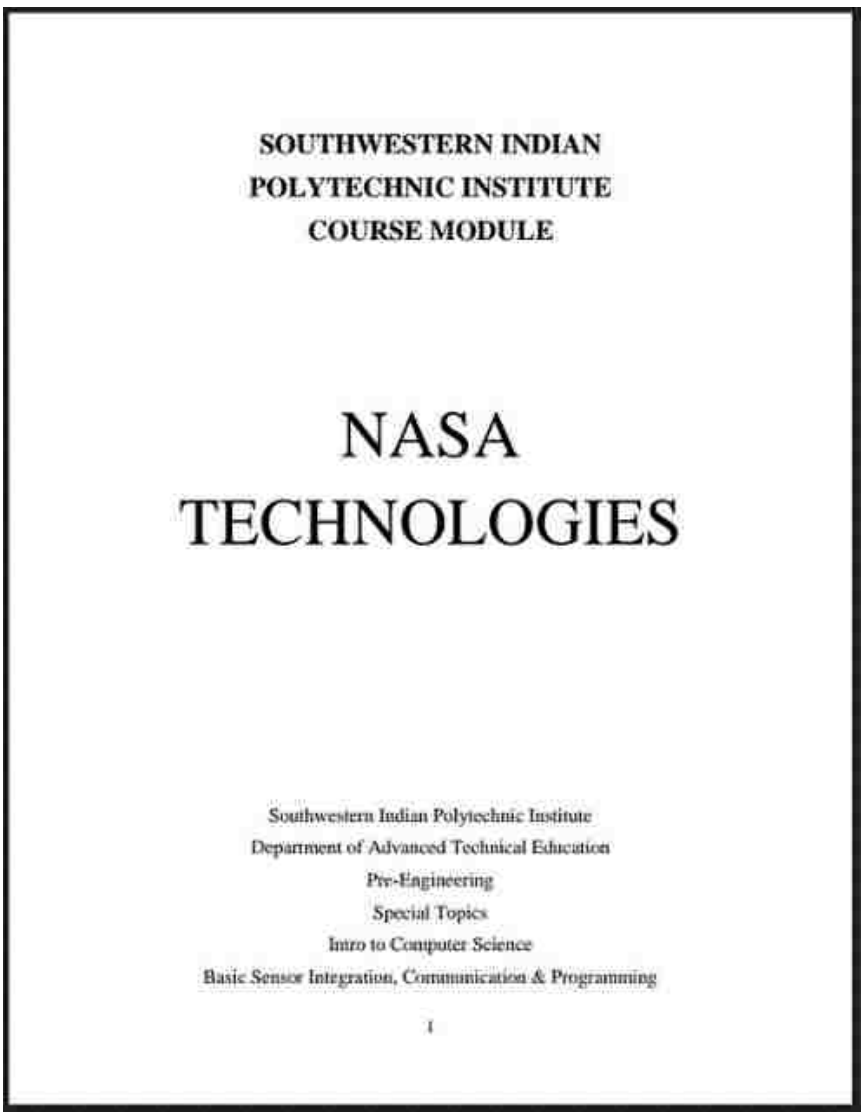


Figure 4.17: Curriculum

Chapter 5

Visual Localization Using Natural and Artificial Landmarks

5.1 Introduction

5.1.1 System Description and Intended Use

This paper describes a platform independent, visual localization algorithm which allows a rover to find its location on a map based on recoginition of visual landmarks. Although it is a platform independent software algorithm, this localization system is specifically designed to address the needs of the SIPI indoor Mars Facility and it therefore has very specific requirements and restrictions placed on it. Some of these restrictions are due to the effort to simulate the actual conditions on Mars. These include operation with significant communication latency, limited power resources, and lack of external positioning systems (such as GPS satellites). While these restrictions are required for the Martian simulation, they are also useful in terrestrial applications. Specifically, the lack of GPS is true for indoor applications

and the power restrictions are important for battery powered systems. Additionally, the desire to make the system applicable and extendable in educational applications demands that the system be as simple, small and inexpensive as possible so that students in schools with limited budgets can participate in the hardware and software development. The simple rover and lack of any sophisticated or expensive sensory equipment means that this rover functionality can reproduced on any platform at very little cost, thus opening the doors for robotics projects to schools which otherwise could not afford them.

The localization algorithm will work on any platform which includes a camera and some means of estimating platrform motion.

5.2 Background and Related Research

The problem of robot localization has been studied for many years. Many efforts have been made to achieve localization using cameras and object recognition[50] and the problem of accurately recognizing visual objects has been the topic of ongoing research for decades. Visual identification methods such as SIFT[51] and SURF[52] have been well established and successfully used in many applications. This project is modular in design which allows any visual object recognition system to be used. For the sake of simpilicy and reliability, a simple Hough transform[53] is used as a circle detector and a pdf based color matching algorithim is used in this system. However, for application to natural landmarks, one of the more sophosticated and computationally burdensome systems must be used.

5.2.1 Position calculation

Once one or more landmarks are identified and their relative positions are estimated, the next issue is how to use those estimates to predict the rover's pose. Basic surveying methods such as triangulation and resectioning[54] can generally be used to locate an observation point based on two or more landmarks.

Triangulation requires knowledge of the distance to each landmark and the angle between them. The angles between objects can be measured with reasonable precision, however the distance estimates are typically poor since they rely solely on the estimate of the objects size in the noisy image. The uncertainty in the distance estimates also increases with distance. The sensitivity of the triangulation calculation with respect to the distance measurements makes it unreliable in this application.

Resectioning depends only on the angles between observed landmarks and not on the distance to them. This seems to be an advantage since the angles between observed landmarks should be simple to calculate based on the camera's calibration parameters. However, it requires at least three landmarks to be visible, which often is not the case. Since fewer than three landmarks are usually visible, the resectioning must rely on propagated estimates of previously observed landmarks which considerably increases the uncertainty. The resection formula also has a very high sensitivity to errors in the angles and therefore requires very precise angular measurements. These facts also make the resectioning formula unreliable in this application.

5.2.2 Estimate computations

The localization measurements have many sources of uncertainty and errors. The landmarks may be incorrectly identified due to poor performance of the object detection routines. The landmark's relative locations may be poorly estimated due

to image noise, and the location of previously observed landmarks may be poorly propagated due to uncertainty in the estimation of the platform motion. Since each object in the camera's field of view may represent a landmark, and each landmark may have one or more (correct or incorrect) identifications, the localization algorithm must consider a large set of possibilities. Each landmark identification is assigned a percentage representing the certainty of that identification. Each object may have any number of identifications assigned to it, all of which must add up to 100%. If several landmarks are detected and each has several possible identifications, this leads to a potentially large matrix of all possible combinations. These combinations must be combined or contrasted with each other in order to come to the best possible estimate. [55]

5.2.3 Kalman filter

Each estimate of the rover's pose includes a covariance matrix indicating the uncertainty of that estimate. These estimates can be noisy due to camera noise and jitter and therefore the individual measurements must be filtered to produce an optimal estimate. The Kalman filter[56] was selected due to its wide application and readily available software library in OpenCV. [57]

5.3 Method

5.3.1 Overview

Although the rover operates in a large, complicated system and can carry additional sensors such as inertial measurement units, ultrasonic and laser range finders and equipment such as grippers and manipulators, those systems are beyond the scope of

this paper. Here we focus exclusively on the design and operation of the localization system.

The rover software runs on a Linux computer (Raspberry Pi B+) under Linux and the Robot Operating System (ROS). However, the localization system is not specific to this particular rover nor to the ROS environment. Therefore, this localization software was written to run without any dependencies on ROS so that it can be compiled, run and tested on desktop or other computer hardware to simplify testing and broaden its potential application. Although it is developed under Linux, it could be ported to Windows or other platforms with relatively little effort.

5.3.2 Software Architecture

The software is written in C++. Each software module is implemented in its own class and each class is written in its own source and header file. Each class also has an associated unit test file that exercises that classes functionality to verify its correct performance. The unit tests use Google gtest libraries where appropriate.

Each class can save and load its entire state to disk in the form of a JSON file. These JSON state files can be used to save the class state and load it again. In the development version of the system, the state of the entire system is saved on exit and can be reloaded on later runs to continue from that state. The JOSN files are also used to configure the classes. For example, the tuning parameters of the control loops, the pixel size and field of view of the camera and all other aspects of the system are configured in the JSON files. Since the JSON files are simple text files, they are directly human readable and editable. They can also be directly edited by any online JSON editor such as jsoneditoronline[58]. This allows a user to customize the behaviour (such as to change to a different camera) without any special software tools or recompiling.

The software classes are extensively documented using doxygen[59].

5.3.3 Communication

The communication to and from the localization system is through JSON formatted messages. JSON messages were selected because they are human readable, and are widely used in the Javascript based web browsers and therefore are well documented and supported by all programming languages. This allows students and other users to write simple web based controls and interfaces to send and received JSON messages to the Mars server.

In the final SIPI system, the communication takes place over websockets which pass JSON messages as ROS messages through rosbridge libraries. The users generate JSON commands in their web browsers and send them to the internet interface on the mars server. The server then passes those JSON messages to the rover to which they are addressed via a rosbridge websocket. The rover then processes those JSON message and takes action and/or sends JSON messages in response.

In this development version, the JSON commands are sent by the user from the keyboard and the responses are to the console or logged to disk. However, the commands and replies are the same as would be used in the final system. The transition from this development version to the final delivered system is achieved by way of a few simple adapter libraries to deal with the different routing of the communications.

5.3.4 Simulation

For testing purposes, a simple simulation environment was developed which keeps track of a simulated rover's pose in a given map and renders simulated noisy images

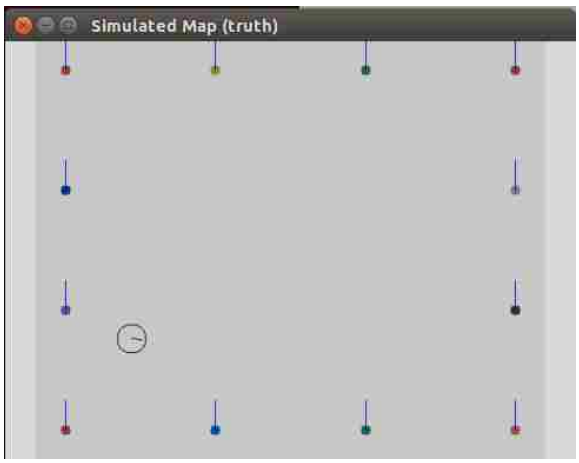


Figure 5.1: Simulator rover pose on map

of the landmarks from that rover's perspective. These images are fed into the localization algorithm in place of, but through the same interface as, the actual camera images. The simulation also responds to the navigation systems motions commands that would normally go to the motor driver. These motor command messages are used to produce the appropriate motion in the simulated rover's pose. The rendered simulation image and simulated odometry of the motion serves as the only inputs to the localization system. The colors used to render the artificial landmarks is taken from samples of camera images of the actual landmarks in the mini Mars yard and therefore accurately represent the detection properties of the actual landmarks. A sample of the siumulated rover position and the corresponding rendering are shown in Figures 5.1 and 5.2.

A more complete simulation environment is being developed using the ROS and Gazebo software packages. However, in order keep the complexity and dependencies of this package small, the simplified simulation environment is used which is adequate to test and evaluate the performance of the localization system in isolation.

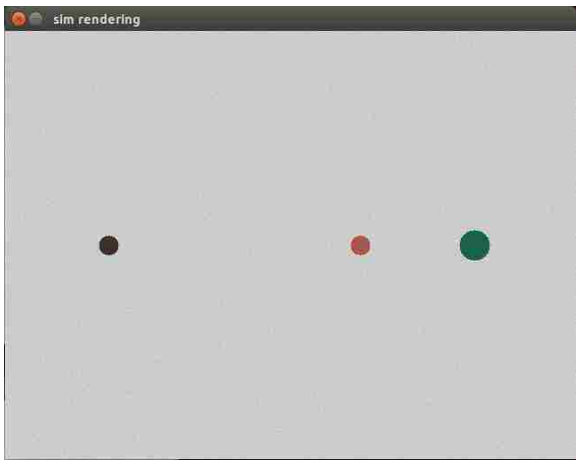


Figure 5.2: Simulator Rendered Image

5.3.5 Testing and Validation

In order to test and evaluate the performance of the localization and navigation system, the software can log all of the internal data at each time interval. The data is saved as JSON files which contain timestamps. At each time interval, the simulator records its simulated pose data that is used as the ground truth. At the same time, each other software module can record its state. The data logging can be individually turned on and off for each module so that selected data can be recorded on each run.

These data files can then be analyzed offline. Several routines have been written using the R language[60] to perform statistical analysis of the data. These R programs read all of the JSON files at each timestamp and then calculate such data as confusion matrices for the landmark identification, error in the pose estimation and path length of the navigation. Using R greatly simplifies the complex process of data analysis and reporting.

The localization algorithm is based exclusively on camera images and odometry estimates of the robots platform motion from the motors. It requires no external

sources of information except for a knowledge of the locations of enough landmarks to provide reference points. There are no GPS or range-finding sensors involved.

The localization is based on the rover maintaining a memory of all of the possible landmarks that it has observed. A landmark can be any object which the rover can visually identify and which has a known location. As the rover acquires images, it identifies potential landmarks and estimates their location relative to its line of sight. As it moves, it remembers previously identified landmarks and propagates their estimated relative position based on the estimate of the rover's motion. In this way, the rover maintains a database of landmarks which it has observed, even if they are no longer visible. As it moves, some old landmarks will move out of the filed of view and new ones will enter it. Landmarks that are no longer visible are propagated, but with an increasing uncertainty due to the uncertainty in the motion estimates. Therefore as the rover moves, older landmarks with large uncertainty are eventually dropped from the database while new landmarks are added. If a new observation coincides with an older one, then the new one replaces the older one in order to reduce its uncertainty.

As the rover develops the database of observed landmarks, it compares them to the known landmarks on its map. The comparison is done by translating and rotating the rover's pose and overlaying that on the map to achieve the best possible match between the observed landmarks and the known landmarks. Since there can be considerable uncertainty in the observed landmarks identities and locations, all of the possible combinations of observations are considered and the best is selected. This best match is then used to calculate the rover's observed pose on the map.

In order to reduce the noise in the observed poses, the observations are processed through a Kalman filter. The Kalman filter combines each individual observed pose with a combination of the previous observations to provide the best cumulative estimate. The output of the Kalman filter is then provided to the rest of the robotic

system as a position and corresponding covariance. Figure 5.3 shows the logical flowchart of the process.

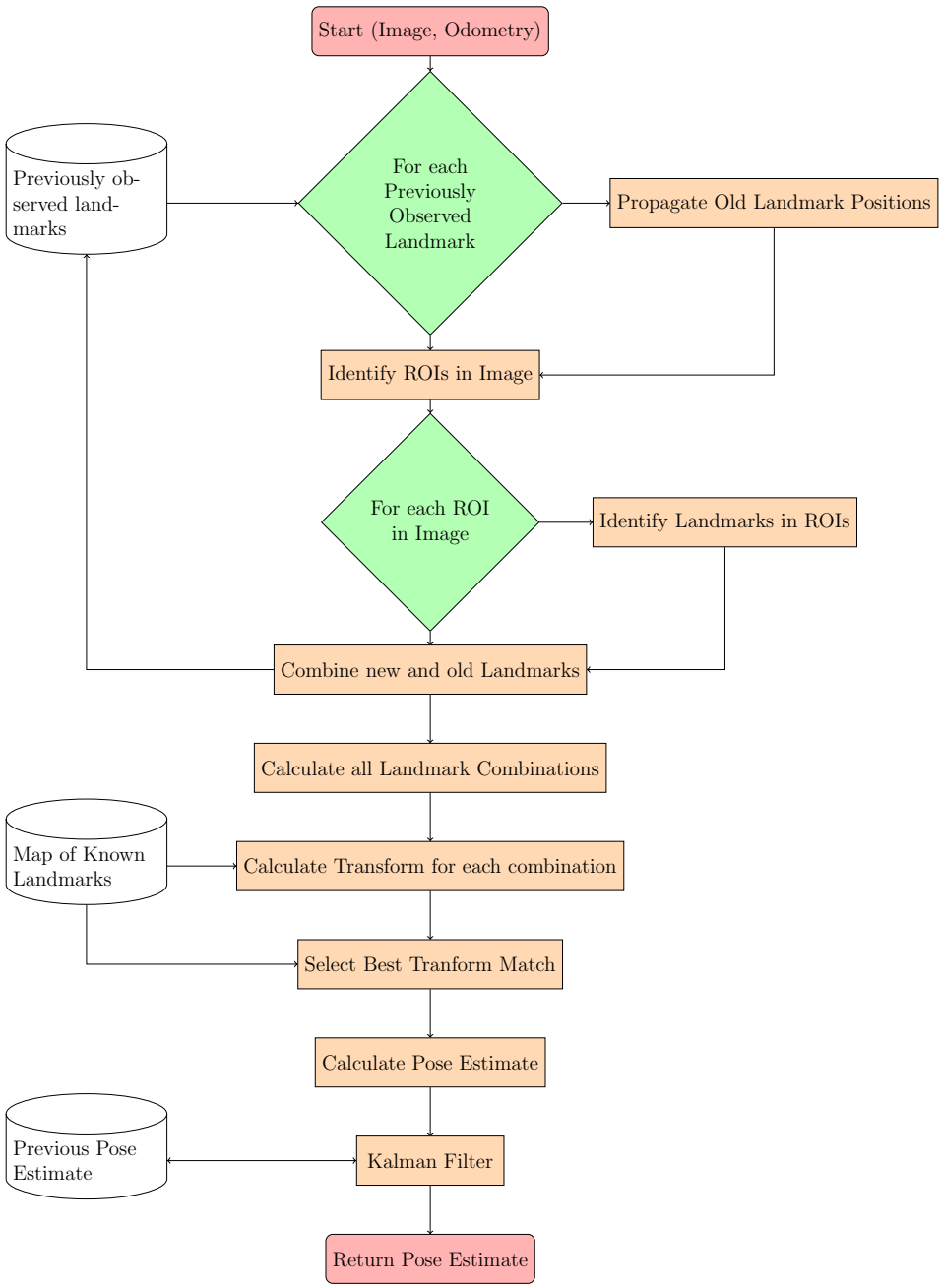


Figure 5.3: Localization Algorithm Flowchart

5.3.6 Initialization

When the rover is first powered on, it has no estimate of its pose, so it begins at a default location, but with extremely large variance.

5.3.7 Image Acquisition and Processing

The image is acquired from the camera, a simulation rendering or a stored video file.

5.3.8 ROI search

The image is first searched for Regions of Interest (ROIs). For the spherical landmarks of this experiment, the ROIs are found by running a low resolution Hough transform which quickly finds circles in the image. Extension to more complex landmark objects can be done through applying different image detection routines such as SIFT or SURF or using artificial landmarks such as April Tags [61].

The result of the ROI search is a list of rectangular areas in the image that may contain items of interest. If a previous pose estimate exists, it is also used to estimate the positions of the known landmarks into the scene and then use those locations as ROIs. Overlapping ROIs are combined. The ROIs for a sample image are shown in Figure 5.4

5.3.9 Landmark Identification

Each ROI is then searched in detail for any of the landmarks in the known map. For the spherical landmarks, this consists of a fine resolution Hough transform to determine the presence of a circle and its diameter. The color of the area enclosed

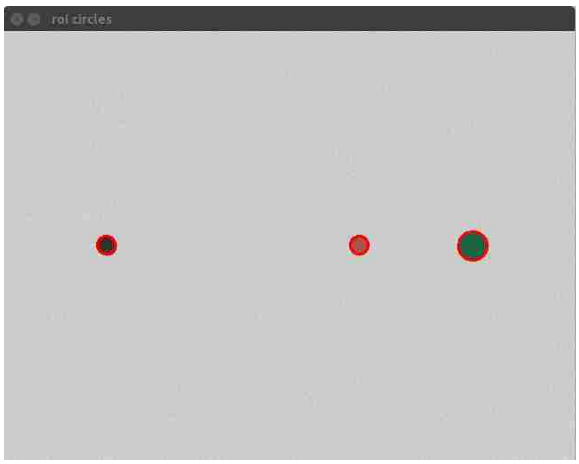


Figure 5.4: ROI detetor output

by the circle is then matched against the colors of the landmarks in the known map using pdf histogram matching and possible identities are calculated.

The direction to the landmark is calculated from the circle center's location in the field of view of the camera. The distance to the landmark is calculated based on the known size of the landmark and the apparent size of the landmark in the image. The possible identities, estimated direction and the estimated distance to the landmark are stored in a table of possible landmark observations.

The position of these landmark observations are estimated in polar coordinates in the rovers local frame of reference. Each potential landmark is stored in a table of landmarks including its estimated identity and relative location. If the object in a ROI has several possible identities, then each is stored along with a certainty corresponding to that identification. This is necessary since landmarks are often misidentified and it is necessary to carry all possibilities forward in order to increase the accuracy of the final result. The landmark identification results for a sample image are shown in Figure 5.5.

This landmark identification is a great simplification due to the well defined and

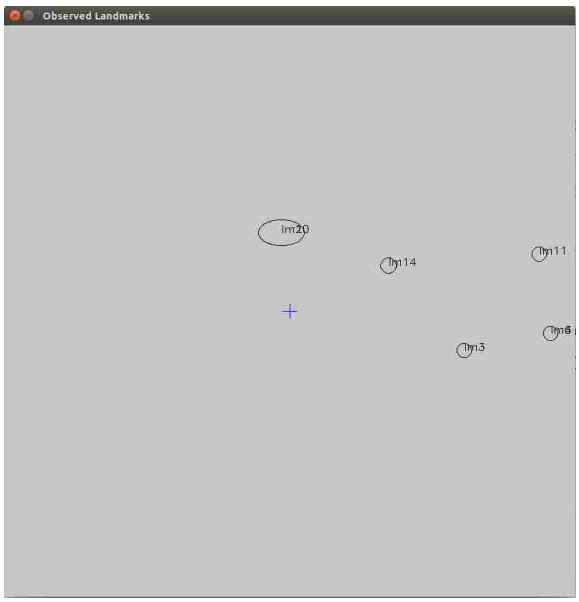


Figure 5.5: Landmark Detection Result

easily identified artificial landmarks that are being used. More complicated algorithms will be required for more natural shapes and colors. This paper is focused on the localization and navigation algorithms and not the detection of the landmarks themselves. Since the software is written in a modular fashion, the Hough transform/color matching algorithm can be replaced by any other process with little impact on the rest of the code.

5.3.10 Propagation of Old Observations

At each update, the relative locations of all of the identified landmarks in the database are propagated based on the control inputs to the motors and the odometry sensors.

The relative location of each landmark is shifted based on the estimated linear and angular motion of the platform and the variance of the estimates is adjusted

to account for the uncertainty of the odometry measurements. This results in an always increasing variance for the estimates. After the propagation, the variance is compared against a threshold and the landmark observation is removed from the database if it is no longer accurate enough to be valuable.

Since the landmark observations are all with respect to the rover's position, the propagation is straightforward. Linear motion is defined as in the x direction.

First, the estimated motion is computed based on the odometry sensors:

$$\Delta x = (\dot{\theta}_l + \dot{\theta}_r)\frac{r}{2}dt \tag{5.1}$$

$$\Delta \theta = (\dot{\theta}_l - \dot{\theta}_r)\frac{rd}{2}dt \tag{5.2}$$

Then each observed landmark's location A_k is adjusted to account for this motion based on its previous estimated location A_{k-1}. The distance and angle to each landmark is computed as follows.

$$r_{k-1} = \sqrt{x_{k-1}^2 + y_{k-1}^2} \tag{5.3}$$

$$\theta_{k-1} = atan2(y_{k-1}, x_{k-1}) \tag{5.4}$$

$$x_k = r_{k-1} * \cos(\theta_{k-1} + \Delta\theta) + \Delta x \tag{5.5}$$

$$y_k = r_{k-1} * \sin(\theta_{k-1} + \Delta\theta) \tag{5.6}$$

5.3.11 Combination of new landmarks with existing database

Once all of the potential landmarks from the current image are calculated, they are compared against previously identified landmarks in the database. If a landmark from the current image matches an existing one in the database based on their identification and position (in rover's frame) then they are combined. If a new landmark does not match any in the database then it is added. a sample landmark database for a position is shown in Figure 5.5.

5.3.12 Visual Localization

The database of observed landmarks is then compared to the known map to determine the rover's pose.

5.3.13 Combining New Landmarks With Previous Observations

The database may have any number of possible landmark identities and locations. Many of these identifications may be incorrect and some of the positions may also be poorly estimated. In order to achieve the highest accuracy, none of the possibilities can be neglected, so a recursive list of all of the possible landmark combinations is created. In order to limit the computational burden, the combinations are limited to 5 landmarks each, but all possible 5 landmark combinations are considered.

5.3.14 Transformation

In order to determine if a combination represents a valid pose, a rigid body transformation is found which best matches all of the landmarks in the combination. The Singular Value Decomposition (SVD) is used to calculate an optimal rigid body translation where the origin of the rover's frame of reference is rotated and translated to find the best possible match to the locations of the landmarks on the known map. The Singular Value Decomposition is a factorization of the form:

$$H = UWV^T \tag{5.7}$$

Where H is the covariance matrix of the of coordinates of the observations and the corresponding coordinates of the landmarks in the map. From this SVD you can

calculate the rotation and translation of your transform by the following: Reference Olga Sorkine

Given two sets of corresponding landmark coordinates

$A =$ Set of coordinates of observed landmark locations

$B =$ Set of coordinates of corresponding known landmark locations

We want a rigid transform of the form:

$$\vec{b} = R\vec{a} + \vec{t} \tag{5.8}$$

Which converts points in set A to points in set B by rotating the coordinates of $\vec{a}$ about the origin using rotation matrix R and then translating by vector $\vec{t}$ where R and $\vec{t}$ are given by:

$$R = V \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & det(VU^T) \end{pmatrix} U^T \tag{5.9}$$

$$\vec{t} = \bar{a} - R\bar{b} \tag{5.10}$$

The SVD is performed by using the OpenCV SVD class. First, the centroids of the A and B sets are calculated by

$$\bar{a} = \frac{\Sigma_{i=1}^{n} \vec{a_i}}{n} \qquad\qquad \bar{b} = \frac{\Sigma_{i=1}^{n} \vec{b_i}}{n} \tag{5.11}$$

Then the covariance matrix H is calculated by shifting A and B centroids to the origin and taking the covariance of A' and B'.

$$A' = A - \bar{a} \qquad\qquad B' = B - \bar{b} \tag{5.12}$$

$$H = A'B'^{T} \tag{5.13}$$

These matrices are then used by the OpenCV SVD class to compute W, U and V^T in Equation 5.7. Equations 5.9 and 5.10 are then used to calculate R and t.

5.3.15 Identifying Best Match

The SVD calculations result in a transformation which is applied to the rover to give it a possible pose. From this estimated pose, the landmarks are projected to their observed locations and then compared to the known map. If the locations of the landmarks statistically agree, then it is considered a potential match. If any of the landmarks do not match the map, then the combination is discarded. The certainty of the match for this combination is determined by combining the certainties of the identifications of the landmarks in the combination. For all of the combinations that were considered, only a limited number will survive the transforming stage. Of the remaining combinations, the combination with the highest certainty is selected. The certainty is weighted to favor combinations with more landmarks over those with few.

The rover pose corresponding to the best match is then calculated. The variance assigned to this pose estimate is calculated based on the combination of the variances of all of the landmarks that were used in its generation. This new pose is used as the best observation for this cycle.

5.3.16 Kalman Filter

As with any sensor based observation, the observed pose is subject to many forms of noise and errors. The pose estimate from the visual localization algorithm includes a covariance matrix to indicate the certainty of that pose. The Kalman filter is used

to combine these potentially noisy observations into a filtered pose estimate.

The Kalman filter is generally defined by the state transition equation:

$$x_k = F_k x_{k-1} + B_k u_k + w_k \tag{5.14}$$

Where:

x_k is the current state estimate

F_k is the transition matrix from the previous to the current states

x_{k-1} is the previous state

B_k is the transition matrix from the control inputs to the change in state

u_k is the control input

w_k is the process noise

And the observation equation:

$$z_k = H_k * x_k + v_k \tag{5.15}$$

Where:

z_k is the current measurement (observation)

H_k is the transition matrix from the state to a measurement

x_k is the current state

v_k is the measurement noise

The state of the rover is defined as:

$$x = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} = \begin{pmatrix} x\ coordinate \\ y\ coordinate \\ azimuth\ angle\ from\ north \end{pmatrix} \tag{5.16}$$

The transition matrix from the previous to current state is constant:

$$F = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5.17}$$

The control inputs are defined as:

$$u = \begin{pmatrix} \dot{\theta}_l \\ \dot{\theta}_r \end{pmatrix} = \begin{pmatrix} angular\ velocity\ of\ right\ wheel \\ angular\ velocity\ of\ left\ wheel \end{pmatrix} \tag{5.18}$$

The translation from control inputs to the change in the current state is dependent on the current state and is defined as:

$$B = \begin{pmatrix} (r/2)\sin\theta dt & (r/2)\sin\theta dt \\ (r/2)\cos\theta dt & (r/2)\cos\theta dt \\ (r/2d)dt & -(r/2d)dt \end{pmatrix} \tag{5.19}$$

Where r is the wheel radius and d is the distance from the center of the rover to the wheels.

The measurement is an estimate of the rover's pose and its estimated covariance and is given by:

$$\vec{z} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} = \begin{pmatrix} x\ coordinate \\ y\ coordinate \\ azimuth\ angle\ from\ north \end{pmatrix} \tag{5.20}$$

And its estimated covariance matrix:

$$z_{cov} = \begin{pmatrix} \sigma_x^2 & cov_{x,y} & cov_{x,\theta} \\ cov_{y,x} & \sigma_y^2 & cov_{y,\theta} \\ cov_{\theta x} & cov_{\theta y} & \sigma_\theta^2 \end{pmatrix} \tag{5.21}$$

The measurement prediction from current state is:

$$H = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5.22}$$

5.3.17 Validation

The variance of the Kalman filter output is used to measure the accuracy of the pose estimate. If this variance is too large for the rover to maneuver safely, then it must improve the localization accuracy before it can proceed. The covariance of the pose estimate is provided to the robot system. If the covariance is too large to manuver, then the robot will have to take appropriate action such as turning in place or scanning with its camera until landmarks are visible and the localization can be improved. The validation code of the Localization routine reports this variance to the controller which then decides if localization is necessary.

5.4 Conclusion

The visual localization system has proven to provide a robust visual localization system for the Mars Yard rovers. Although the current system requires Apriltag or artificial colored landmarks, the same algorithm could be used with natural landmarks as well. However the image recognition processing load is very high for natural landmarks and would therefore require upgraded processors.

References

[1] A. Prorok, M. A. Hsieh, and V. Kumar, "Formalizing the impact of diversity on performance in a heterogeneous swarm of robots," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 5364–5371.

[2] P. J. Cruz and R. Fierro, "Building coalitions of heterogeneous agents using weighted bipartite graphs," in *2015 54th IEEE Conference on Decision and Control (CDC)*, Dec 2015, pp. 2822–2828.

[3] C. Benjamin, G. Kahn, S. Patil, S. Liu, K. Goldberg, P. Abbeel, N. Michael, and V. Kumar, "Information-theoretic planning with trajectory optimization for dense 3d mapping." in *Robotics: Science and Systems. 2015.*, 2015.

[4] A. Howard, L. E. Parker, and G. S. Sukhatme, *The SDR Experience: Experiments with a Large-Scale Heterogeneous Mobile Robot Team.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 121–130.

[5] S. Wang, B. Krishnamachari, and N. Ayanian, "The optimism principle: A unified framework for optimal robotic network deployment in an unknown obstructed environment," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 2578–2584.

[6] Z. Feng, C. Sun, and G. Hu, "Robust connectivity preserving rendezvous of multi-robot systems under unknown dynamics and disturbances," *IEEE Transactions on Control of Network Systems*, vol. PP, no. 99, pp. 1–1, 2016.

[7] G. A. Hollinger and S. Singh, "Multirobot coordination with periodic connectivity: Theory and experiments," *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 967–973, Aug 2012.

[8] M. M. Zavlanos, "Synchronous rendezvous of very-low-range wireless agents," in *49th IEEE Conference on Decision and Control (CDC)*, Dec 2010, pp. 4740–4745.

[9] P. J. Cruz, B. M. Sadler, and R. Fierro, "Sensor localization using hybrid rf/optical wireless communications for an aerial data mule," in *2016 American Control Conference (ACC)*, July 2016, pp. 7085–7091.

[10] J. Stephan, J. Fink, V. Kumar, and A. Ribeiro, "Concurrent control of mobility and communication in multirobot systems," *IEEE Transactions on Robotics*, vol. PP, no. 99, pp. 1–7, 2017.

[11] P. J. Cruz and R. Fierro, "Towards optical wireless communications between micro unmanned aerial and ground systems," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2015, pp. 669–676.

[12] M. J. Bays and T. A. Wettergren, "A solution to the service agent transport problem," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 6443–6450.

[13] S. Bernardini, M. Fox, and D. Long, "Combining temporal planning with probabilistic reasoning for autonomous surveillance missions," *Autonomous Robots*, pp. 1–23, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10514-015-9534-0

[14] G. Best, J. Faigl, and R. Fitch, "Multi-robot path planning for budgeted active perception with self-organising maps," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 3164–3171.

[15] S. S. Chouhan and R. Niyogi, "Multi-agent planning with quantitative capability," in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sept 2016, pp. 602–606.

[16] C.-H. Wang and K.-N. Hung, "Adaptive som-based fuzzy neural network controller design for multi-agent system dispatching and path planning," in *2012 IEEE International Conference on Fuzzy Systems*, June 2012, pp. 1–7.

[17] G. M. Mathews and H. F. Durrant-whyte, "Scalable decentralised control for multi-platform reconnaissance and information gathering tasks," in *2006 9th International Conference on Information Fusion*, July 2006, pp. 1–8.

[18] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 1, 2000, pp. 476–481 vol.1.

[19] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart, "Distributed multirobot exploration and mapping," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1325–1339, July 2006.

[20] A. Maoudj, B. Bouzouia, A. Hentout, and R. Toumi, "Multi-agent approach for task allocation and scheduling in cooperative heterogeneous multi-robot team: Simulation results," in *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, July 2015, pp. 179–184.

[21] M. Maniadakis, E. E. Aksoy, T. Asfour, and P. Trahanias, "Collaboration of heterogeneous agents in time constrained tasks," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, Nov 2016, pp. 448–453.

[22] J. Butzke, K. Gochev, B. Holden, E. J. Jung, and M. Likhachev, "Planning for a ground-air robotic system with collaborative localization," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 284–291.

[23] A. V. Savkin and H. Li, "Collision free navigation of a non-holonomic ground robot for search and building maps of unknown areas with obstacles," in *2016 35th Chinese Control Conference (CCC)*, July 2016, pp. 5409–5414.

[24] M. Balclar, E. Uslu, F. akmak, N. Altunta, S. Marangoz, M. F. Amasyal, and S. Yavuz, "An architecture for multi-robot hector mapping," in *2016 International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*, Aug 2016, pp. 1–5.

[25] N. Mahdoui, E. Natalizio, and V. Fremont, "Multi-uavs network communication study for distributed visual simultaneous localization and mapping," in *2016 International Conference on Computing, Networking and Communications (ICNC)*, Feb 2016, pp. 1–5.

[26] R. S. Ponmagal, K. Sujatha, and T. Godhavari, "Cloud computing for autonomous navigation of unmanned ground vehicle," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, March 2016, pp. 3790–3795.

[27] S. A. Miratabzadeh, N. Gallardo, N. Gamez, K. Haradi, A. R. Puthussery, P. Rad, and M. Jamshidi, "Cloud robotics: A software architecture: For heterogeneous large-scale autonomous robots," in *2016 World Automation Congress (WAC)*, July 2016, pp. 1–6.

[28] K. Kamei, S. Nishio, N. Hagita, and M. Sato, "Cloud networked robotics," *IEEE Network*, vol. 26, no. 3, pp. 28–34, May 2012.

[29] R. Arumugam, V. R. Enti, L. Bingbing, W. Xiaojun, K. Baskaran, F. F. Kong, A. S. Kumar, K. D. Meng, and G. W. Kit, "Davinci: A cloud computing framework for service robots," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 3084–3089.

[30] G. Hu, W. P. Tay, and Y. Wen, "Cloud robotics: architecture, challenges and applications," *IEEE Network*, vol. 26, no. 3, pp. 21–28, May 2012.

[31] A. Rahman, J. Jin, Y. W. Wong, and K. S. Lam, "Development of a cloud-enhanced investigative mobile robot," in *2016 International Conference on Advanced Mechatronic Systems (ICAMechS)*, Nov 2016, pp. 104–109.

[32] D. Hunziker, M. Gajamohan, M. Waibel, and R. D'Andrea, "Rapyuta: The roboearth cloud engine," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 438–444.

[33] E. Andres, P. Nehlig, and J. Françon, "Supercover of straight lines, planes and triangles," in *Proceedings of the 7th International Workshop on Discrete Geometry for Computer Imagery*, ser. DGCI '97. London, UK, UK: Springer-Verlag, 1997, pp. 243–254. [Online]. Available: http://dl.acm.org/citation.cfm?id=648317.754070

[34] S. F. Conservancy, "git." [Online]. Available: http://git-scm.com

[35] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[36] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, Sept 2004, pp. 2149–2154 vol.3.

[37] S. Fuhrmann, F. Langguth, and M. Goesele, "Mve - a multi-view reconstruction environment," in *Eurographics Workshop on Graphics and Cultural Heritage, Darmstadt, Germany*, 2014.

[38] (2017) Open drone map. [Online]. Available: https://www.opendronemap.org

[39] (2014) 2014 native youth report. [Online]. Available: https://obamawhitehouse.archives.gov/sites/default/files/docs/20141129nativeyouthreport_final.pdf

[40] (2009) Next generation science standards. [Online]. Available: http://www.nextgenscience.org

[41] J. Gordon, "SIPI IC MARS project program evaluation report," SIPI, Tech. Rep., 09 2016.

[42] A. McMahon, N. Vadiee, and J. West, "SIPI NASA TCU-ELO final report: Information technology experiences using simulated tele-science exploration of mars," SIPI, Tech. Rep., 08 2017.

[43] SIPI. (2016) NASA TCU ELO program & SIPI IC-MARS Project education and outreach. [Online]. Available: https://sites.google.com/site/sipimodules/home

[44] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[45] NASA. (2016) Nasa swarmathon. [Online]. Available: https://www.nasaswarmathon.com

[46] EPSCOR. (2016) New mexico experimental program to stimulate competitive research. [Online]. Available: https://www.nmepscor.org

[47] O. Kreylos. (2016) Augmented reality sandbox. [Online]. Available: https://idav.ucdavis.edu/ okreylos/ResDev/SARndbox/

[48] COSGC. (2016) Rockon the next how-to workshop. [Online]. Available: http://spacegrant.colorado.edu/national-programs/rockon-2017-home

[49] WSGC. (2016) First nations launch. [Online]. Available: https://spacegrant.carthage.edu/first-nations-launch

[50] D. Schleicher, L. Bergasa, R. Barea, E. Lopez, and M. Ocana, "Real-time simultaneous localization and mapping using a wide-angle stereo camera and adaptive patches," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct 2006, pp. 2090–2095.

[51] Z. Wang, H. Xiao, W. He, F. Wen, and K. Yuan, "Real-time sift-based object recognition system," in *Mechatronics and Automation (ICMA), 2013 IEEE International Conference on*, Aug 2013, pp. 1361–1366.

[52] M. Du, J. Wang, J. Li, H. Cao, G. Cui, J. Fang, J. Lv, and X. Chen, "Robot robust object recognition based on fast surf feature matching," in *Chinese Automation Congress (CAC), 2013*, Nov 2013, pp. 581–586.

[53] D. Kerbyson and T. Atherton, "Circle detection using hough transform filters," in *Image Processing and its Applications, 1995., Fifth International Conference on*, Jul 1995, pp. 370–374.

[54] P. Milburn and R. Allaby, "A field-ready solution to the resection problem given two coordinated points." *Canadian Agricultural Engineering*, vol. 29, pp. 93–96, 1987.

[55] R. Roopesh, "Combining predictive distributions," *University of Heidelberg*, 2011.

[56] G. Welch and G. Bishop, "An introduction to the kalman filter," *University of North Carolina at Chapel Hill*, 2006.

[57] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[58] "Online json editor," jsoneditoronline.org, 2015, [Online; accessed 14-November-2015].

[59] "Doxygen code documentaion," www.doxygen.org, 2015, [Online; accessed 14-November-2015].

[60] "The R project for statistical computing," www.r-project.org, 2015, [Online; accessed 14-November-2015].

[61] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2016.