COLLABORATION BETWEEN UNMANNED AERIAL AND GROUND VEHICLES FOR SEARCH AND RESCUE MISSIONS

A PROJECT REPORT

Submitted by

RAMANAN SEKAR (312214105077)

SAI SHANKAR N (312214105082)

SHIVA SHANKAR B (312214105091)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRICAL AND ELECTRONICS ENGINEERING

SSN COLLEGE OF ENGINEERING, KALAVAKKAM

ANNA UNIVERSITY:: CHENNAI 600 025

APRIL 2018

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**COLLABORATION BETWEEN UNMANNED AERIAL AND GROUND VEHICLES FOR SEARCH AND RESCUE MISSIONS"** is the *bonafide* work of **RAMANAN SEKAR (312214105077), SAI SHANKAR N (312214105082),** and **SHIVA SHANKAR B (312214105091)** who carried out the research under my supervision.

SIGNATURE

DR.V.KAMARAJ

HEAD OF THE DEPARTMENT

Professor

Electrical and Electronics Engineering

SSN College of Engineering

Kalavakkam-603110

SIGNATURE

DR. RANGANATH MUTHU

SUPERVISOR

Professor

Electrical and Electronics Engineering

SSN College of Engineering

Kalavakkam-603110

VIVA –VOCE EXAMINATION

The Viva-voce examination for the project work, **"COLLABORATION BETWEEN UNMANNED AERIAL AND GROUND VEHICLES FOR SEARCH AND RESCUE MISSIONS"** submitted by **RAMANAN SEKAR (312214105077), SAI SHANKAR N (312214105082),** and **SHIVA SHANKAR B (312214105091)** and held on **----------**

Internal Examiner **External Examiner**

ACKNOWLEDGEMENTS

We are grateful to our Project Supervisor, **Dr. Ranganath Muthu, Professor**, Department of Electrical and Electronics Engineering for his expert guidance and constant support throughout the project.

We express our sincere gratitude to **Dr.S.Salivahanan, Principal**, SSN College of Engineering, **Dr. V. Kamaraj, Professor and Head**, Department of Electrical and Electronics Engineering, and the Management for providing us with the encouragement to complete this project.

We are indebted to all the faculty members and assistants of the Electrical and Electronics Engineering Department for their invaluable assistance.

RAMANAN SEKAR

SAI SHANKAR N

SHIVA SHANKAR B

ABSTRACT

Autonomous robot missions in unknown environments are challenging. In many cases, the systems involved are unable to use a priori information about the scene. This is especially true in disaster response scenarios, where existing maps are now out of date. GPS-denied areas are another concern, especially when the involved systems are tasked with navigating a global path planned by a base station. Scene understanding via robots' perception data can greatly assist in overcoming these challenges. This project work is an implementation of a collaborative robot system that help overcome these challenges, where there is a focus on the application of autonomously searching and rescuing people in disaster zones such as earthquakes with unmanned aerial vehicles (UAV) and unmanned ground vehicles (UGV) in unknown and unstructured environments. The approach proposed here is motivated by the need to deliver a fast, unmanned response in a previously unexplored environment using a collaborative robot team. We address the problem of exploration of an unknown environment by a flying robot using its onboard sensors from an overhead perspective. This information will be processed using traditional computer vision techniques to yield a binary occupancy grid, which will be used for motion and mission planning for the ground vehicle. The ground vehicle will perform its duties in such a way that prioritizes minimum time and the service of maximum number of people.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

SYMBOL	DESCRIPTION
R	Rotation Operation
t	Translation Operation
s	Scale Factor
$\overline{x}$	Point in homogeneous coordinates
x'	Point in transformed coordinates

CHAPTER 1
INTRODUCTION

1.1 MOTIVATION

In disaster environments or search and rescue scenarios, time is a critical factor in the success of the first responders, as shown by Murphy R.R (2014). These are also scenarios where those same rescue personnel must often put themselves in dangerous situations in order to provide aid. Unmanned systems have the possibility to provide new capabilities for these operators, as well as increase their safety and decrease the response time in delivering that aid. However, one challenge for these scenarios is that the environment may have been altered by the disaster (e.g., an earthquake or a mudslide), potentially invalidating any prior information about the environment, such as maps from aerial surveys or satellite imagery. Consequently, robotic systems that can benefit first responders must be capable of gathering and using data on demand, without reliance on a priori maps.

Since 2001, rescue robots have been deployed for disaster response. For example, after the earthquake and tsunami in Fukushima, Japan in 2011, ground robots were utilized to explore the situation in the contaminated reactor building as shown by Nagatani et. al (2013). In all previous disaster response missions, the robots were remote-controlled by trained professionals. Three operators per robot were required on average and the executed missions took very long. Since time is the most critical factor in rescue missions, the proposed idea is to deploy teams of heterogeneous robots, namely ground and aerial robots, to speed up disaster response. Their sense-act capabilities are complementary: ground robots can carry high payloads and manipulators. However, their field of view is limited and they can

be blocked by obstacles on the ground. Aerial robots, in contrast, can overcome obstacles with ease and can provide a bird's-eye view, which is ideal for mapping and monitoring tasks. To reduce the number of required operators and speed up their mission, the robots must expose a good level of autonomy. Instead of sending low-level commands, they must be able to autonomously execute high-level tasks such as "grasp that object" or "fly to location X". This allows the operator to focus on the mission instead of low-level robot details. For reliable local navigation, robots must rely only on their onboard sensors, since the communication infrastructure is likely to be affected during disaster situations. Our passion for robotics, control theory and computer vision, and our desire to work on a small-scale search and rescue setup using collaborative robotics is the main motivation for pursuing this project.

1.2 DETAILED DESCRIPTION

Autonomous unmanned systems have the potential to provide safer and more efficient solutions to problems that currently rely on manned missions. Relevant applications include disaster response, search and rescue, public transportation, infrastructure health monitoring and precision agriculture. The application focused on for this project work is the autonomous search and rescue missions using collaborative robots. For these purposes, one needs advanced perception and planning capabilities with respect to the performances of the robots. Specifically, we focus on unmanned aerial and ground vehicles for the search and rescue scenario.

We are interested in exploring the possibility of leveraging an autonomous quadrotor in such environments through field experiments that focus on cooperative mapping and planning using both ground and aerial robots. Aerial robots offer several advantages over ground robots, including the ability to manoeuvre through complex three-dimensional (3D) environments and gather data from vantages

inaccessible to ground robots. Further, quadrotors are able to hover in place, making them well-suited for observation and human-guided or autonomous inspection. However, aerial robots also have several limitations that reduce their applicability in disaster scenarios. Such limitations include the need for wireless communication and a limited on-board power supply that restricts the platform's payload capacity and flying time.

Understanding the capabilities that each system requires and the goals that are to be accomplished at the current state, our project will be structured to provide a very simple reconstruction of a disaster setup, which in our case, will be the earthquake, and to use a collaborative robotics scheme using UAVs and UGVs. A typical setup is shown in Fig1.1. The objectives that are to be completed, thus satisfying the goals that were stated earlier, are given as objectives in Section 1.3.

Fig.1.1 An air-ground vehicle system, from Butzke et. al (2016).

1.3 OBJECTIVES:

- To use a UAV to sequentially map the given area that needs to be surveyed by flying over the area through the specific waypoints.

- To use the video footage obtained from the UAV and sample the footage strategically and obtain images to be stitched
- Use a suitable computer vision technique to stitch the images
- Use the stitched images to identify and localize obstacles and keypoints where people are stuck, and identify the number of people
- Transform this information into a binary occupancy grid for motion planning of the UGV
- Use a suitable motion planning strategy for the UGV to execute
- Plan a mission in such a way that the total time for the execution of the mission is minimized, taking into account distances to be travelled and number of people to save.

1.4 ORGANIZATION OF THE REPORT

- **CHAPTER 2** gives a brief overview of the major milestones in the systems development and performance improvements in these collaborative systems, be it in air-air, air-ground, or ground-ground. This chapter also provides an overview of the current state-of-the-art in the use of unmanned aerial and ground vehicles for search and rescue operations.
- **CHAPTER 3** is dedicated to the explanation of the entire computer vision system that is the backbone of the entire process. This chapter gives a very brief introduction to what computer vision is, and delves into the details of transformations, filters, RANSAC, scale-invariant feature transforms, their application for object detection and panoramic stitching, and finally, the Viola Jones face (and object) detection algorithm.
- **CHAPTER 4** is dedicated to the study of motion planning algorithms. This chapter covers the combinatorial and sampling-based search algorithms and

focuses on why the sampling techniques are much better than the combinatorial techniques. This chapter finally focuses on our own mission planning algorithm that we developed, which would prioritize distances and the density of populations to minimize the overall time for the mission.

- **CHAPTER 5** gives detailed descriptions of the hardware that is used in the project, all the way from the unmanned ground vehicle build using the Arduino controllers, to the use of the Raspberry Pi 3 for wireless streaming of camera footage.
- **CHAPTER 6** gives the hardware implementation of the project on a large area, by setting up a mock disaster zone setup. The results are discussed in this chapter.
- **CHAPTER 7** is a reflection on the work that was done, and discusses the successes and obstacles that were faced during the completion of the project, and discusses what could be done in the future to improve on this work.

CHAPTER 2

LITERATURE REVIEW

2.1 A BRIEF OVERVIEW OF PREVIOUS WORK

The main research paper that our project work is based off is from the Robotics and Perception group at ETH Zurich, headed by Professor Davide Scaramuzza. The paper that his group published, namely the work by Mueggler et. al (2014), inspired us the most. They demonstrate the fully autonomous collaboration of an aerial and a ground robot in a mock-up disaster scenario. Within this collaboration, they make use of the individual capabilities and strengths of both robots. The aerial robot first maps an area of interest, then it computes the fastest mission for the ground robot to reach a spotted victim and deliver a first-aid kit. Such a mission includes driving and removing obstacles in the way while being constantly monitored and commanded by the aerial robot. Their mission planning algorithm distinguishes between movable and fixed obstacles and considers both the time for driving and removing obstacles. The entire mission is executed without any human interaction once the aerial robot is launched and requires a minimal amount of communication between the robots. They describe both the hardware and software of their system and detail their mission-planning algorithm. They present exhaustive results of both simulation and real experiments. Their system was successfully demonstrated more than 20 times at a trade fair. Their setup is shown in Fig.2.1.

The collaboration between autonomous unmanned systems has been studied for a large number of applications. These unmanned systems include autonomous underwater vehicles (AUV), unmanned surface vehicles (USV), UAVs, and UGVs. Some examples of the applications of these unmanned systems are search and rescue operations, post-disaster surveying, target localization and tracking, and precision

agriculture monitoring. Previous works have focused on the collaboration between multiple UAV (Yu et al, 2013), multiple UGV (Bruggemann et al 2012), the collaboration between UAV and UGV (Duan, 2014), and much more.

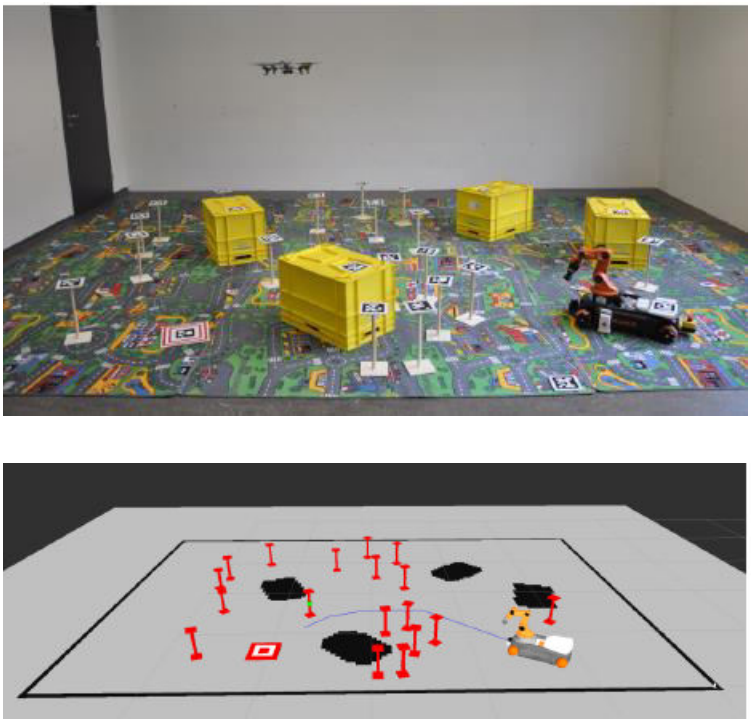


Fig.2.1 Overview of the setup of Mueggler et. al (2014)

Reference (Garz'on et al, 2015) presents a solution for multiple UGV to perform signal searching tasks in large outdoor scenarios. They propose different path planning strategies for coverage, which depend on the size and shape of the field. For the topic of UAV-UGV collaboration, (Totekar et al, 2013) studied the problem of coordinating UAV and UGV for precision agriculture, where they found energy efficient ways to visit areas with misclassified nitrogen levels. UAV and UGV have also been used in a collaborative manner to perform target localization Butzke et. al (2016). In (Mueggler et. al ,2014) a mock-up disaster scenario was setup, where a UAV maps the area and then computes the fastest mission for a UGV to reach the destination and deliver a first-aid kit. Cooperative environment mapping

(Michael et al, 2012) and surveillance (Saska et al, 2012) have also been studied. While our experiments are fairly specific, and therefore difficult to compare to existing approaches, Schneider et al (2015) discuss how EURATHLON and ELROB have provided a way of standardizing and benchmarking the evaluation of methods in outdoor robotics through competition. Teams at these competitions build impressive systems that are capable of executing missions in real-time for important tasks such as search and rescue. Others have used overhead imagery to improve UGV path planning capabilities. In (Sofman et al, 2006), a self-supervised online learning algorithm is used on a UGV to learn a model that integrates information about the current terrain and overhead imagery that is then used to predict traversal costs at other regions in the overhead map. These predicted traversal costs were then used to perform path planning. While many of these works demonstrate successful collaboration between UAV and UGV, we try to focus more on using semantic segmentation for scene understanding in a real-world search task by training on a dataset of imagery that is annotated with semantic categories. As more images are captured and annotated by low-flying aircraft, we believe it will be important to integrate existing models with online learning algorithms, such as the one presented in (Sofman et al, 2006). These models will be able to provide valuable context to a UGV during tasks such as radiation search, as existing maps (e.g. satellite) may be too old to capture important information about the scene.

The really specific and successful application to a real earthquake hit zone of the collaborative robotics idea came from (Michael et al, 2012), whose setup is shown in Fig.2.2. They report results from field experiments conducted with a team of ground and aerial robots engaged in the mapping of an earthquake-damaged building. We focus on the investigation of the feasibility of deploying aerial robots, specifically a quadrotor, into disaster scenarios where a building may be critically

damaged but is still accessible to robots and humans for experimental purposes. The experimental environment covered the top three floors of a building on the campus of Tohoku University in Sendai, Japan. Michael et al (2012) report results from field experiments conducted with a team of ground and aerial robots engaged in the collaborative mapping of an earthquake-damaged building. The goal of the experimental exercise is the generation of three-dimensional maps that capture the layout of a multi-floor environment. Michael et al (2012) provide details of the approach to the collaborative mapping and report results from the experiments in the form of maps generated by the individual robots and as a team. The first platform is a ground robot equipped with an onboard sensing suite that enables the generation of dense 3D maps. The vehicle is teleoperated through the multifloor environment while simultaneously collecting sensor data. After the operators identify locations in the environment that are inaccessible to the ground platform, a second ground platform equipped with an automated helipad is teleoperated to these locations and carries a quadrotor robot equipped with onboard sensing that is able to remotely open and close the helipad and autonomously take off and land from the helipad. Their map is shown in Fig.2.3.

Fig.2.2 Overview of setup of Michael et al (2012)

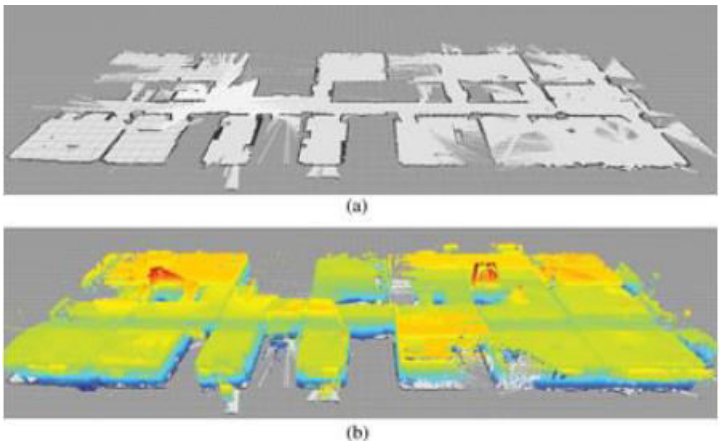


Fig.2.3 Mapping of the earthquake hit area, by Michael et al (2012)

2.2 STATE-OF-THE-ART

The state-of-the-art for collaborative robotics for search and rescue mission, once again comes from the Robotics and Perception group at ETH Zurich. In (Delmerico et al, 2016), they address the problem of planning a path for a ground robot through unknown terrain, using observations from a flying robot. In search and rescue missions, which are their target scenarios, the time from arrival at the disaster site to the delivery of aid is critically important. Previous works required exhaustive exploration before path planning, which is time-consuming but eventually leads to an optimal path for the ground robot. Instead, they propose active exploration of the environment, where the flying robot chooses regions to map in a way that optimizes the overall response time of the system, which is the combined time for the air and ground robots to execute their missions. In their approach, we estimate terrain classes throughout their terrain map, and we also add elevation information in areas where the active exploration algorithm has chosen to perform 3D reconstruction. This terrain information is used to estimate feasible and efficient paths for the ground robot. By exploring the environment actively, they achieve superior response times compared to both exhaustive and greedy exploration strategies. they demonstrate the

performance and capabilities of the proposed system in simulated and real-world outdoor experiments. This is the first work to address ground robot path planning using active aerial exploration. Their setup is shown in Fig.2.4.

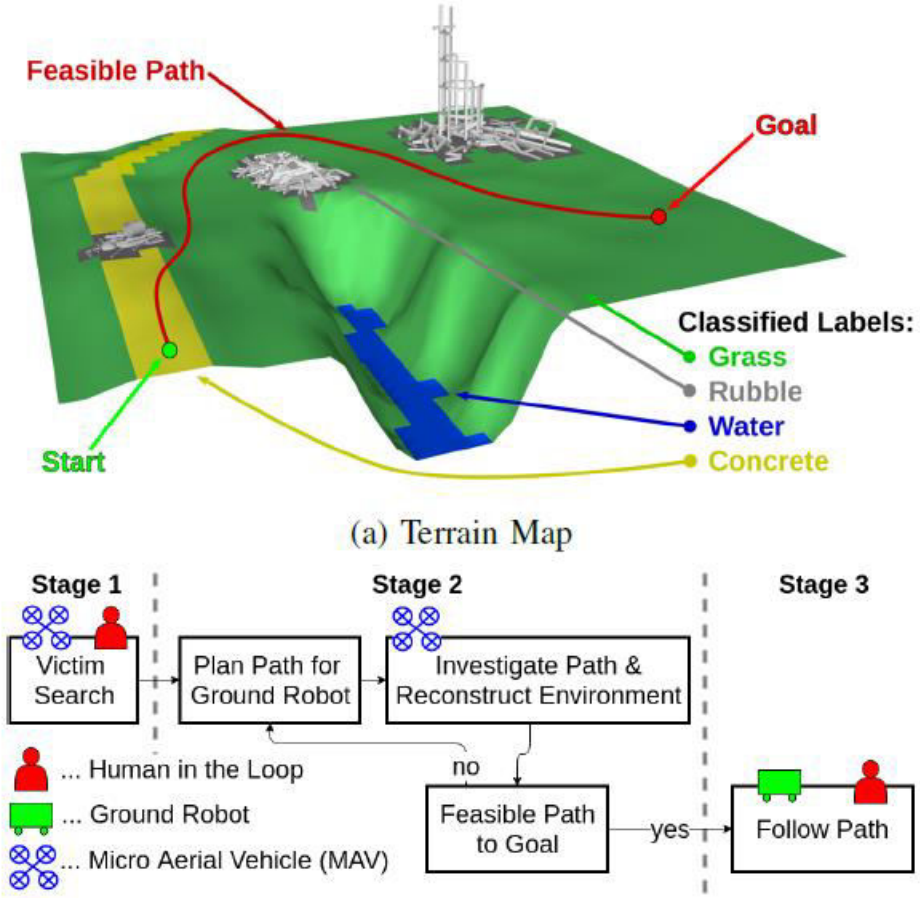


Fig.2.4 Setup of Delmerico et al, (2016)

2.3 SUMMARY

While the state-of-the-art has been established by (Delmerico et al, 2016), we will only focus on replicating most of the results of (Mueggler et al, 2014), owing to our limited time and resources. The rest of the report will focus on the relevant theory and the experimental implementations.

CHAPTER 3

COMPUTER VISION

3.1 INTRODUCTION

The reference (Szeliski, 2010) is a fantastic textbook on the computer vision and was our primary reference material with respect to the subject, in addition to the research papers that are published. This chapter is dedicated to the explanation of the entire computer vision system that is the backbone of the entire process. This chapter gives a very brief introduction to what computer vision is, and delves into the details of transformations, filters, RANSAC, scale-invariant feature transforms, their application for object detection and panoramic stitching, and finally, the Viola Jones face (and object) detection algorithm.

3.2 WHAT IS COMPUTER VISION?

As humans, we perceive the three-dimensional structure of the world around us with apparent ease. Think of how vivid the three-dimensional percept is when you look at a vase of flowers sitting on the table next to you. You can tell the shape and translucency of each petal through the subtle patterns of light and shading that play across its surface and effortlessly segment each flower from the background of the scene. Looking at a framed group portrait, you can easily count (and name) all of the people in the picture and even guess at their emotions from their facial appearance. Perceptual psychologists have spent decades trying to understand how the visual system works and, even though they can devise optical illusions to tease apart some of its principles, a complete solution to this puzzle remains elusive. Researchers in computer vision have been developing, in parallel, mathematical techniques for

recovering the three-dimensional shape and appearance of objects in imagery. We now have reliable techniques for accurately computing a partial 3D model of an environment from thousands of partially overlapping photographs. Given a large enough set of views of a particular object or facade, we can create accurate dense 3D surface models using stereo matching. We can track a person moving against a complex background. We can even, with moderate success, attempt to find and name all of the people in a photograph using a combination of face, clothing, and hair detection and recognition. However, despite all of these advances, the goal of having a computer interpret an image at the same level as a two-year old remains elusive. Why is vision so difficult? In part, it is because vision is an inverse problem, in which we seek to recover some unknowns given insufficient information to fully specify the solution.

3.3 GEOMETRIC TRANSFORMS

A general image is represented with the help of pixels and intensities. Each pixel has a specific intensity ranging from completely white to completely black, specified by certain designated numbers, and an RGB code. These images can be viewed differently from different angles and can be subjected to geometric transforms. Common geometric transforms that we will be seeing in this report are discussed here. We only discuss 2D transforms here. 2D translations can be written as

$$\bar{x}' = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \bar{x} \tag{3.1}$$

The rotation and translation operation is represented with ‘R’ and ‘t’ respectively, and is given by the below equation

$$x' = \begin{bmatrix} R & t \end{bmatrix} \bar{x} \tag{3.2}$$

where

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{3.3}$$

is an orthonormal rotation matrix with $RR^T = I$ and $|R| = 1$.

The similarity transform preserves the angles between lines, and is specified by a scale factor 's', and is given by the equation:

$$x' = \begin{bmatrix} sR & t \end{bmatrix} \bar{x} = \begin{bmatrix} a & -b & t_x \\ b & a & t_y \end{bmatrix} \bar{x}, \tag{3.4}$$

The affine transform preserves the parallel lines, and at the end of the transform, parallel lines remain parallel. This is given by the equation

$$x' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \bar{x}. \tag{3.5}$$

The visual representation of each of these transforms is shown below

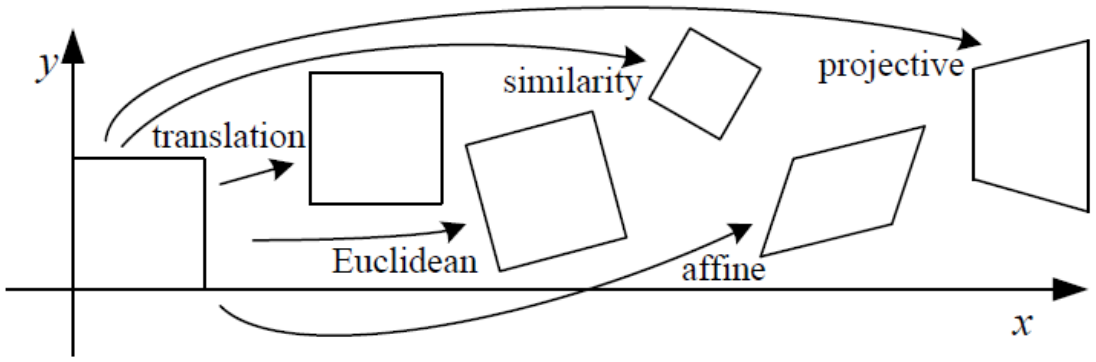


Fig.3.1 Geometric Transforms from Szeliski et. al (2010)

3.4 RANSAC

The RANSAC algorithm is a learning technique to estimate parameters of a model by random sampling of observed data. Given a dataset whose data elements contain both inliers and outliers, RANSAC uses the voting scheme to find the optimal fitting result. Data elements in the dataset are used to vote for one or multiple models. The implementation of this voting scheme is based on two assumptions: that the noisy features will not vote consistently for any single model (few outliers) and there are enough features to agree on a good model (few missing data). The RANSAC algorithm is essentially composed of two steps that are iteratively repeated:

1. In the first step, a sample subset containing minimal data items is randomly selected from the input dataset. A fitting model and the corresponding model parameters are computed using only the elements of this sample subset. The cardinality of the sample subset is the smallest sufficient to determine the model parameters.
2. In the second step, the algorithm checks which elements of the entire dataset are consistent with the model instantiated by the estimated model parameters obtained from the first step. A data element will be considered as an outlier if it does not fit the fitting model instantiated by the set of estimated model parameters within some error threshold that defines the maximum deviation attributable to the effect of noise.

The set of inliers obtained for the fitting model is called consensus set. The RANSAC algorithm will iteratively repeat the above two steps until the obtained consensus set in certain iteration has enough inliers.

The input to the RANSAC algorithm is a set of observed data values, a way of fitting some kind of model to the observations, and some confidence parameters. RANSAC achieves its goal by repeating the following steps:

1. Select a random subset of the original data. Call this subset the *hypothetical inliers*.
2. A model is fitted to the set of hypothetical inliers.
3. All other data are then tested against the fitted model. Those points that fit the estimated model well, according to some model-specific loss function, are considered as part of the *consensus set*.
4. The estimated model is reasonably good if sufficiently many points have been classified as part of the consensus set.
5. Afterwards, the model may be improved by re-estimating it using all members of the consensus set.

This procedure is repeated a fixed number of times, each time producing either a model which is rejected because too few points are part of the consensus set, or a refined model together with a corresponding consensus set size. In the latter case, we keep the refined model if its consensus set is larger than the previously saved model.

3.5 SCALE INVARIANT FEATURE TRANSFORM (SIFT)

The seminal paper on SIFT was given by Davide Lowe, in (Lowe, 2004). For any object in an image, interesting points on the object can be extracted to provide a "feature description" of the object. This description, extracted from a training image, can then be used to identify the object when attempting to locate the object in a test image containing many other objects. To perform reliable recognition, it is important

that the features extracted from the training image be detectable even under changes in image scale, noise and illumination. Such points usually lie on high-contrast regions of the image, such as object edges.

Another important characteristic of these features is that the relative positions between them in the original scene shouldn't change from one image to another. For example, if only the four corners of a door were used as features, they would work regardless of the door's position; but if points in the frame were also used, the recognition would fail if the door is opened or closed. Similarly, features located in articulated or flexible objects would typically not work if any change in their internal geometry happens between two images in the set being processed. However, in practice SIFT detects and uses a much larger number of features from the images, which reduces the contribution of the errors caused by these local variations in the average error of all feature matching errors.

SIFT can robustly identify objects even among clutter and under partial occlusion, because the SIFT feature descriptor is invariant to uniform scaling, orientation, illumination changes, and partially invariant to affine distortion. This section summarizes the original SIFT algorithm and mentions a few competing techniques available for object recognition under clutter and partial occlusion.

3.5.1 SIFT for Panoramic Stitching

From (Lowe and Brown, 2007), SIFT feature matching can be used in image stitching for fully automated panorama reconstruction from non-panoramic images. The SIFT features extracted from the input images are matched against each other to find k nearest-neighbours for each feature. These correspondences are then used to find m candidate matching images for each image. Transformations between pairs

of images are then computed using RANSAC and a probabilistic model is used for verification. Because there is no restriction on the input images, graph search is applied to find connected components of image matches such that each connected component will correspond to a panorama. Finally for each connected component Bundle adjustment is performed to solve for joint camera parameters, and the panorama is rendered using multi-band blending. Because of the SIFT-inspired object recognition approach to panorama stitching, the resulting system is insensitive to the ordering, orientation, scale and illumination of the images. The input images can contain multiple panoramas and noise images (some of which may not even be part of the composite image), and panoramic sequences are recognized and rendered as output. The outputs are shown from Fig.3.2-3.5.

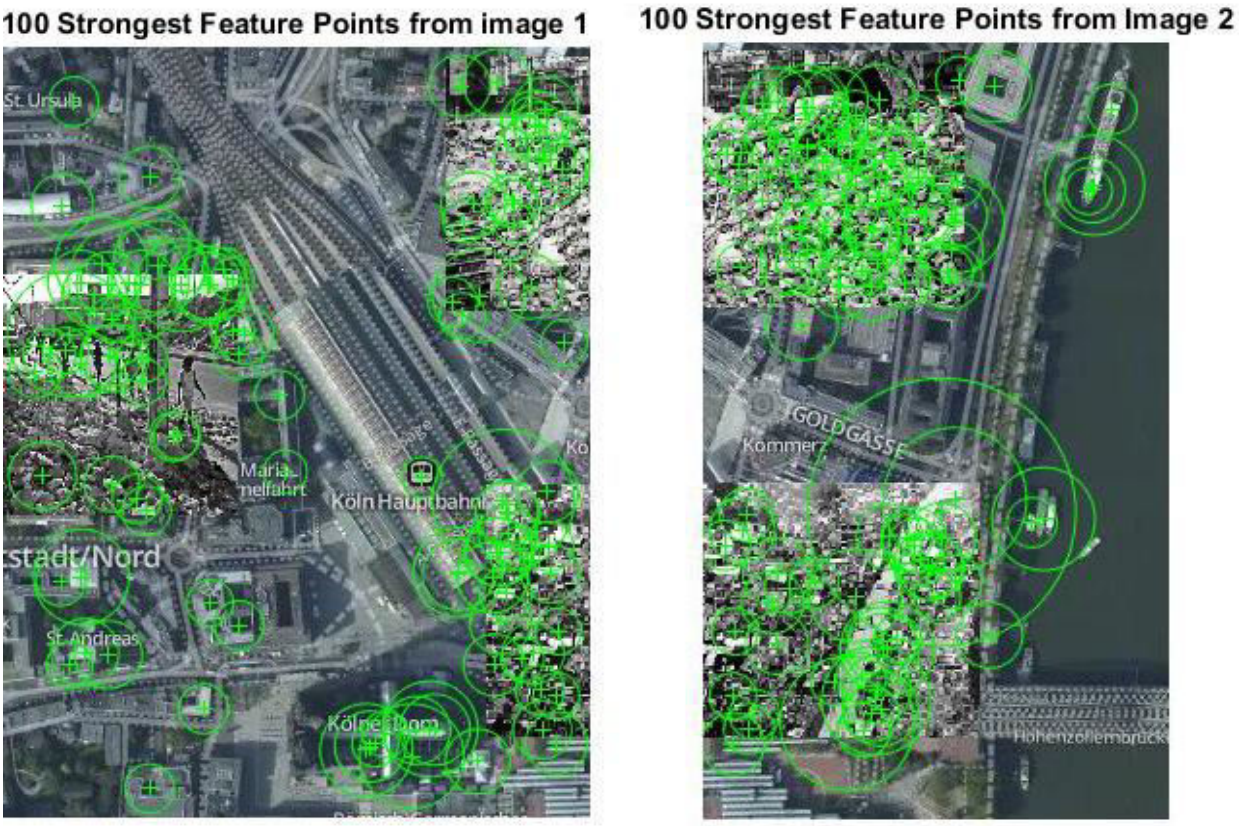


Fig.3.2 Features obtained from two given scenes

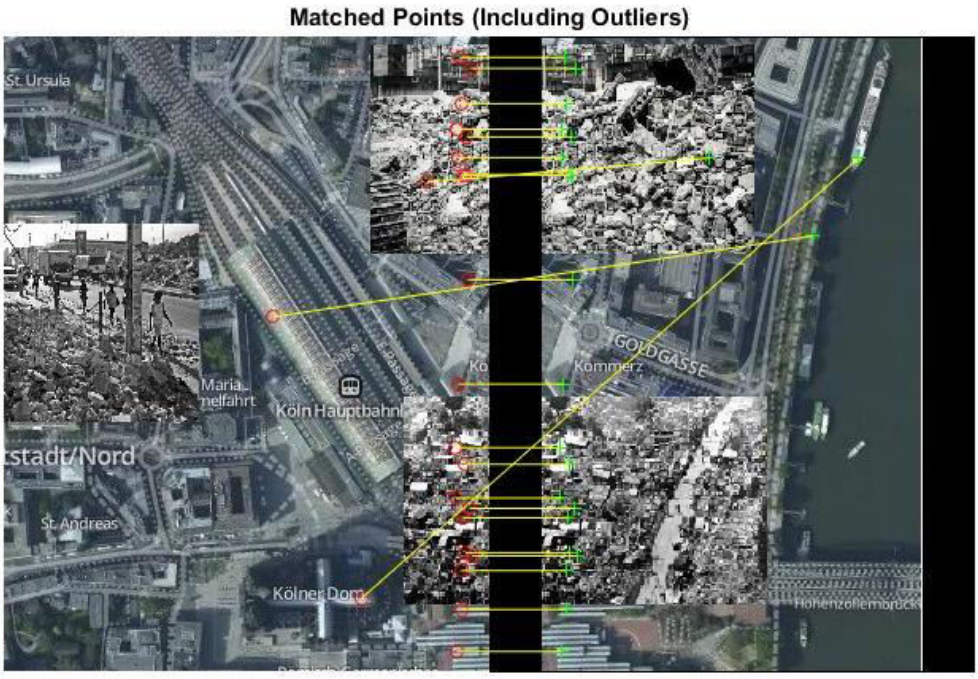


Fig.3.3 Feature match with outliers

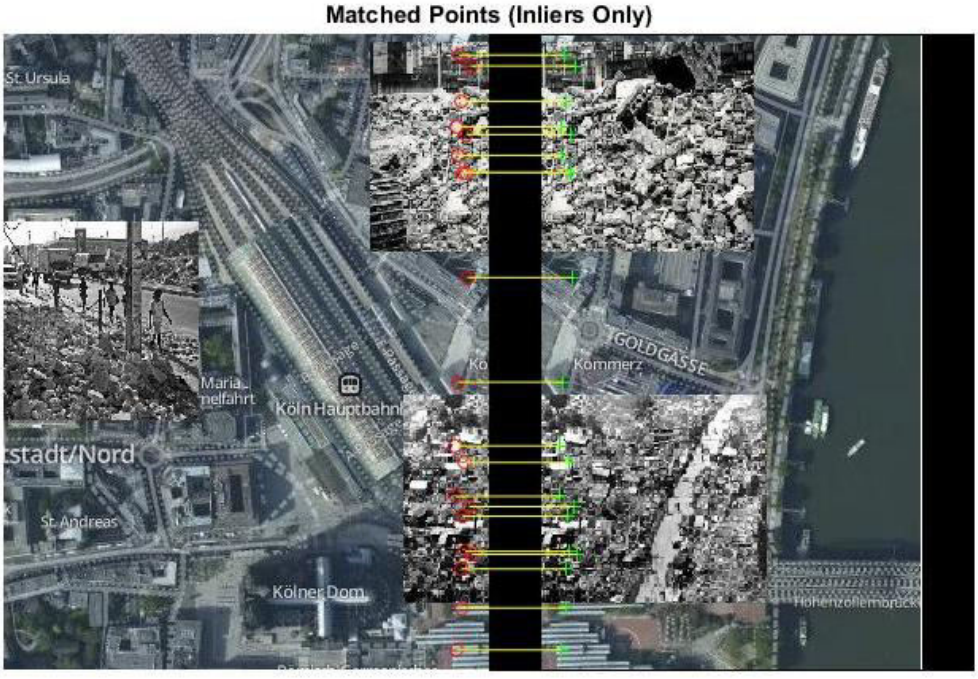


Fig.3.4 Feature match with Inliers

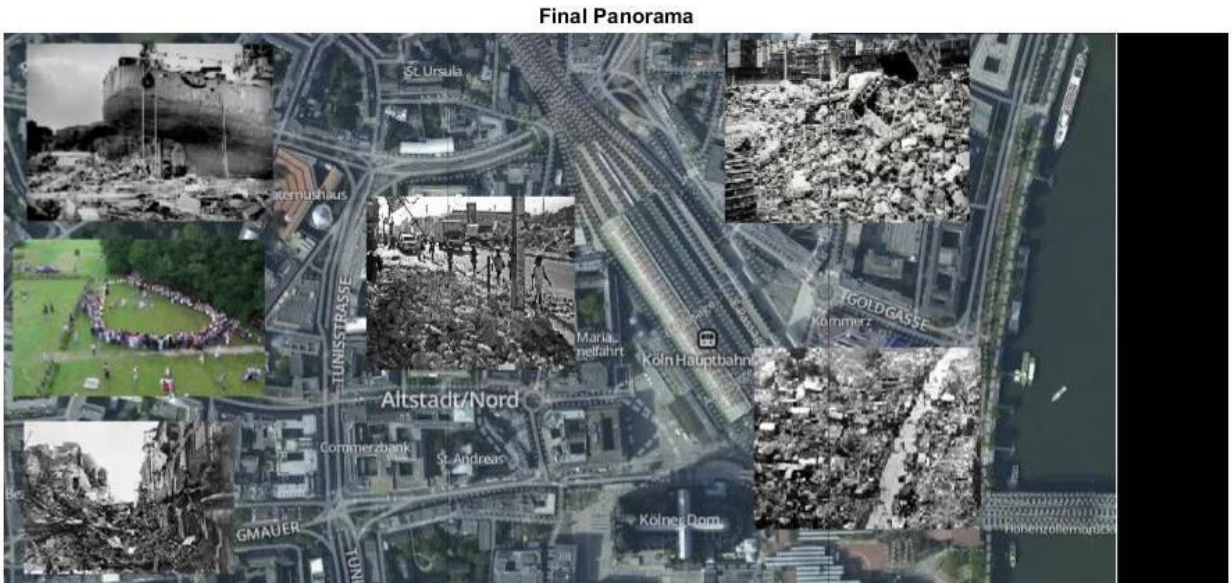


Fig.3.5 Final Panorama

3.5.2 SIFT for Object Detection

Given SIFT's ability to find distinctive keypoints that are invariant to location, scale and rotation, and robust to affine transformations (changes in scale, rotation, shear, and position) and changes in illumination, they are usable for object recognition. The steps are given below.

- First, SIFT features are obtained from the input image using the algorithm described above.
- These features are matched to the SIFT feature database obtained from the training images. This feature matching is done through a Euclidean-distance based nearest neighbor approach.
- Although the distance ratio test described above discards many of the false matches arising from background clutter, we still have matches that belong to different objects. Therefore, to increase robustness to object identification, we want to cluster those features that belong to the same object and reject the matches that are left out in the clustering process. This is done using the RANSAC. This

Fig.3.6 Detected debri features

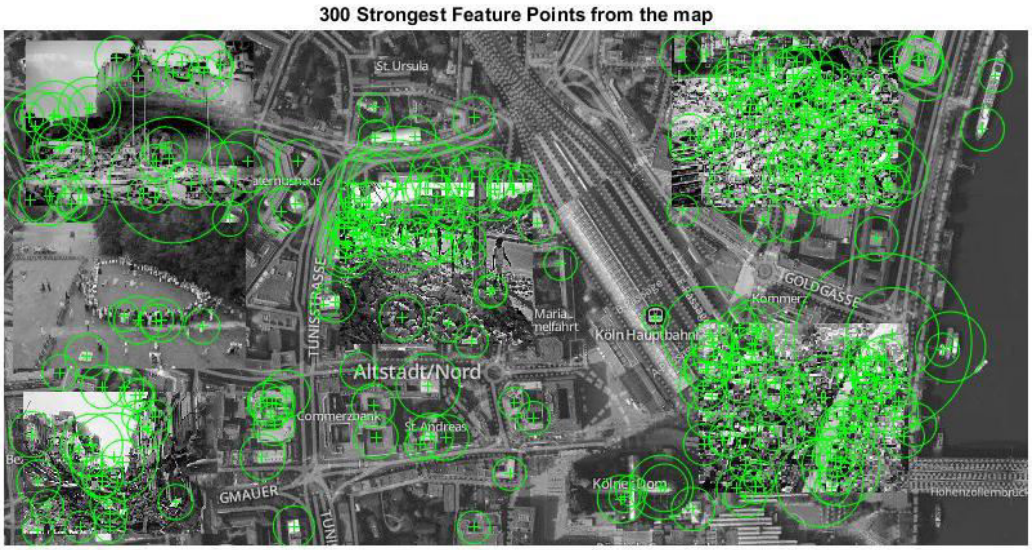


Fig.3.7 Detected map features

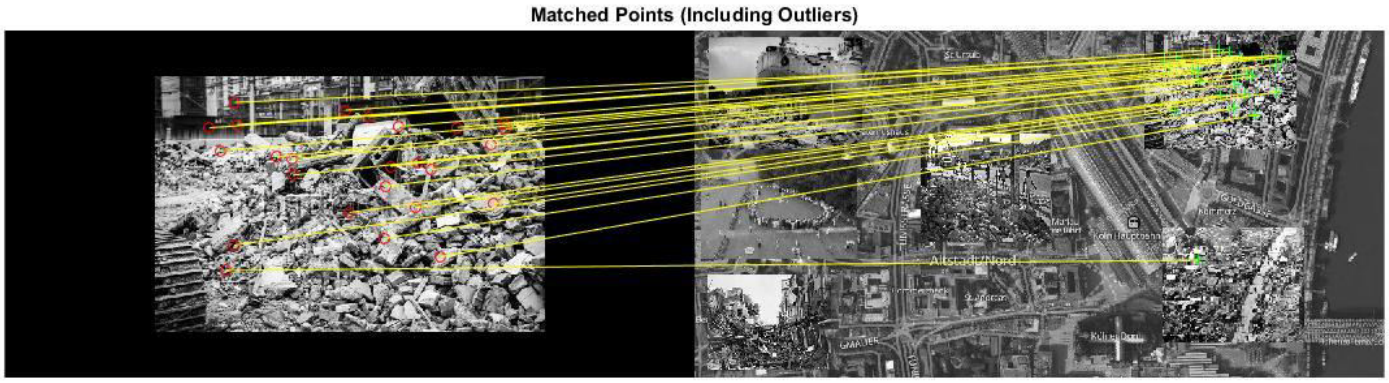


Fig.3.8 Matched points with outliers

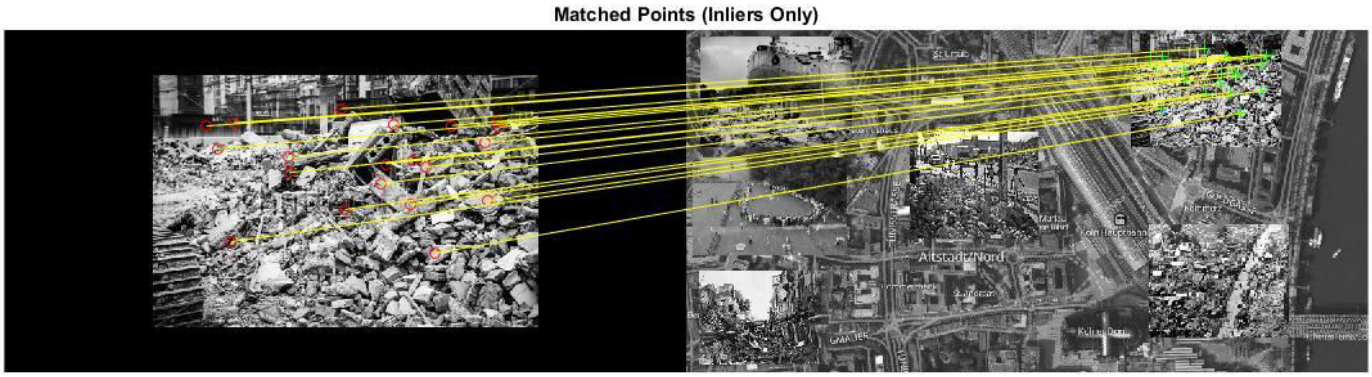


Fig.3.9 Matched points with Inliers

Fig.3.10 Detected Objects

- will identify clusters of features that vote for the same object pose. When clusters of features are found to vote for the same pose of an object, the probability of the interpretation being correct is much higher than for any single feature. Each keypoint votes for the set of object poses that are consistent with the keypoint's location, scale, and orientation. *Bins* that accumulate at least 3 votes are identified as candidate object/pose matches.
- For each candidate cluster, a least-squares solution for the best estimated affine projection parameters relating the training image to the input image is obtained. If the projection of a keypoint through these parameters lies within half the error range that was used for the parameters in the RANSAC bins, the keypoint match is kept. If fewer than 3 points remain after discarding outliers for a bin, then the object match is rejected. The least-squares fitting is repeated until no more rejections take place. This works better for planar surface recognition than 3D object recognition since the affine model is no longer accurate for 3D objects.

SIFT features can essentially be applied to any task that requires identification of matching locations between images. The outputs for our code, which was written on MATLAB, is given below, Fig.3.6-3.10. The procedure is discussed in (Lowe, 2004).

3.6 VIOLA JONES OBJECT AND FACE DETECTION

The Viola–Jones object detection framework is the first object detection framework to provide competitive object detection rates in real-time proposed in 2001 by Paul Viola and Michael Jones, in (Viola and Jones, 2001). Although it can be trained to detect a variety of object classes, it was motivated primarily by the problem of face detection. A sample output of the face detection is shown in Fig.3.11.

The characteristics of Viola–Jones algorithm which make it a good detection algorithm are:

- Robust – very high detection rate & very low false-positive rate always.
- Real time – For practical applications at least 2 frames per second must be processed.
- Face detection only - The goal is to distinguish faces from non-faces (detection is the first step in the recognition process).

The algorithm has four stages: Haar Feature Selection, Creating an Integral Image, Adaboost Training, Cascading Classifiers.

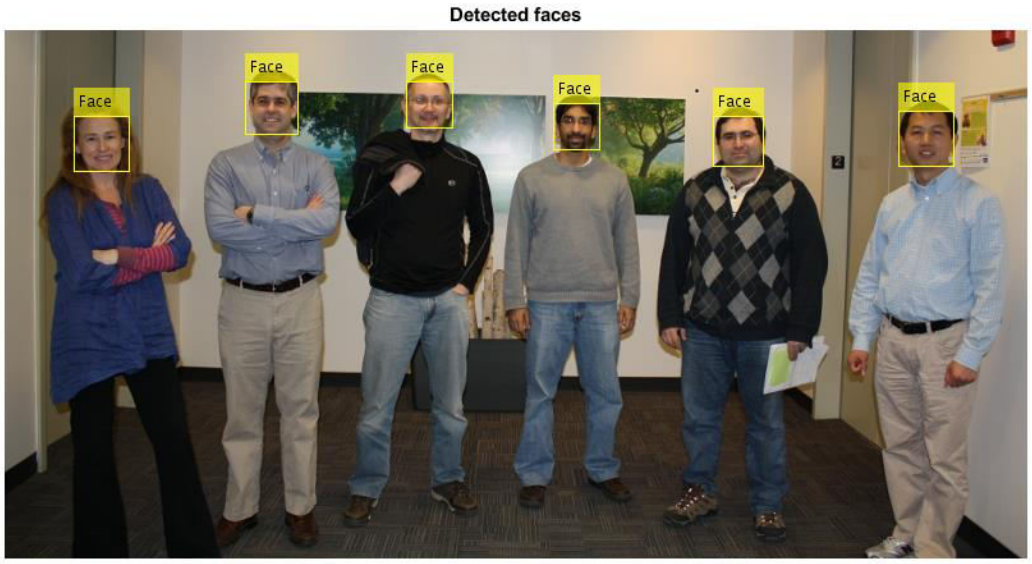


Fig.3.11 Detected Faces using Viola Jones algorithm

3.7 SUMMARY

This chapter discussed the various computer vision algorithms that we will be using in this project work, the reference of which can be found in the textbook (Szeliski, 2010).

CHAPTER 4

MOTION PLANNING

4.1 INTRODUCTION

The defining reference for understanding all things motion planning is (LaValle, 2006). This chapter is dedicated to the study of motion planning algorithms. This chapter covers the combinatorial and sampling-based search algorithms and focuses on why the sampling techniques are much better than the combinatorial techniques. This chapter finally focuses on our own mission planning algorithm that we developed, which would prioritize distances and the density of populations to minimize the overall time for the mission.

4.2 WHAT IS MOTION PLANNING?

It is the process of breaking down a desired movement task into discrete motions that satisfy movement constraints and possibly optimize some aspect of the movement. For example, consider navigating a mobile robot inside a building to a distant waypoint. It should execute this task while avoiding walls and not falling down stairs. A motion planning algorithm would take a description of these tasks as input and produce the speed and turning commands sent to the robot's wheels. Motion planning algorithms might address robots with a larger number of joints, like industrial manipulators, more complex tasks, like manipulation of objects, different constraints, like a car that can only drive forward, and uncertainty, like imperfect models of the environment or robot. Exact motion planning for high-dimensional systems under complex constraints is computationally intractable. Potential-field algorithms are efficient but fall prey to local minima. Sampling-based algorithms

avoid the problem of local minima and solve many problems quite quickly. They are unable to determine that no path exists, but they have a probability of failure that decreases to zero as more time is spent. Sampling-based algorithms are currently considered state-of-the-art for motion planning in high-dimensional spaces and have been applied to problems which have dozens or even hundreds of dimensions (robotic manipulators, biological molecules, animated digital characters, and legged robots). We will now look at grid-based search, an example of which is Dijkstra's Algorithm, and look at sampling-based search, an example of which is probabilistic roadmap.

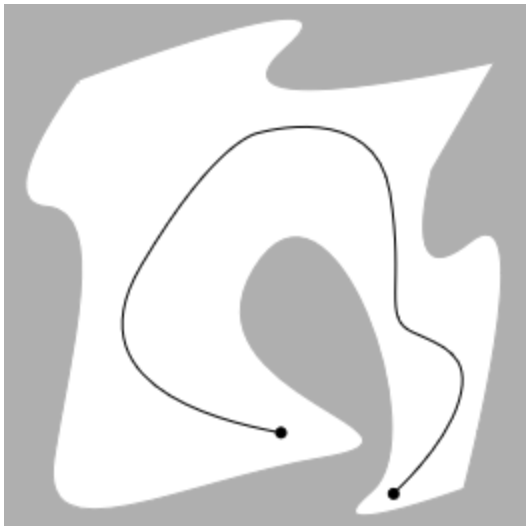


Fig.4.1 An example of motion planning on an occupancy grid, with the grey places representing obstacles and white places representing free spaces.

Fig. 4.1 gives an example of motion planning. The configuration space is basically represented as union of all possible configurations of the environment. Where the robot cannot move in, both physically and higher dimensionally, are marked as one, and where the robot can move, those spaces are called free spaces and are marked as zero. This specific setup is called the binary occupancy grid. An example of an inverted binary occupancy grid is shown below in Fig.4.2.

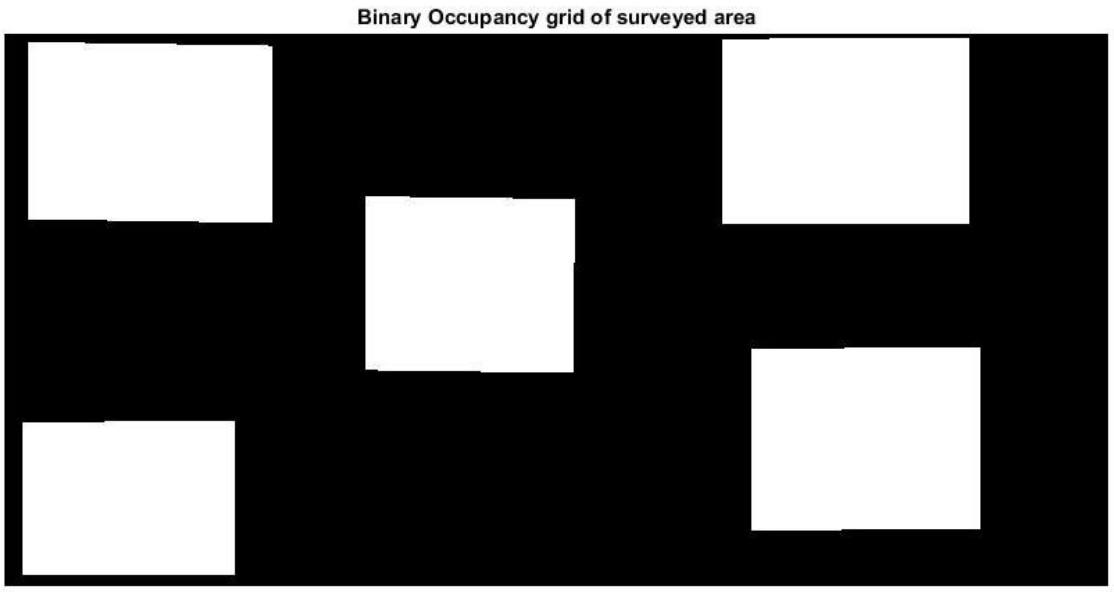


Fig.4.2 An example of an inverted binary occupancy grid.

4.3 DIJKSTRA'S ALGORITHM

Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. This is a graph-based search algorithm.

The algorithm exists in many variants; Dijkstra's original variant found the shortest path between two nodes, but a more common variant fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph, producing a shortest-path tree.

For a given source node in the graph, the algorithm finds the shortest path between that node and every other. It can also be used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined.

Let the node at which we are starting be called the initial node. Let the distance of node Y be the distance from the initial node to Y. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step.

1. Mark all nodes unvisited. Create a set of all the unvisited nodes called the unvisited set.
2. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes. Set the initial node as current.
3. For the current node, consider all of its unvisited neighbors and calculate their tentative distances through the current node. Compare the newly calculated tentative distance to the current assigned value and assign the smaller one. For example, if the current node A is marked with a distance of 6, and the edge connecting it with a neighbor B has length 2, then the distance to B through A will be 6 + 2 = 8. If B was previously marked with a distance greater than 8 then change it to 8. Otherwise, keep the current value.
4. When we are done considering all of the neighbors of the current node, mark the current node as visited and remove it from the unvisited set. A visited node will never be checked again.
5. Move to the next unvisited node with the smallest tentative distances and repeat the above steps which check neighbors and mark visited.
6. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the unvisited set is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.
7. Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new "current node", and go back to step 3.

When planning a route, it is actually not necessary to wait until the destination node is "visited" as above: the algorithm can stop once the destination node has the

smallest tentative distance among all "unvisited" nodes (and thus could be selected as the next "current").

For an example graph, the full output of the Dijkstra's algorithm is shown below in Fig. 4.3.

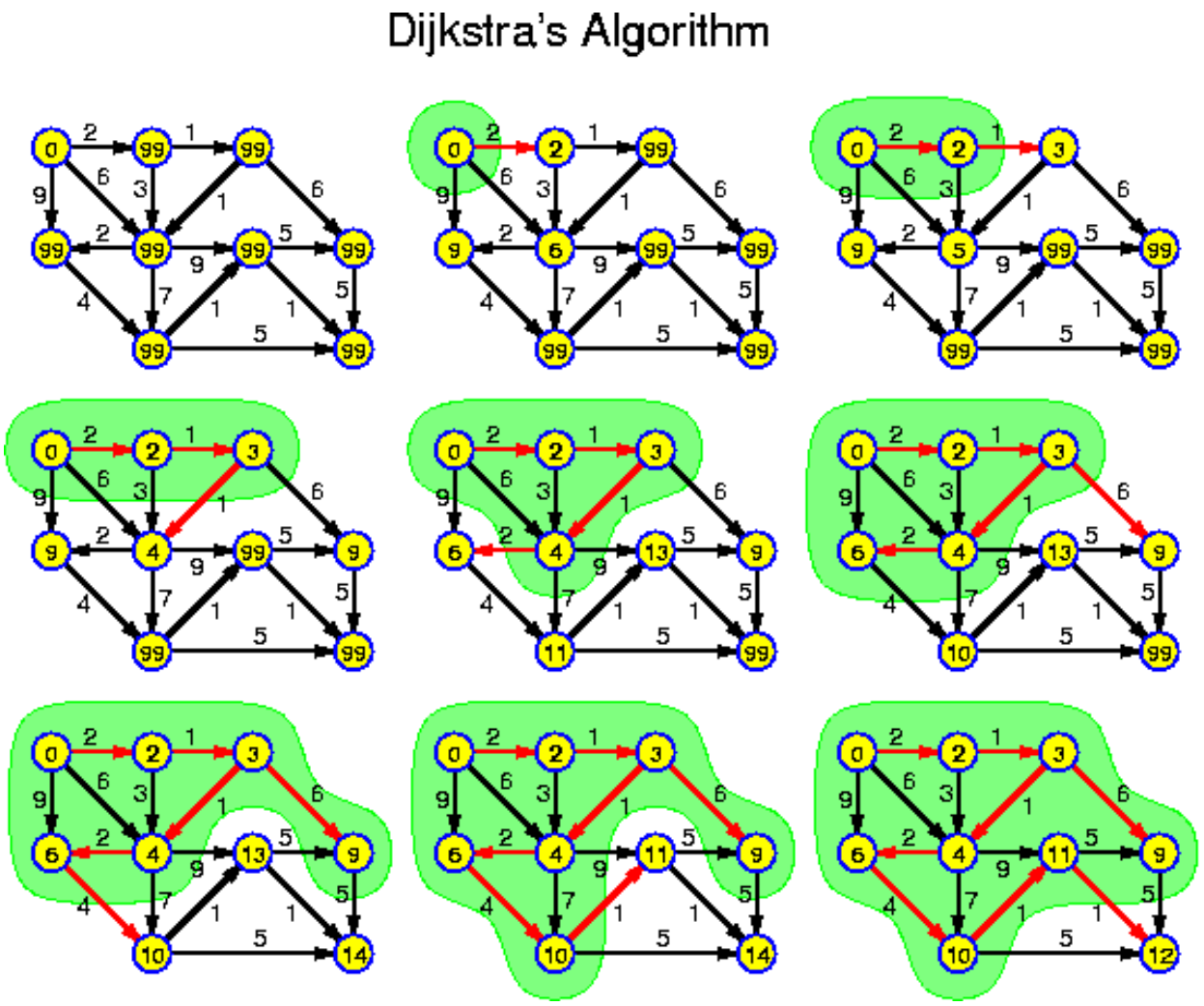


Fig.4.3 Output of Dijkstra's algorithm on a sample graph

4.4 PROBABILISTIC ROADMAP

Probabilistic Roadmap, or PRM, is a motion planning algorithm that is not graph-based but is sampling-based. The basic idea behind PRM is to take random samples from the configuration space of the robot, testing them for whether they are in the

free space, and use a local planner to attempt to connect these configurations to other nearby configurations. The starting and goal configurations are added in, and a graph search algorithm is applied to the resulting graph to determine a path between the starting and goal configurations.

The probabilistic roadmap planner consists of two phases: a construction and a query phase. In the construction phase, a roadmap (graph) is built, approximating the motions that can be made in the environment. First, a random configuration is created. Then, it is connected to some neighbours, typically either the k nearest neighbours or all neighbours less than some predetermined distance. Configurations and connections are added to the graph until the roadmap is dense enough. In the query phase, the start and goal configurations are connected to the graph, and the path is obtained by a Dijkstra's shortest path query.

Given certain relatively weak conditions on the shape of the free space, PRM is provably probabilistically complete, meaning that as the number of sampled points increases without bound, the probability that the algorithm will not find a path if one exists approaches zero. The rate of convergence depends on certain visibility properties of the free space, where visibility is determined by the local planner. Roughly, if each point can "see" a large fraction of the space, and also if a large fraction of each subset of the space can "see" a large fraction of its complement, then the planner will find a path quickly. An example run of the PRM algorithm on the binary occupancy grid that was shown earlier is given in Fig. 4.4. The large number of nodes represent the sampling that was done on the original configuration space, and the network is connected using the k-nearest neighbours algorithm that was described earlier. On the connected graph, Dijkstra’s algorithm is run to find the shortest path between any two points, and the green lines that run from one end to another represent that shortest path that was found.

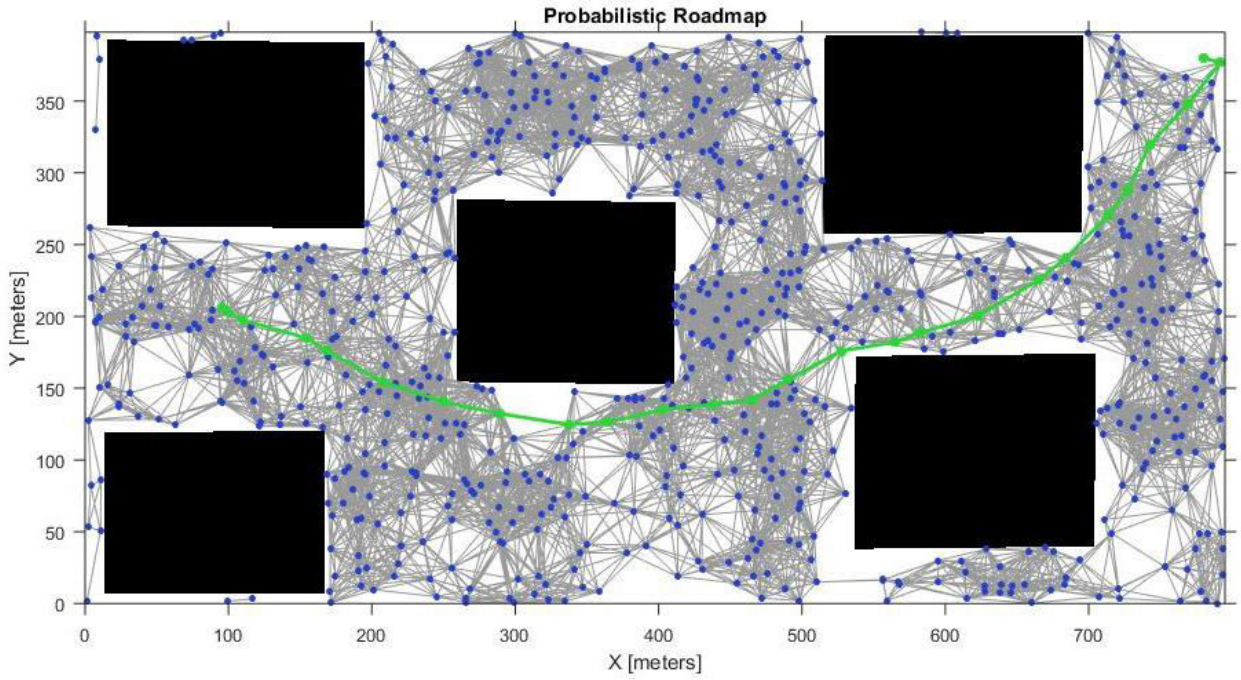


Fig.4.4 Output of PRM on the given binary occupancy grid

4.4.1 Probabilistic Roadmap VS. Dijkstra's algorithm

For our purposes in this project, we have chosen a sampling-based search strategy rather than an all-out graph-based search like Dijkstra's algorithm. The computational complexity of Dijkstra's algorithm, or any graph-based search for that matter, scales horribly as the number of nodes increases. This gets worse as the number of dimensions increases, as the number of nodes exponentially increases. The final result of the shortest path, although guaranteed through graph-based searches, will take an extremely long time, and will not be preferable for any real-time applications or any complex applications like manipulators and grasping. The sampling-based searches, such as PRM, although cannot guarantee that a result will be found every time the program is run, can guarantee that in a probabilistic sense, there will always be a result to the problem. Furthermore, the main advantages of the sampling-based methods, as opposed to the graph-based searches, is that they are very fast in their results. Hence, we are trading accuracy and speed, and in our case,

both seem to favour the sampling-based search, and hence we have chosen to use the PRM technique in our project.

4.5 MISSION PLANNING

Our own work in the process of motion planning lies with the mission planning algorithm that we developed. Mission planning algorithms are those which are combined with the path planning algorithms for a specific mission or purpose. This mission could be to serve the maximum number of people in a restaurant, or to save the maximum number of people, or to save energy while travelling, or to go around places in minimum time, and so on. For our case, we have taken different population densities located at different places and have given the constraint that the ground robot should traverse all the nodes and come back to the initial starting position in minimum time. We have also introduced the constraint that the ground vehicle should reach places where the population density is high, thereby serving maximum number of people before any mishap happens to the ground vehicle. To do this, we developed a cost function that represents this problem. This will be a trade-off between the distance to be travelled and the people to rescue, and one such example case is that even though the distance might be the greatest, a specific population will receive the highest priority if the number of people in that population group is the highest. This cost function can be represented by an inverse relationship between the distance and the number of people. For every possible permutation of the nodes configuration, the cost function is evaluated. The configuration that the ground vehicle will visit the nodes in is chosen in such a way that the cost function is a minimum.

4.6 SUMMARY

This chapter discussed the main path planning algorithm that we will be using, and also discussed our own modifications to it through mission planning algorithms.

CHAPTER 5

HARDWARE

5.1 INTRODUCTION

Wheeled robots are robots that navigate around the ground using motorized wheels to propel themselves. This design is simpler than using treads or legs and by using wheels they are easier to design, build, and program for movement in flat, not-so-rugged terrain. They are also well controlled than other types of robots. Their differential steering provides low cost and simplicity. Robots can have any number of wheels, but three wheels are sufficient for static and dynamic balance. Additional wheels can add to balance.

Our initial design for the ground vehicle had 2-by-2 powered (Fig 5.1) wheels for tank-like movement. This kind of robot uses 2 pairs of powered wheels. Each pair (connected by a line) turn in the same direction. The tricky part of this kind of propulsion is getting all the wheels to turn with the same speed. If the wheels in a pair are not running with the same speed, the slower one will slip (inefficient). If the pairs do not run at the same speed the robot won't be able to drive straight.

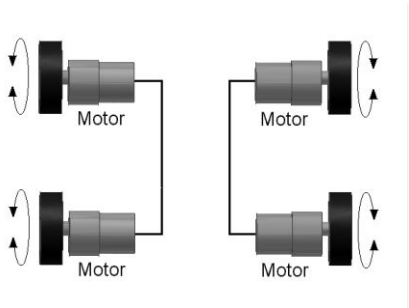


Fig 5.1- 2-by-2 powered wheels

Hence for the purpose of accurate turning, the design has been changed to a three wheeled vehicle in which two are differentially steered and one is freely rotating

(Fig 5.2). The centre of gravity in this type of robot has to lay inside the triangle formed by the wheels. If too heavy of a mass is mounted to the side of the free rotating wheel, the robot will tip over.

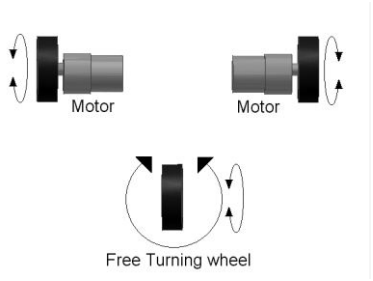


Fig 5.2 – Three wheeled vehicle

5.2 MOTOR CONTROLLER

The ground robot works on ARDUINO platform. The differentially steered wheels are attached to motors that are powered using a Quad Motor Driver Shield (Fig 5.3). The driver shield can control four motors at a time. It includes two TB6612FNG motor driver chips. When compared with the traditional L298N chip, efficiency is improved and the component size is also greatly reduced. The chip doesn't heat in to the rated range, and generates a single path maximum output 1.2A continuous current. The module includes a built-in low voltage detection circuit and thermal shutdown protection circuit, which is safe and reliable. The drive shield with motors are connected to the digital pins 3, 4, 5, 6, 7, 8, 11 and 12 of an ARDUINO MEGA 2560 board.

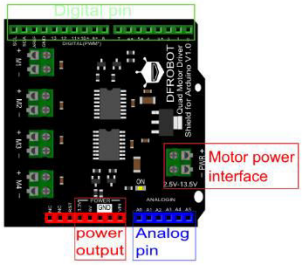


Fig 5.3 - Quad Motor Driver Shield

5.3 WiFi MODULE

The ground robot is equipped with the ability to transfer data wirelessly to a remote desktop (Master node) using ESP8266 WiFi module. ESP8266 is an impressive, low cost WiFi module suitable for adding WiFi functionality to an existing microcontroller project via a UART serial connection. The hardware connections for powering up the WiFi module and transferring data to and from the module are extremely simple (Fig 5.4).

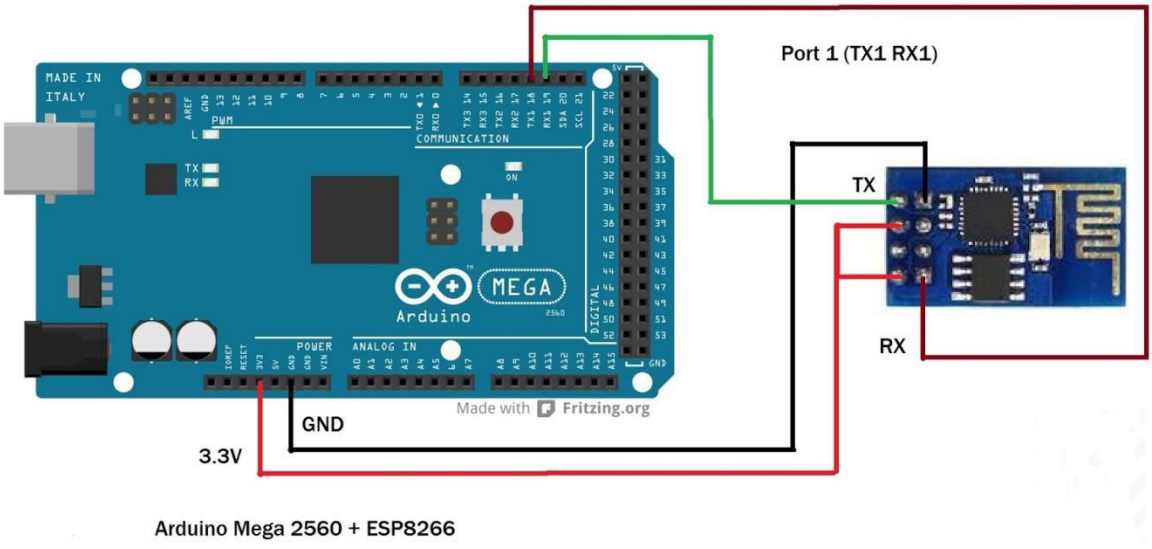


Fig 5.4 – Hardware connection between Esp8266 and MEGA 2560

The programming part to send and receive data can be done using AT commands. It is necessary to ensure that we receive proper response (which can be viewed on the serial monitor) for the basic AT commands. The list of basic AT commands, their functions and their response are summarised here in Table 5.1

Table 5.1 – list of basic AT commands

Command	Function	Response
AT	Test if AT system works correctly	OK
AT+RST	Reset the module	OK
AT+GMR	Print firmware version	*version*, OK
AT+GSLP = *time*	Enter deep sleep mode for *time* milliseconds	*time* OK

Few commands related to establishing the WiFi settings, identifying the available WiFi networks are, given in Table 5.2.

Table 5.2 – commands related to establishing the WiFi settings

AT+CWMODE= *mode*	Set AP's info which will be connect by ESP8266	OK
AT+CWJAP= *ssid,pwd*	Commands ESP8266 to connect a SSID with supplied password	OK
AT+CWLIF	List information on of connected clients	*ip*, other information

ESP8266, when connected to a system, can act as a TCP client/server and the related AT commands in Table 5.3 are,

Table 5.3 – Related AT commands with respect to ESP8266

AT+CIPMUX=*mode*	Enable / disable multiplex mode	OK
AT+CIPSTATUS	Get information about connection	STATUS:*status*
AT+CIPSTART=*type,addr,port*	Start a connection as client	OK
AT+CIPSEND=*length*	Set length of the data that will be sent	SEND OK
AT+CIFSR	Get local *IP* address	+CIFSR:*ip* OK
AT+CIPSERVER=*mode[,port]*	Configure ESP8266 as server	OK
telnet	Connect another device to the listening port on the same network	Link
+IPD	Receive network data from single connection	+IPD,*len:data*

5.4 THE ThingSpeak PLATFORM

For the ground robot to move from one location to another, motor commands are generated based on the range and bearing calculation which is performed on MATLAB in a remote desktop. To transfer the motor commands wirelessly to the

ground robot, it is necessary to have a common platform interfaceable with both MATLAB and ARDUINO so that motor commands and acknowledgements can be sent and received using HTTP requests. One such platform is **ThingSpeak**.

Fig 5.5 Data transfer via ThingSpeak

ThingSpeak is an open source Internet of Things (IoT) platform to store and retrieve data from things using the HTTP protocol over the Internet or via a Local Area Network. ThingSpeak allows users to create channels with specific fields, allows them to upload/read data from specific fields of the channels using write/read keys respectively. The outline of the process is shown in Fig.5.5.

5.5 OBJECT DETECTION MODULES

In order to detect the presence of obstacles in robot's path, the ground robot is equipped with an ultrasonic sensor (ARDUINO interfaceable) and a camera module (RASPBERRY PI interfaceable). The camera module detects the presence of the obstacle and the ultrasonic sensor determines the obstacle's distance from the robot's position.

5.5.1 Camera Module

Raspberry Pi camera module V2 is an 8 MP device which can take photos and record videos. Its horizontal and vertical field of view are 62.2° and 48.8° . The frame rate can go up to 90fps. The image output can be obtained in the following formats: JPEG (accelerated), JPEG + RAW, GIF, BMP, PNG, YUV420, RGB888. The video out is available in only in raw h.264 (accelerated). The camera module requires almost

200 – 250 mA of current. The connection of a camera module with a Raspberry Pi board is a one step process and is shown below (Fig 5.6). The ribbon cable has to be firmly seated on the Camera Serial Interface (CSI).

Fig 5.6 Hardware connection between camera module and Rapberry Pi board

There are four applications provided: *raspistill, raspivid, raspiyuv* and *raspividyuv. raspistill* and *raspiyuv* are very similar and are intended to capture images while *raspivid* and *raspividyuv* are for capturing videos.

All the applications are driven from the command line, and written to take advantage of the MMAL API which runs over OpenMAX. The MMAL API provides an easier to use system than that presented by OpenMAX.

5.5.2 Ultrasonic Sensor Module

The Ultrasonic sensor module used is HC-SR04. The module provides 2cm to 40 cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back. The time from sending the ultrasonic pulse to receiving the echo pulse is recorded using a built in counter. From the recorded time, the object's distance can be determined as follows. The connection is shown in Fig.5.7.

Test distance = (recorded time × velocity of sound) / 2 (5.1)

The device can detect any object that lies within a span of 15°. 5V DC and 15mA current is required for the proper functioning of the device.

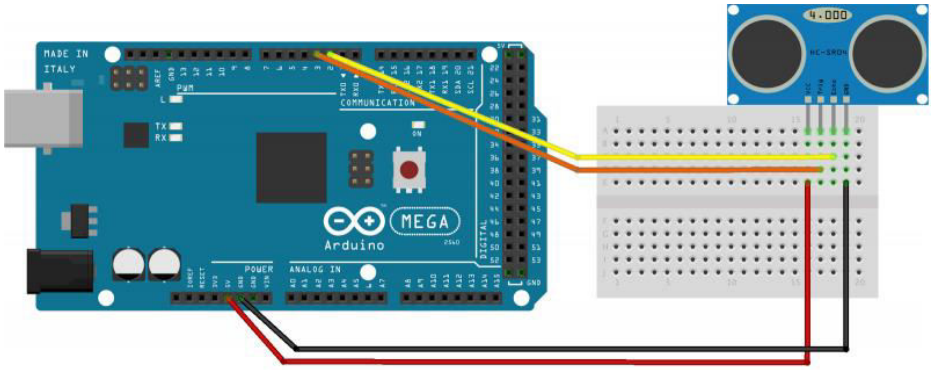


Fig 5.7 – Hardware connection between HC SR04 and MEGA 2560

5.6 POWER SOURCES

The ARDUINO MEGA 2560 and the RASPBERRY PI type 3 model B are powered up using different power sources. In order to supply sufficient current to the two motors, WiFi module and the Ultrasonic sensor, six rechargeable AA batteries, each with 2800mAh and 1.2V are used. According to the specifications of the camera module, the RASPBERRY PI board is powered with a power bank that can provide 5V and 2A current.

5.7 SUMMARY

In this chapter, a brief description about the components present in the ground robot, their specifications, functionalities, interconnections and the method of programming involved has been given.

CHAPTER 6

IMPLEMENTATION AND RESULTS

6.1 INTRODUCTION

In this chapter, a brief description about the experimental setup is given. A mock disaster zone was setup in an area of about 7×9 square metres. A total of 30 obstacles, each with size 60×65 square centimetres, were designed and laid out. A single obstacle was designed by taking disaster zone images from the internet and pasting six of them in a Cardboard. The obstacles were lifted to a height by placing bushes below them. Goal points are indicated by the images of people and about four goal points were setup with different number of people in each goal point.

The overview of the setup is shown in Fig.6.1.

Fig 6.1 – Our setup

6.2 SAMPLING OF THE AERIAL VEHICLE'S VIDEO

The aerial vehicle that we used initially was a toy RC helicopter, to which a Raspberry Pi interfaceable camera was attached. The aerial vehicle was unable to hover over a particular location for an interval of 2 or 3 seconds which resulted in a

very bad footage which could not be clearly sampled. Also such RC helicopters are not suggested during relatively strong wind currents.

The RC helicopter was replaced by an RC Quadrotor was a belief that a quadrotor could have better stability than a helicopter. The Quadrotor was built from a Do It Yourself construction kit. This device with a Raspberry Pi interfaceable camera at the bottom, was able to hover for 2 to 3 seconds over a particular location. But it suffered from maintaining the required height and repeatedly collided with the obstacles. All the aforementioned problems were solved by using a STANDARD DJI PHANTOM 3, with which the entire mock disaster setup was surveilled and a video was obtained. The sampling of the video from the aerial vehicle is shown in Fig.6.3

Fig 6.2 – DJI PHANTOM 3

6.3 COMPUTER VISION

The usage of computer vision in our setup is discussed. The sampled images were stitched, and the stitched image is shown in Fig.6.4. The obstacles were detected using the SIFT object detection algorithm, where our reference image was the obstacle taken at a close-up, and which recognized the obstacles in the stitched image. We transformed this into the binary occupancy grid, which is shown in Fig.6.5.

Fig 6.3 – Some set of sampled images from the video

Fig 6.4 – A section that was stitched

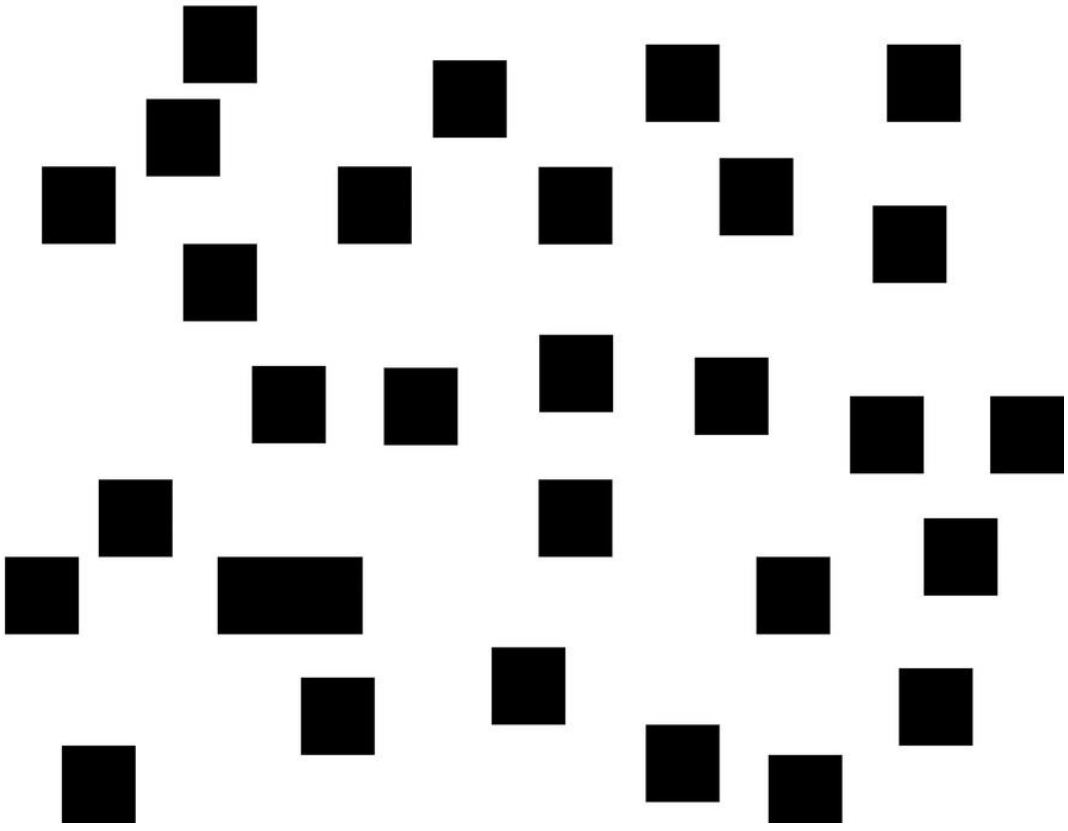


Fig 6.5 – Binary occupancy grid

6.4 PATH, MISSION, AND RE-PLANNING

The ground robot was initially a four wheeled robot that works on an ARDUINO platform (Chapter 3), and is shown in Fig.6.6. The tricky part of this kind of propulsion is getting all the wheels to turn with the same speed. If the wheels in a pair do not in a pair are not running with the same speed, the slower one will slip (inefficient). If the pairs do not run at the same speed. So the four wheeled ground robot was replaced with a three wheeled ground robot. This works well with less control strategies, only in a perfectly smooth terrain. In rugged terrain, the third free wheel turns randomly causing severe problems to the robot motion. This can be only corrected by using robust control strategies. So the design was switched to the initial four wheeled system and the problems mentioned above can be tackled by using rotary encoders which can precisely identify the distance moved by each wheel.

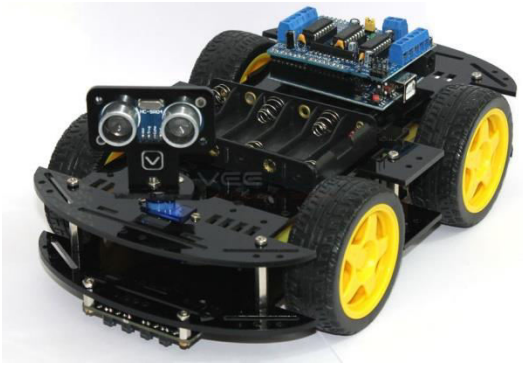


Fig 6.6 – Four wheeled ground robot

This video obtained from the drone was sampled suitably in MATLAB to obtain images of the disaster setup such that there exists a partial overlap between any two images. A stitching algorithm was run in MATLAB, which connected the overlapped images correctly to obtain a panoramically stitched image.

In the panoramically stitched image, obstacles can be identified by an object detection algorithm and the presence of the obstacles were blackened out to obtain

a binary occupancy grid. If the initial location of the robot and the final goal point are mentioned, the Probabilistic Road Map tool in the Robotics toolbox of MATLAB, determines the shortest path between these two locations which varies with the size of the ground robot.

The Probabilistic Road Map tool has been run iteratively, considering four goal points. The goal points contain different number of people and hence choosing the order of points to be traversed was done by building a cost function. The cost function at any time takes the ground robot's current position, one of the possible destinations for the ground robot, distance of that destination from the current position and the number of people to be rescued in that destination. The cost function is directly proportional to the distance between the two points and inversely proportional to the number of people rescued at that particular trip.

For the four goal points, all possible combinations were identified and the cost was evaluated. The solution that provided the least cost was identified and the ground robot was made to traverse through the goal points in the obtained order.

The problem of introducing a new obstacle all of a sudden in the robot's path (unexpected land slide in the traversing path) was also addressed. This was done by using the presence of the camera and sonar sensor in the ground robot. The camera was used to take the picture of the path in front of the ground robot at every waypoint. This picture was processed to see if the planned path ahead is free or blocked. If free, the robot moves, else the distance of the new obstacle is determined using the sonar sensor. This information is used to update the occupancy grid and a new path to the desired destination is generated. The motion planning output from point to point is shown in Fig.6.7, and the mission planning output is shown in Fig.6.8.

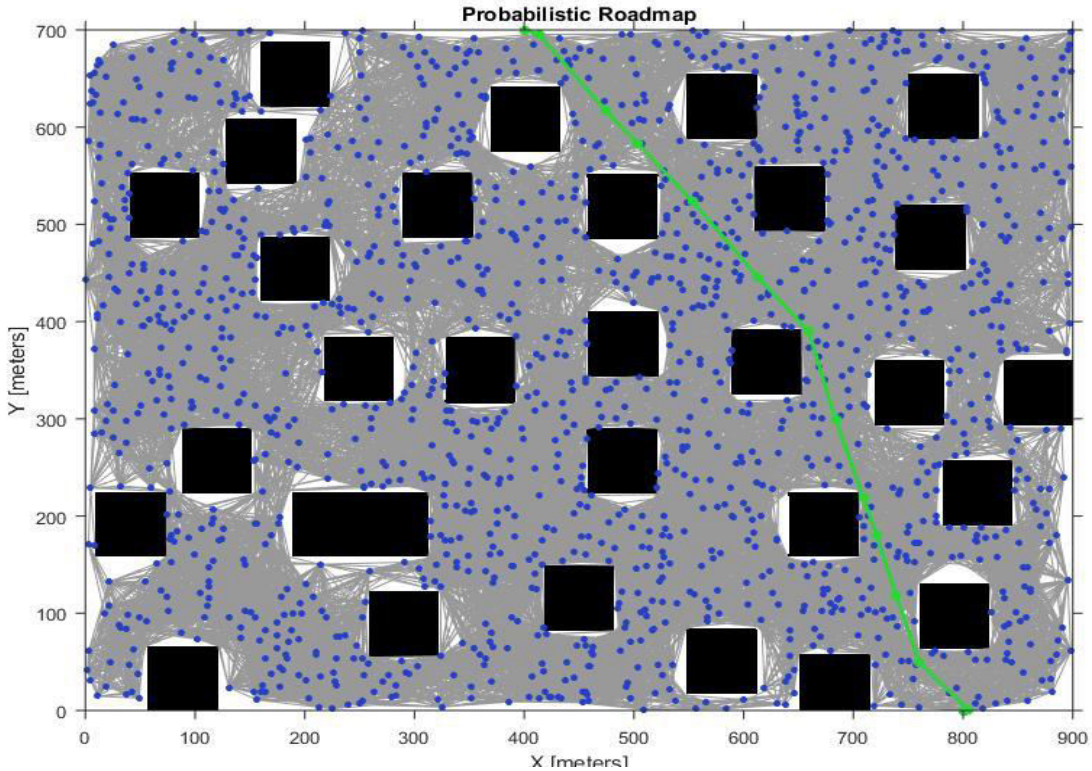
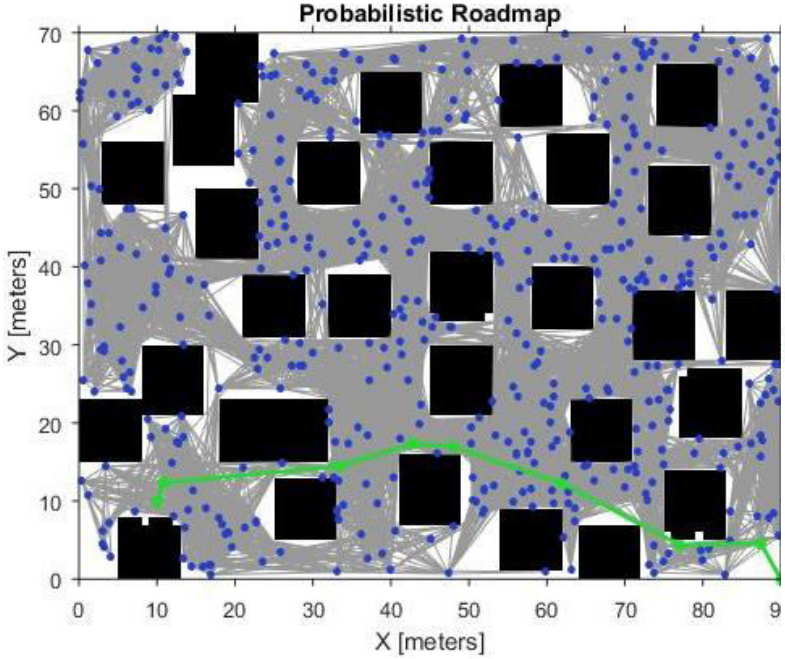
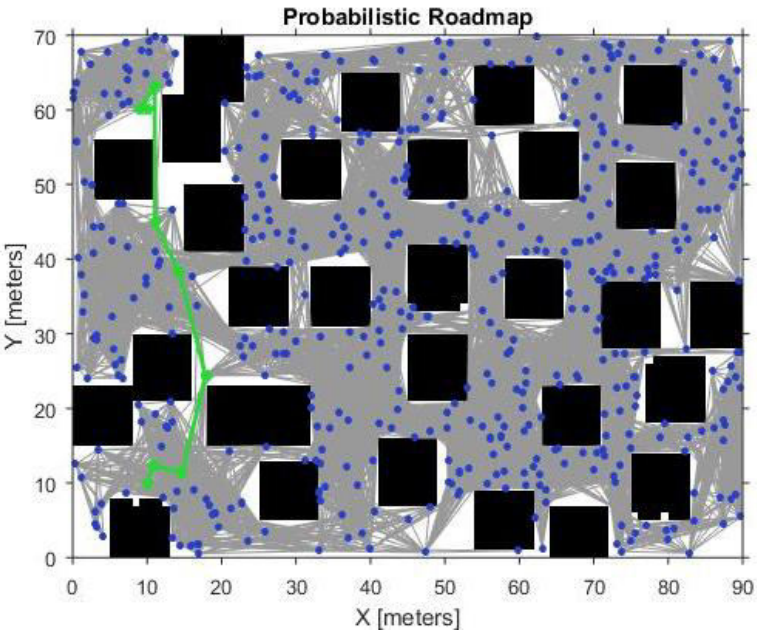


Fig 6.7 – Point to Point path planning





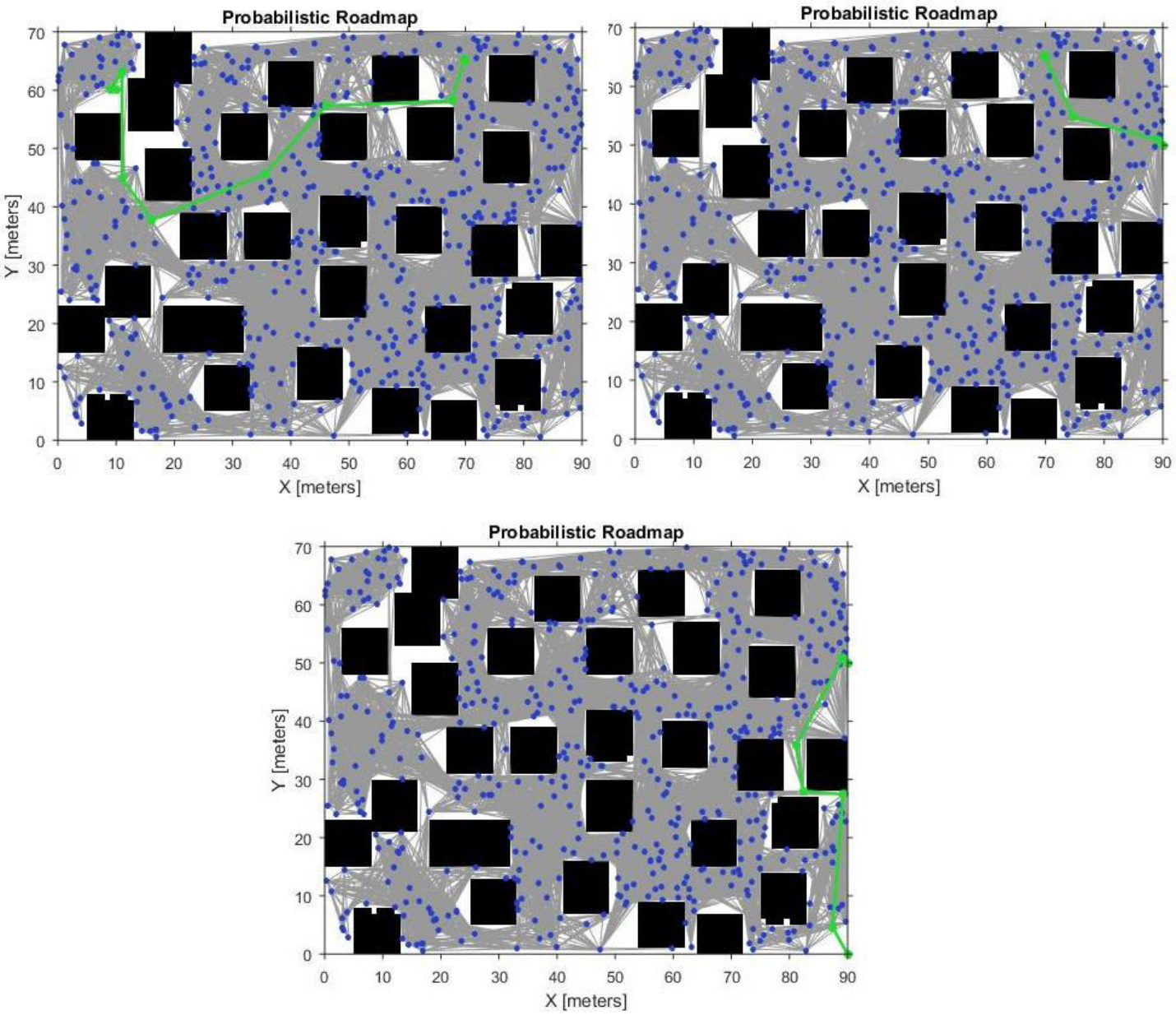


Fig 6.8 – Mission planning

6.5 SUMMARY

The real-time implementation of the aerial and ground vehicle collaboration was carried out on a mock disaster setup, where earthquake imagery was overlaid on carboard boxes and goal-points were represented with the help of density of population stranded in certain areas. The results were also presented through images

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 CONCLUSION

From the experimental results, it can be seen that the combined efforts of an Unmanned Aerial Vehicle and an Unmanned Ground Vehicle surpasses the human way of approaching the same problem. Without the collaboration, it would have been very difficult to identify places that require immediate attention. Thus the collaboration between a UAV and a UGV has worked efficiently and provided better way of approaching search and rescue missions. The provided solution can be developed in many possible ways.

7.2 FUTURE WORK

Even the DJI PHANTOM 3 faced stability issues as it was radio controlled. Replacing Radio Controlled drones with computer controlled drones will a significant improvement as it can improve the footage quality.

The ground vehicle was built on an ARDUINO platform. This led to many difficulties like absence of multithreading operations, difficulties in precise turning etc. These problems can be prevented by building a robust ground vehicle, built with micro controllers with better functionality than ARDUINO. This can also remove the necessity of a master node and provide direct communication between a UAV and a UGV

The proposed solution makes use of Scale Invariant Feature Transformation (SIFT) and performs panoramic stitching which is an offline process. By the use of visual odometry, the map can be updated online which saves a lot of time.

The data transfer modules that are used here are of low cost but, they consume a lot of time to transfer data. They can be replaced by modules that offer quicker communication as this is a problem where time plays a key role.

It is to be noted that predefined obstacles were used here. This is not the case in a real life disaster scenario. To overcome this problem, Learning techniques can be used to detect if the image contains an obstacle or not.

REFERENCES

1. Brown, M. and Lowe, D.G., 2007. Automatic panoramic image stitching using invariant features. *International journal of computer vision*, *74*(1), pp.59-73.

2. Brüggemann, B., Brunner, M. and Schulz, D., 2012. Outdoor navigation with a coordinated multi-robot system that maintains spatial constraints. *IFAC Proceedings Volumes*, *45*(28), pp.1-6.

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

4. Delmerico, J., Mueggler, E., Nitsch, J. and Scaramuzza, D., 2017. Active autonomous aerial exploration for ground robot path planning. *IEEE Robotics and Automation Letters*, *2*(2), pp.664-671.

5. Duan, H. and Li, P., 2016. *Bio-inspired computation in unmanned aerial vehicles*. Springer.

6. Garzón, M., Valente, J., Roldán, J.J., Cancar, L., Barrientos, A. and Del Cerro, J., 2016. A multirobot system for distributed area coverage and signal searching in large outdoor scenarios. *Journal of Field Robotics*, *33*(8), pp.1087-1106.

7. LaValle, S.M., 2006. *Planning algorithms*. Cambridge university press.

8. Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), pp.91-110.

9. Michael, N., Shen, S., Mohta, K., Mulgaonkar, Y., Kumar, V., Nagatani, K., Okada, Y., Kiribayashi, S., Otake, K., Yoshida, K. and Ohno, K., 2012. Collaborative mapping of an earthquake-damaged building via ground and aerial robots. *Journal of Field Robotics*, *29*(5), pp.832-841.

10. Mueggler, E., Faessler, M., Fontana, F. and Scaramuzza, D., 2014, October. Aerial-guided navigation of a ground robot among movable obstacles.

In *Safety, Security, and Rescue Robotics (SSRR), 2014 IEEE International Symposium on* (pp. 1-8). IEEE.

11. Murphy, R.R., 2014. *Disaster robotics*. MIT press.

12. Nagatani, K., Kiribayashi, S., Okada, Y., Otake, K., Yoshida, K., Tadokoro, S., Nishimura, T., Yoshida, T., Koyanagi, E., Fukushima, M. and Kawatsuma, S., 2013. Emergency response to the nuclear accident at the Fukushima Daiichi Nuclear Power Plants using mobile rescue robots. *Journal of Field Robotics*, *30*(1), pp.44-63.

13. Saska, M., Krajnik, T. and Pfeucil, L., 2012, March. Cooperative μUAV-UGV autonomous indoor surveillance. In *Systems, Signals and Devices (SSD), 2012 9th International Multi-Conference on* (pp. 1-6). IEEE.

14. Schneider, F.E., Wildermuth, D. and Wolf, H.L., 2015, July. ELROB and EURATHLON: Improving search & rescue robotics through real-world robot competitions. In *Robot Motion and Control (RoMoCo), 2015 10th International Workshop on* (pp. 118-123). IEEE.

15. Sofman, B., Lin, E., Bagnell, J.A., Cole, J., Vandapel, N. and Stentz, A., 2006. Improving robot navigation through self-supervised online learning. *Journal of Field Robotics*, *23*(11-12), pp.1059-1075.

16. Szeliski, R., 2010. *Computer vision: algorithms and applications*. Springer Science & Business Media.

17. Tokekar, P., Vander Hook, J., Mulla, D. and Isler, V., 2016. Sensor planning for a symbiotic UAV and UGV system for precision agriculture. *IEEE Transactions on Robotics*, *32*(6), pp.1498-1511.

18. Viola, P. and Jones, M., 2001. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on* (Vol. 1, pp. I-I). IEEE.

19. Yu, B., Dong, X., Shi, Z. and Zhong, Y., 2013, July. Formation control for quadrotor swarm systems: Algorithms and experiments. In *Control Conference (CCC), 2013 32nd Chinese* (pp. 7099-7104). IEEE.