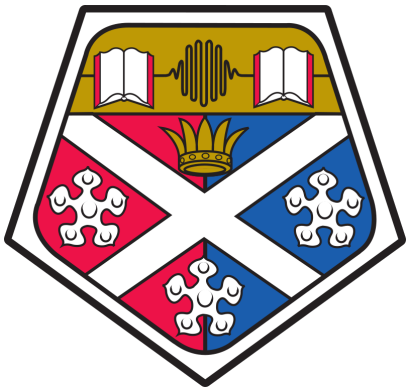
Co-operative Robotics
Using Environmental Sensors

– 19520 Interim Report –

Peter De Jonckheere, R. David Dunphy,
Andrew Fagan, Matthew Gaffney,
Kyle Miller

University of Strathclyde, Glasgow



Proposer: Dr John Levine
Supervisor: Dr Marilyn Lennon
30th November 2018

Abstract

This project aims to develop co-operative robotic systems which effectively map and search an area using non-telemetric sensors and communication. This will allow the systems to perform in environments which hinder telemetry, such as caves or tunnels. Each robot should be able to solve this problem individually; however, after the introduction of multiple robots to the area, they should be able coordinate their movements to distribute the task and solve the problem faster. The robots will be tested in a custom-made testing environment—likely taking the form of a simple maze with a target that must be located—and evaluated based on their performance both individually and co-operatively.

Detailed analysis of the electrical, software and mechanical design solutions are presented. The robots will use a differential drive system and will be designed and constructed with the necessary hardware to complete the task of exploring their maze-like environment in a distributed fashion. Computer vision systems on the robots will be used in conjunction with ultrasonic sensors to perceive the environment and identify other robots in the group, and incremental encoders and an inertial measurement unit will be used to track the robots position and orientation. Communication will be conducted over either Wi-Fi or Bluetooth systems, which will be artificially restricted to require line-of-sight, so as to simulate situations where global communication is not possible.

The report also contains extensive discussion of the project management approach used to properly coordinate the project team. The structure of the team and practices employed have been considered carefully and a detailed timeline developed to limit project risk.

Contents

1 Introduction **3**
1.1 Problem context . . . 3
1.2 Report outline . . . 4

2 Background **5**
2.1 Co-operative robotics . . . 5
2.2 SLAM . . . 5
2.3 Computer vision . . . 7

3 Design **8**
3.1 Mechanical design decisions . . . 8
3.1.1 Chassis and motors . . . 8
3.1.2 Drive system . . . 9
3.1.3 Mechanical conclusions . . . 9
3.2 Sensor selection . . . 10
3.2.1 Range sensing . . . 10
3.2.2 Inertial Measurement Unit . . . 11
3.2.3 Encoder hardware . . . 12
3.3 Software design . . . 13
3.3.1 Software engineering practices and structure . . . 13
3.3.2 SLAM and sensor fusion . . . 14
3.3.3 Computer vision . . . 15
3.3.4 Communication . . . 16
3.3.5 AI and control modules . . . 16
3.4 Software conclusions . . . 17

4 Project management **18**
4.1 Project structure . . . 18
4.2 Project workflow . . . 18
4.3 Objectives . . . 19
4.4 Risk evaluation . . . 19
4.5 Timeline . . . 19

5 Conclusion **22**

References **23**

1 Introduction

1.1 Problem context

Using multiple co-operating robots can be a useful strategy when attempting to complete tasks that can be parallelised, such as exploration of a large area. Usually, this requires a centralised system that can properly coordinate the efforts of the robots in accomplishing the task. However, in environments where radio communication is impossible or severely restricted, such as underground or in areas with high interference, the systems would be unable to complete their task. In these circumstances, communication may be limited to non-telemetric sensors that require line-of-sight. Therefore, intelligent algorithms are required to co-operate effectively in environments where line-of-sight is not always available.

Co-operative robotics can be used to solve a variety of tasks more effectively than a more complex individual robot and introduces distribution and redundancy into the system which can be highly advantageous over single agent systems [1]. This redundancy can be mission critical in scenarios where a hazardous environment poses risks to individual robots, and means that the loss of a single robot is not fatal to the mission.

Previous research into co-operative robotics has focused on UAVs [2], non-autonomous agents [3], or made use of extensive communication, such as by using the cloud [4]. This study will use restricted non-telemetric communication (i.e., local communication between neighbouring robots rather than communication over a central server). Co-operating well under this limitation poses additional challenges, which need to be overcome in order to allow collaborative robotic exploration of environments such as caves. This application area was inspired in part by a recent incident necessitating cave exploration and rescue in Thailand [5].

In order to abstract away the physical complexities of operating on difficult terrain and allow research to focus on communication and problem-solving algorithms, a toy problem has been devised for the robots to solve. This will take the form of a simple maze which contains an target that needs to be found and identified by the robots. The primary aim of the study is to construct multiple robots which can navigate this maze and co-ordinate their efforts to find the target more quickly than could be achieved by each robot individually.

An additional requirement is that the robots should be constructed using inexpensive components. This is necessitated by the increased cost implied by the need for multiple robots.

1.2 Report outline

The purpose of this report is to present the preparatory work performed to date as well as our plan for the completion of the project.

Chapter 2 gives an overview of the subject area and describes a number of concepts relevant to robotics for the uninitiated reader. A summary of the high-level design decisions taken so far and their justifications can be found in Chapter 3. Finally, Chapter 4 details our approach to the management of the project and provides details of our expected timeline and budget.

2 Background

This section will explore each of the broad topics in general detail, laying the basis for the discussion of specific choices in each of the topics in Chapter 3.

2.1 Co-operative robotics

Co-operative robotics has varied definitions across different papers. One such definition, which generalises co-operative behaviour in robotics, describes it as "joint collaborative behaviour that is directed toward some goal in which there is a common interest or reward" [6]. This description fits the objectives of this study more appropriately than the specific term swarm robotics, which has a number of additional requirements, including that problem solving should be distributed across the swarm [7]. This definition does not apply to this project, as the robots designed here are capable of operating autonomously and will be able to solve certain tasks individually. In this case, collaboration is used to deliver a performance improvement. For this reason, the more general term of co-operative robotics will be used throughout. The aim is to reduce the time taken or increase performance of the system over a single-robot system [8]. Additionally, by creating a decentralised and distributed system across a number of homogeneous agents, agent redundancy is introduced which can improve the completion rate of tasks, especially in potentially volatile environments [9, 10].

Co-operative robotics goes beyond the idea of collaborative robotics in requiring an additional aspect of intelligence in the communication and coordination of the individual agents [11].

2.2 SLAM

An essential skill for mobile robots navigating unknown environments is being able to build a map of the environment whilst simultaneously identifying its own position. This is especially true in the absence of external referencing systems such as GPS. This is known as the Simultaneous Localisation And Mapping (SLAM) problem and has been one of the most extensively researched topics in mobile robotics over the last two decades [12]. As the robot's estimate of its position is affected by both the previous state's uncertainty and the any errors in the current measurement, uncertainties in the readings compound over time. To rectify this, a map with distinctive landmarks can reduce its localisation error by revisiting these known areas. This is known as loop closure.

SLAM implementations rely on sensor fusion algorithms which take in readings from an array of sensors and calculates a estimated state changed based on the probability or error for each sensor. This effectively allows errors between multiple sensors to be cancelled out, resulting in more reliable estimates. A typical approach is to use sensor

fusion to combine odometry readings from wheel encoders with acceleration information obtained from an inertial measurement unit (IMU) to correct for errors caused by wheels slipping and sensor imperfections.

There are a large variety of solutions to suit various system requirements, which fall into the two categories of filtering and smoothing. Filtering involves creating a state estimation where the state of the system consists of the current robot position and the map. The estimate is augmented and improved by considering the new measurements as they become available. Some popular approaches to filtering are techniques such as Kalman filters, particle filters and information filters. This technique is known as online SLAM where only the most recent pose (the robot's position and orientation) is recovered with previous poses marginalised out. Smoothing approaches involve a full estimate of the trajectory of the robot from all available measurements. These typically use least-square error minimisation techniques and are used to address the problem known as the full SLAM problem. Full SLAM involves attempting to map the entire path.

The state of the system is known as x_k and this can be defined as a function which uses the previous state to determine the next state. As the model can not be perfectly accurate, there will be uncertainty in the readings and so this must be taken into account by the model. As a result, x_k, which is known as the motion model, is defined as

$$x_k = f(x_{k-1}, q_{k-1}) \,, \tag{1}$$

where q_{k-1} is the randomness introduced to the system. As such, this can also be represented by the probability distribution

$$x_k \sim p(x_k|x_{k-1}) \,. \tag{2}$$

Both of these imply that the state is stochastic and depends on the previous state. The probability distribution helps emphasise that the current state is drawn from a distribution of possible states based on the previous state. Given that a perfect sensor is not possible, the current state will also have noise in the reading. This is known as the measurement model and can be defined as

$$y_k = h(x_k, r_k) \,, \tag{3}$$

where r represents the uncertainty that is present in the model of the sensor. As before this can be expressed as an uncertainty model:

$$y_k \sim p(y_k|y_{k-1}) \,. \tag{4}$$

It is assumed that the motion and measurement models are Markovian in that the current state only depends on the previous state. The measurement model only depends on the current state and no previous values.

By applying Bayes' theorem and marginalisation the current state can be described as

$$p(x_k|y_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|y_{1:k-1})dx_{k-1} \tag{5}$$

$$p(x_k|y_{1:k}) = \frac{p(y_k|x_k)p(x_k|x_{1:k-1})}{p(y_k|y_{1:k-1})}\,. \tag{6}$$

Equation 5 is known as the predict equation. By integrating over the previous state, all potential outcomes of the state x_k are considered. Equation 6 is referred to as the update equation, as the prediction is updated using the new measurement information [13].

One of the most common methods of implementing SLAM is filtering using an Extended Kalman Filter (EKF). An EKF is an efficient, recursive filter that estimates the state of a dynamic system from a series of noisy measurements [14]. This uses the premise of the predict and update equations as joint probability distributions. Given variables defined on a probability space, the joint probability distribution gives the probability that each of the variables falls in any particular range or set. It uses these techniques to estimates a state vector containing both the location of landmarks in the map and the robot pose [15].

2.3 Computer vision

Computer vision is the analysis of digital image or video data to allow a computer system to gain a high-level understanding of the 3D environment contained within the image[16].A common application for computer vision is the identification and classification of objects. Identification generally involves the recognition of features of the object and the comparison of these features and their relative positions to a known model of the object. Classification usually involves machine learning algorithms to build up a definition of the object based on its visible properties[17].

Computer vision can also be used to triangulate the position of objects in the field of view by measuring the discrepancy in the objects position in the two camera frames given a translation matrix relating the two cameras.

Computer vision can be well integrated with SLAM providing a means of both the measure of distance (if the CV system is bi-ocular) and the identification of distinctive features in the environment to allow loop closure[18].

3 Design

3.1 Mechanical design decisions

The mechanical design of each agent is central to its functionality, with sensors being heavily reliant on the accuracy of the mechanical construction. As multiple robots are being created, the mechanical similarity is also important to ensure the sensors and software function consistently across each robot. The placement of sensors will be critical to this and so the first robot will be completed and tested before further agents are constructed. Printed Circuit Boards (PCBs) are likely to be used for this purpose in the final iteration of the design—with strip board being used for prototyping—as this will provide additional robustness and guaranteed repeatability between robots.

3.1.1 Chassis and motors

The main design decision to be made involving the chassis was whether to use a pre-built "hobbyist" chassis or create a bespoke design which could be easily replicated for each agent. The initial design plan had been to use pre-built chassis with the primary concern regarding this was that motors were often included in these pre-built packages and these motors could be insufficient for wheel odometry purposes.

Following a consultant meeting with Dr Mark Post (see Section 4.1), it was suggested that a bespoke chassis may better fit the requirements of the task. This sentiment was echoed by Dr Harle in a class meeting when it was suggested that mechanical assistance was readily available from the mechanical workshop in the EEE department. In order to do this, a CAD (Computer Aided Design) model would have to be created and taken to the mechanical workshop for a model to be either laser cut or 3D printed.

Chassis selection became integral to the flow of the project as a bespoke design would have to be created last in the design process following the detailed design of other aspects such as the sensor layout and PCB design, whereas a pre-built chassis could be used to prototype design solutions prior to a final design decision.

These options were considered against each other and it was decided that a pre-built chassis would be used due to a number of factors, namely: the lack of in-depth knowledge of CAD amongst the team; the possible cost of a bespoke chassis compared to pre-built; and the lack of control surrounding the creation of the chassis. Having weighed the pros and cons of each approach it was concluded that the cons of designing a bespoke chassis outweighed the pros and so a pre-built chassis would be used. This will be ordered in the coming weeks as per the Gantt chart (Figure 4).

Having investigated multiple possible options, two-wheeled robots were found to be the simplest and most mechanically simple to work with while providing the required precision of movement. Many of these chassis are accompanied by motors which would be

sufficient for the purposes required. One particluar example chassis [19] is accompanied by two motors with sufficient gearing (120:1) to allow for the required precision when driving the robot. This chassis also has a number of additional accompanying parts which will be useful in maintaining both mechanical stability and a high level of repeatability.

3.1.2 Drive system

The drive system for the robots will be a differential drive system (DDR). This drives each wheel independently using independent actuators and the wheels are not connected by a single axle [20, p. 146]. When using a two wheeled robot, DDR allows the robot to rotate on the spot around its central axis when the wheels are driven in opposite directions. This provides a high level of mobility which will aid in the sensing and mapping capabilities of the robot. Each wheel will require a individual motor and an encoder for wheel odometry data.

The drive system will be controlled either by the central computer on the Raspberry Pi or by a microcontroller uniquely assigned to managing the drive system. In either case the wheel odometry data will be gathered and sent for fusion with data from other sensors on the Raspberry Pi for the purpose of localisation. The benefits of using a microcontroller for the drive system would predominantly be the higher level of precision with which tasks could be timed and carried out. This would also allow the odometry data to be collected quickly and regularly with little interference from other tasks, ensuring the data was correct. The decision on this has not yet been made and this will be based predominantly on the load of the algorithms on the Raspberry Pi which is currently difficult to determine. A cheap and effective microcontroller such as the MSP430 Launchpad could be ordered and integrated at fairly short notice following consultation with Dr James Irvine (see Section 4.1).

A pre-built motor drive board is likely to be used from the same supplier as the pre-built chassis. As an accompanying component the circuit is guaranteed to fit neatly onto the chassis [21]. This circuit also includes a power distribution board which will be useful in reducing the mechanical complexity of the robot. Using these pre-built solutions will improve the repeatability of the robot as the number of parts which have to be created and fitted will be reduced.

3.1.3 Mechanical conclusions

A number of pre-built solutions have been chosen for the mechanics of the robot as the consistency between robots is vital to their coordination and although bespoke solutions may have provided a small increase in consistency, these would have required a disproportionate amount of design and implementation time in an aspect of the project which is not a novel concept. A sharp learning curve would also be involved in creating

bespoke mechanics, and while alone this is not as bad thing it would be accompanied by the sharp learning curves of SLAM and computer vision. It was therefore decided that pre-built parts should be used and time of the project diverted towards those areas instead.

The main objectives for this mechanical design are therefore as follows:

- Construct an easily replicable robot
- Use predominantly pre-built components with PCBs to maintain consistency
- Order pre-built components as per Gantt chart Figure 4

3.2 Sensor selection

3.2.1 Range sensing

Localisation requires a sensor or set of sensors capable of measuring the distance to objects in the robot's environment. The primary range sensors considered for this project are all active sensors, meaning that they measure the echo from a transmitted signal. Range-finding sensors of this sort include infrared range-finders, ultrasonic transceivers, and lidar.

Of these, lidar gives the most accurate readings, and has the advantage of covering 360°. However, the most affordable lidar systems exceed the budget of the project and are too large to be placed on the chassis. For these reasons, lidar was immediately discounted as a possibility for our application.

Infrared range-finders form an inexpensive alternative to lidar. Here, a narrow beam of infrared light is emitted, and the distance is determined by trigonometric calculations based on the angle of reflection. The main disadvantages of these sensors is a significant decrease in reliability, due partly to the possibility that the narrow light beam may not detect objects that take up a small proportion of the field of view, and partly to the risk of interference by other light sources [22].

Ultrasonic transceivers are another inexpensive option. This sensor consists of a transceiver which transmits and receives ultrasound pulses of a specific frequency. By measuring the time delay between transmitting and receiving, the distance to an object which is reflecting the signal can be calculated. At a low price range, ultrasonic sensors result in less accurate readings than infrared sensors; however, the wider effectual angle of these sensors combined with the advantage that they are not affected by ambient light or the reflectance or gloss of materials make them significantly more reliable in many circumstances.

Ultrasonic sensors have a minimum range, below which they are prone to incorrect measurements caused by registering subsequent echoes. This "deadband" distance is

typically in the range of less than 3 cm. In order to avoid the possibility of objects appearing within this range, the sensors need to be set back from the edge of the robot, so that objects directly adjacent to the robot fall within the material window of the sensor.

The maximum range of ultrasonic sensors varies between sensor models, and further depends on the size and material of the object being detected. A practical sensor range of around 1 m is typical for low-budget sensors. As the robots will be operating in a controlled environment for the purposes of this project, the maximum distance of an object within the environment can be controlled.

Ultrasonic sensors can only detect objects that fall within the conical beam emitted by the transmitter. The measuring angle of this beam is typically around 15°. In order to identify objects in a wider field of view, multiple sensors need to be arranged around the robot. Our intention is to use between three and five sensors on each robot, spaced at between 20° and 30° to create a field of view of around 90° to 120°, facing towards the front of the robot as shown in Figure 1.

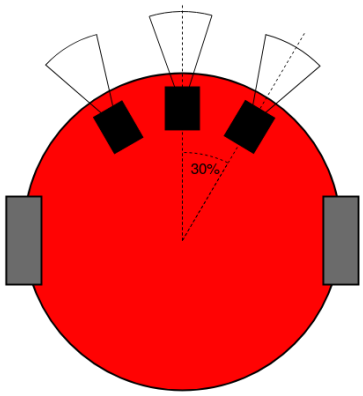


Figure 1: Ultrasound Sensor Layout

One drawback of using active range-finders is that there is the possibility that sensors will interfere with each other. This problem is normally solved by synchronising the sensors so that they fire alternately, ignoring any signals received outwith a specified time period. Synchronising ultrasonic sensors on a single robot is a trivial task; however, synchronising between robots will require some coordination which will be discussed further in Section 3.3.4.

3.2.2 Inertial Measurement Unit

A 6 degree of freedom (dof) IMU measures both its acceleration in 3 axes and its angular velocity in about these axes. This allows the robot to keep track of its position and orientation relative to where it started over time.

The IMU currently being used is the MPU-6050. At its smallest range of accuracy ($\pm 250°/s$ and $\pm 2g$[23], which will be adequate for our application) it was the most accurate sensor available at a similar budget in terms of non-linearity, cross-axis sensitivity and noise. Offset tolerance was not a major consideration as calibration can compensate for this, as long as it is not so significant that it can reduce the range. In terms of power consumption, while it was not the most efficient, it was reasonable, and power is not a major constraint of the project. Additionally, it performs some on board computation with the results, easing the load in interpreting the raw data for the control system.

A major problem with using the IMU is that as it has to integrate the rotational velocity and double integrate the acceleration to track the position, small errors or offsets can accumulate over time and cause a large "drift" in the estimated position.

3.2.3 Encoder hardware

An encoder is a feedback device which can be connected to motors to provide sensing data, such as the wheel speed and position, which can aid in localisation [20, p. 60]. There are two main types of encoder, namely: linear, which senses motion along a fixed path, and rotary, which senses rotational motion [20, p. 60]. As the robot will use wheels to move, a rotational encoder for each wheel will be used. There are a further two sub-types of rotational encoder: incremental rotary encoders and absolute rotary encoders. The incremental encoders considered use light and a glass disk to generate a series of pulses. The position of the incremental encoder is determined using the pulses produced. They are not as accurate as the other form of rotary encoder: absolute encoders. Absolute encoders also use light to detect movement, however these use a stationary mask giving each position its own unique set of bits.

A number of factors influence the selection of encoders including the output type, the desired resolution, and the size of the encoder and casing. Since a pre-built chassis will be used, the size of the encoder is the most high priority requirement as it must fit onto the body of the robot. The desired resolution is also important due to the lack of accurate sensors for mapping (see Section 3.2), wheel odometry will be of greater importance for localisation, hence it is important that this is accurate. With a few exceptions, absolute encoders are significantly more expensive than incremental encoders and so it is likely that incremental encoders will be used for this project.

If using the expected pre-built chassis [19], this also comes with accompanying encoders [24]. These will be used as they are made specifically for the pre-built chassis and operate in the correct revolutions per minute (rpm) to match the motors provided with the chassis and a high number of counts per revolution providing high levels of accuracy.

3.3 Software design

3.3.1 Software engineering practices and structure

In considering frameworks and tools to apply to the project, there are several key decisions to be made, crucially in software architecture style and language choice.

In the high level structure of the software, several architectures might be employed. The highly modular nature of the system, with many interacting components each having very specific functions, lends itself especially well to an Object Oriented or Implicit Invocation architecture. The object oriented paradigm is well known, abstracting the system into smaller functional groups called objects which invoke each other's methods. Perhaps less well known, the event driven implicit invocation paradigm is very similar to object oriented design, but with the crucial difference that objects do not interact by directly calling each other, but rather by observing each other for events that require a response. This change is well suited to this project, as many of the modules will be constantly outputting data (such as the sensors) or else performing analysis of new data (such as SLAM) and all performing in parallel.

Another important consideration is which programming languages to utilise for the project. The main possibilities are Python, which is well supported by the Raspberry Pi and easy to use, allowing for rapid prototyping and development, and C/C++, which is more difficult to develop in but leads to more efficient programs.

A solution to both of these dilemmas is presented by using the Robot Operating System (ROS) framework. ROS is an open-source programming framework for complex modular robotic networks. It is not an operating system as such, despite the name, but rather a collection of libraries and tools to make it easier to develop modules within a robot and allow them to communicate.

The framework treats the robot as a network of nodes (individual programs with one or few purposes). Nodes communicate via a modified version of the Observer design pattern known as the Publish/Subscribe model. Data is attached to a *"Topic"*, which is published by a node. Any number of nodes can subscribe to a topic, and when data is published on that topic an interrupt function is triggered in the Subscriber node, allowing access to the data.

ROS creates very loose coupling between nodes, which promotes good software engineering practices, and will allow the system to be easily modified and expanded. This also makes testing far easier, as modules can be unit tested on their own and integrated together after the fact.

All this is to say that by utilising the framework and implementing modules as ROS nodes, the high level architecture of the system becomes very much akin to implicit invocation. Moreover, ROS supports nodes within the same system being programmed in

several languages, including Python, C++ and Java, meaning that this decision can be made on a module by module basis.

With this in mind, the high level architecture of the system is shown in figure 2, with each block in the diagram representing a ROS node, and each arrow corresponding to communication of a topic between those nodes.

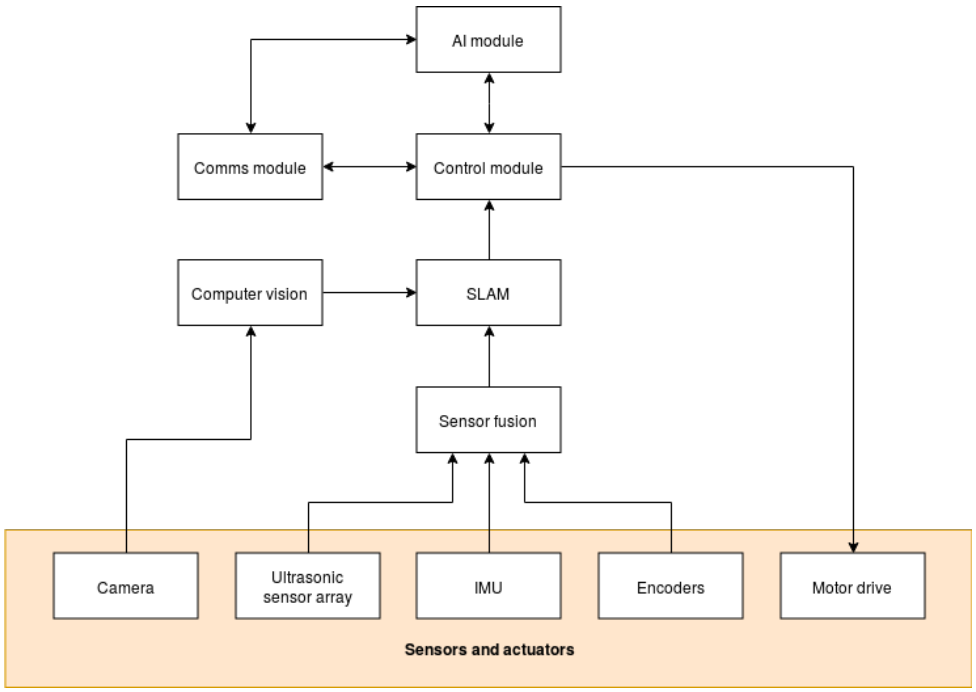


Figure 2: High-level module diagram of software components

3.3.2 SLAM and sensor fusion

Implementing SLAM is one of the key challenges of the project. The two main options are either to create an implementation from scratch or to use an existing library or framework to aid in providing this functionality for the robots. An example of such a library is PythonRobotics, which contains a collection of robotics algorithms, especially with the focus for use in autonomous navigation. There are also libraries for use with ROS that could be used to implement the SLAM such as "gmapping".

FastSLAM is a commonly used implementation, a factored solution first proposed in [25]. This involves recursively estimating the full posterior distribution over the robot's pose and landmark locations, but scales logarithmically over the number of landmarks in the map. This, therefore, is very scalable and thus used for areas with a greater number of landmarks—it was tested up to 50,000 landmarks initially. An improvement to the algorithm was released as FastSLAM 2.0 [26]. An issue found in the original FastSLAM algorithm was that the performance degraded if the sensors were too accurate and the motion uncertainty was too great relative to the observations. FastSLAM 2.0 utilises the current observation when creating the proposal distribution to create better matches and

improve its accuracy. As a result, the original algorithm is worse in nearly all aspects with the main disadvantage being that FastSLAM 2.0 is more difficult to implement.

The alternative implementation approach considered was to write an implementation from scratch without the use of any frameworks. This would have the advantage that the solution would be fully customised to the needs of the project. Alternatively, however, SLAM in itself is a difficult concept and so it may not be feasible in the time to implement it successfully.

The implementation of SLAM will most likely involve using FastSLAM, or an equivalent implementation algorithm. This is because, even when using such a library, this is already a very challenging problem. Implementing SLAM from scratch would likely prove very difficult to manage in the time constraints of the project, especially when it is only part of the full solution. Both the input to and the output from the framework will need altered and analysed before the robot can interpret where they are and create a map which will likely be difficult.

3.3.3 Computer vision

Solutions both using computer vision and relying on other sensor types were discussed. While it can be computationally intensive, especially for the microcomputers that we intend to use, computer vision can provide the system with a wealth of useful information, is more flexible than most other practical sensors, and provides unique solutions to several problems encountered in this project.

Firstly, a monocular, feature-based object identification system could be used to implement loop closure when mapping the environment. This can drastically reduce error, especially over larger areas, where drift is most prolific.

This could also be used for the identification of other robots. This then allows the system to not map the robots in the environment and identify what other robots are doing and coordinate itself without direct communication.

If the CV system was bi-ocular, it could also be used for measuring distance, and could perhaps replace or improve the distance sensing system.

It was decided that the monocular solution was to be implemented as it would allow loop closure and robot identification, which would otherwise be very difficult. The two-camera solution would be far more computationally expensive than using ultrasound distance sensors.

There are several distinct types of feature detection that can be used for SLAM, each with their own key point detection and feature description algorithms. In a quantitative comparison of several different algorithms (SIFT, SURF, KAZE, AKAZE, ORB, and BRISK), it was found that ORB is the most efficient, with BRISK slightly slower, which also proves to be the most accurate [27] (with the exception of SIFT, however this is not appropriate given the licensing requirements). Therefore either ORB or

BRISK will be used going forward. It is currently expected that ORB will be used, as BRISK's performance benefits are only significant when there are large changes in scale or rotation [27], which is not expected given our application.

It should be noted that while an ideal solution would implement a system as described, due to time constraints and the focus of the project on the co-operative performance, a simpler method of simulating this behaviour may be used, such as identifying robots by colour.

Research was also conducted into the hardware required for the computer vision. As the processing will be done on a Raspberry Pi (RPi) zero, it was decided to use the standard RPi camera. This allows the use of the RPi's dedicated Camera Serial Interface (CSI) port instead of the USB ports that would be required by most alternatives. This reduces the computation required by both the camera and, more importantly, the RPi.

3.3.4 Communication

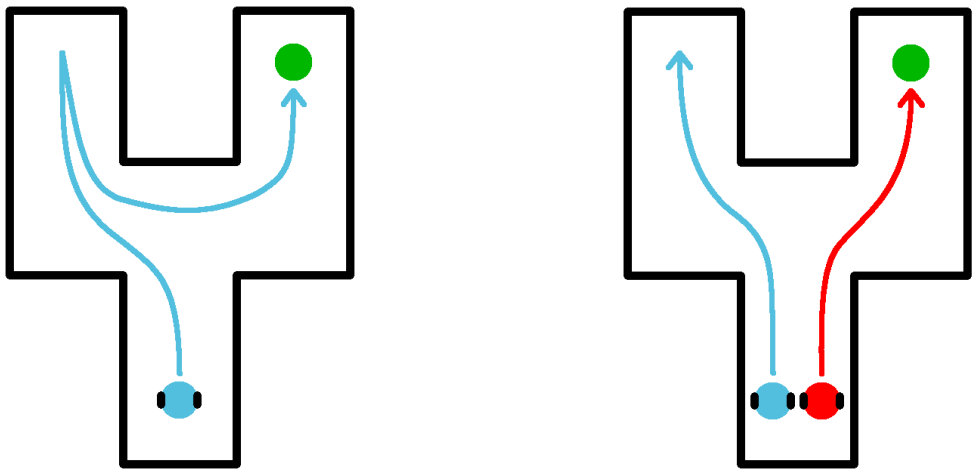
The requirement that robots can only communicate when they have line-of-sight poses severe restrictions on the types of communication that can be performed. While the robots will be operating in a small enough area that it will always be possible for them to communicate with all other robots, we will be artificially restricting their ability to communicate in order to simulate an environment in which physical limitations apply.

Communication may be performed using either Wi-Fi or Bluetooth. At present, the most promising solution appears to be a wireless ad-hoc network (WANET), which allows the robots to communicate directly with each other using Wi-Fi without the need to connect to a router. This has the added advantage that all nodes on the network are equal. For debugging and monitoring purposes, the robots may also be communicate with a computer on the same network.

3.3.5 AI and control modules

A central control module will interact with the low-level systems and coordinate the robots' sensors and actuators. High-level decisions about how to navigate the environment will be made by a separate AI module, which will interact with the communications system to coordinate the search of the map with other robots.

Basic algorithms for exploring a physical maze-like environment include depth-first search, where the robot continually takes arbitrary paths until it reaches a dead end or encounters a previously explored position again, and then doubles back to the last fork in the maze and takes the next unexplored path, as shown in Figure 3a. This algorithm will always terminate on a finite maze.



(a) One robot solving the maze using depth-first search

(b) Two robots co-operatively solving the same maze

Figure 3: Example of robots solving a simple maze

Given multiple robots, this algorithm can be extended to search a space more efficiently. By travelling together until they encounter a fork and then splitting up, they can potentially explore the environment significantly quicker (Figure 3b).

To allow the system to be modified and extended, the control module should be implemented as a rich API, allowing the AI to be easily rewritten by abstracting the robot to a single external interface.

3.4 Software conclusions

Utilising ROS as the underlying framework, and programming in any language deemed reasonable for the specific module, several key modules must be implemented. A computer vision module must be created, and integrated with a module implementing SLAM for the robot. Moreover, a communication system will be implemented to allow the robots to interact with each other. Finally, a central control module must be devised, as well as an AI module which can interact with the system as an API.

4 Project management

4.1 Project structure

At a high level, the project was structured as a team within a hypothetical engineering consultancy being commissioned to undertake the project. This creates several key roles both within and outside of the team, including a sponsor, a supervisor, a project manager, and one or more consultants. The sponsor of the project, Dr John Levine, provided the topic and the initial aims of the project, but has no formal role in the project as it develops, except for clarification of requirements and recommendations for design decisions.

The team supervisor, Dr Marilyn Lennon, serves as an advisor and manager of the team, providing guidance in project management and general engineering guidance, though without in-depth technical knowledge of the subject matter. This role is very useful, as it enforces good engineering practices and project structure, while preventing the project management from being inhibited by details overly specific to the project.

Since the supervisor does not have all the technical knowledge the team requires, other academics will take on the role of consultants, allowing the team access to in-depth specialist knowledge. Examples of consultants who have already contributed advice to the project are Dr Mark Post and Dr James Irvine, who have supplied insight into practical robotics and microcontrollers, respectively. This also has the benefit that, while the team retains access to the expertise of the consultants, they are not formally part of the project, so the team still has the final judgement in design decisions.

Within the team, a project manager was appointed. The manager is not a team leader, and has no increased authority over design decisions, but assigns tasks and enforces deadlines on the whole team. This attempts to ensure each team member is able to contribute to decisions made by the group, and that all options are given equal consideration. It also imposes order on the structure of the team while minimising needless bureaucracy.

4.2 Project workflow

All files relevant to the project are stored on a GitHub repository for version control and access by all team members. The master branch cannot be modified except by branch merging, so team members must work on their own branches, and another team member must approve changes to allow them to be finalised.

The repository is integrated with Slack, a tool for team communication. Discussions are either held on Slack, or minutes are uploaded to Slack. Tasks are created during meetings and posted as issues on GitHub immediately. Each task is attributed to team members on a voluntary basis, and failing that by assignment by the project manager.

4.3 Objectives

The specific objectives of this project were as follows:

- Design a simple differential drive robot capable of exploring and perceiving its environment—Major Objective (Source: Section 3.1.3)
- Construct two robots using this design—Major Objective (Source: 1.1)
- Develop SLAM algorithm to allow the robots to explore an area—Major Objective (Source: Section 3.4)
- Develop a system to allow the robots to interact and communicate with each other—Major Objective (Source: Section 3.4)
- Develop algorithms to search a maze which can be dynamically parallelised over any number of agents—Major Objective (Source: Section 3.4)
- Develop a test environment and evaluate the robots' performance—Major Objective (Source: Section 1.1)
- Add additional robot(s) and evaluate scalability of approach—Optional Objective(Source: Section 1.1)
- Improve SLAM by adding loop closure between robots—Optional Objective (Source: Section 2.2)

4.4 Risk evaluation

The main technical risks of the project initially revolve around possible lead times for the ordering of parts. This will be mitigated by ordering parts as soon as feasibly possible and also by carrying out other work such as research and further design planning in the interim. Following on from this, components will also form the main aspect of technical risk in the latter stages of the project such as if a part is damaged or stops working. In order to mitigate this risk, time is left at the end of the project where no parts are being worked on directly and so it is unlikely for components to be broken. Time for testing, evaluation and report writing can also be used to order new components in serious cases. For components with extensive lead times in the first instance, spares may be ordered—if not too expensive—to further eliminate this risk.

4.5 Timeline

The Gantt chart here Figure 4 was created using the objectives determined in each section of the design (Section 3). The timescales were determined approximately using knowledge

obtained from research with the addition of time for possible technical risks. These were considered carefully and an agile method was adopted in order to allow the tasks to be completed and give a small margin of error within the Gantt chart. For example, assembly of remaining robots is allocated a longer time than expected as this will involved lead times for components and also spans the period of time when the labs will be closed and construction work may not be possible.

Figure 4 shows a Gantt chart of the project timeline.

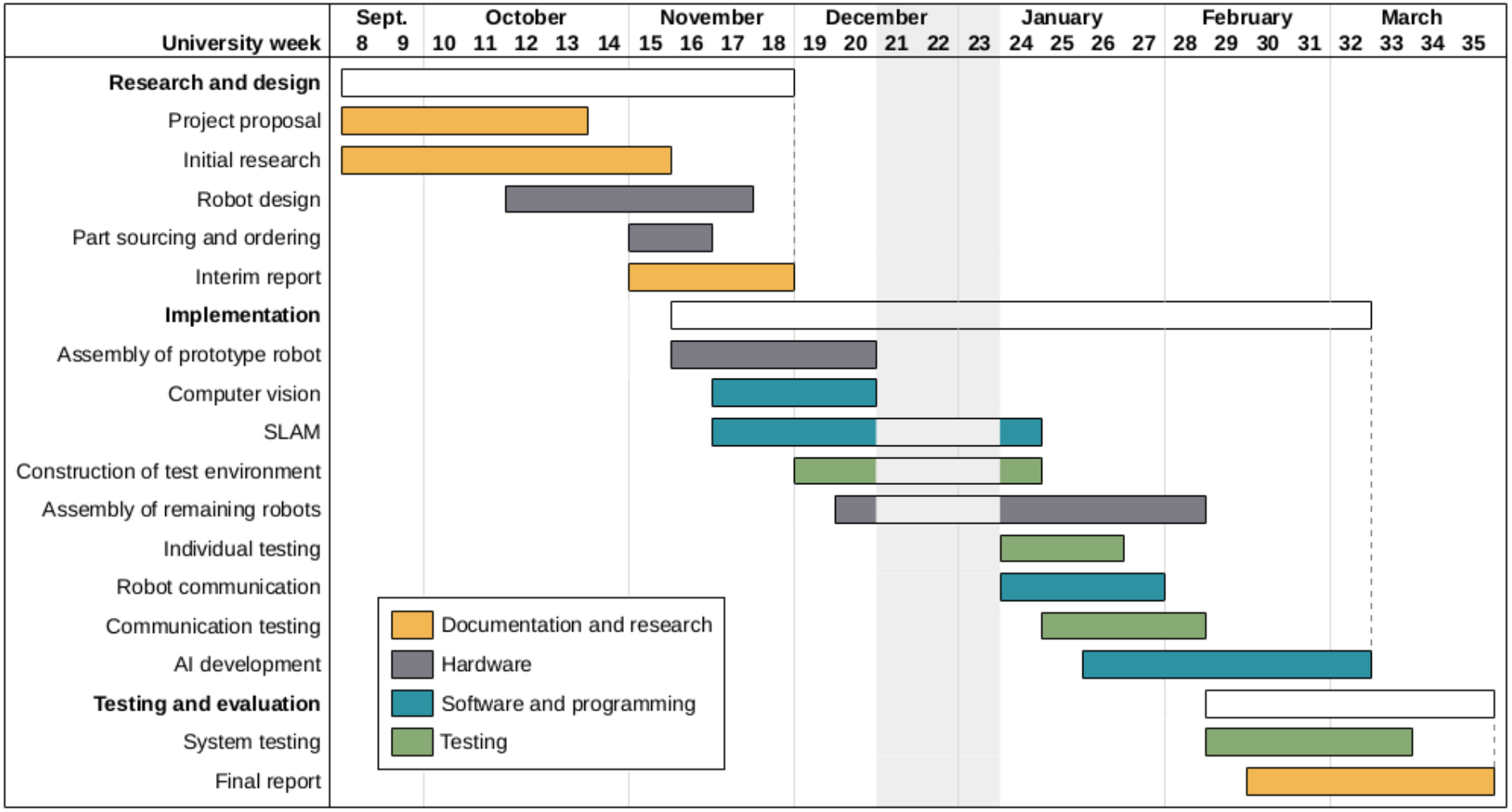


Figure 4: Gantt chart

5 Conclusion

The aim of this project is to develop a co-operative robotic system capable of searching an environment in a coordinated and distributed manner, using exclusively non-telemetric communication.

The robots will use a differential drive system mounted on a pre-built chassis. This will both minimise the time required to build the robot and provide more consistent frames than could be achieved using bespoke solutions at a similar budget. This also makes it far more straightforward to mount the motors and encoders as they are designed specifically for the chassis chosen.

Several ultrasonic range finders will be positioned along the front of the robot to take distance measurements to the surrounding walls and obstacles in the environment. Over time these measurements will be used to develop a map of the surrounding area. This will be augmented using computer vision system capable of object identification, which will identify moving objects (that should not be mapped) such as other robots, and allow loop closure though recognising distinctive areas in the environment. The computer vision will also reduce the communication required, as robots will be able to observe other robots without direct communication. The object identification will likely be feature-based, probably either ORB or BRISK due to their computational efficiency relative to other feature-based algorithms. The The robots' positions and orientations will be tracked using wheel odometry and 6 DOF IMU data which will be combined using a sensor fusion algorithm.

The team has appointed a project manager responsible for ensuring project deliverables, decided upon several workflow practices, and developed a detailed project timeline to ensure there is minimal project risk.

References

[1] G. Dudek, M. R. M. Jenkin, E. Milios, and D. Wilkes, "A taxonomy for multi-agent robotics," *Autonomous Robots*, vol. 3, pp. 375–397, dec 1996.

[2] A. Khan, B. Rinner, and A. Cavallaro, "Cooperative robots to observe moving targets: Review," *IEEE Transactions on Cybernetics*, vol. 48, pp. 187–198, jan 2018.

[3] J. Pliego-Jimenez and M. Arteaga-Perez, "Telemanipulation of cooperative robots: A case of study," *International Journal of Control*, vol. 91, no. 6, pp. 1284–1299, 2018.

[4] P. M. Wensing and J.-J. Slotine, "Cooperative adaptive control for cloud-based robotics," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6401–6408, IEEE, 2018.

[5] BBC, "The full story of thailand's extraordinary cave rescue - bbc news," November 2018.

[6] D. Barnes and D. Eustace, "A behaviour synthesis architecture for mobile robot control," in *Autonomous Guided Vehicles, IEE Colloquium on*, pp. 3–1, IET, 1991.

[7] E. Sahin, "Swarm robotics: From sources of inspiration to domains of application," in *Swarm Robotics*, pp. 10–20, 2004.

[8] S. Premvuti and S. Yuta, "Consideration on the cooperation of multiple autonomous mobile robots," in *Intelligent Robots and Systems' 90.'Towards a New Frontier of Applications', Proceedings. IROS'90. IEEE International Workshop on*, pp. 59–63, IEEE, 1990.

[9] R. Beckers, O. Holland, and J.-L. Deneubourg, "From local actions to global tasks: Stigmergy and collective robotics," 1994.

[10] L. E. Parker, "The effect of action recognition and robot awareness in cooperative robotic teams," in *IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, vol. 3, pp. 212–219, 1995.

[11] Y. U. Cao, A. S. Fukunaga, A. B. Kahng, and F. Meng, "Cooperative mobile robotics: Antecedents and directions," in *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, vol. 1, pp. 226–234, IEEE, 1995.

[12] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[13] M. Kam, X. Zhu, and P. Kalata, "Sensor fusion for mobile robot navigation," *Proceedings of the IEEE*, vol. 85, no. 1, pp. 108–119, 1997.

[14] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello, "Bayesian filtering for location estimation," *IEEE pervasive computing*, no. 3, pp. 24–33, 2003.

[15] S. Huang and G. Dissanayake, "Convergence and consistency analysis for extended kalman filter based slam," *IEEE Transactions on robotics*, vol. 23, no. 5, pp. 1036–1049, 2007.

[16] D. H. B. C. M. Brown, *Computer Vision*. Prentice Hall, 1982.

[17] M. Paoletti, J. Haut, J. Plaza, and A. Plaza, "A new deep convolutional neural network for fast hyperspectral image classification," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 145, pp. 120–147, 2018.

[18] K. L. Ho and P. Newman, "Loop closure detection in slam by combining visual and spatial appearance," *Robotics and Autonomous Systems*, vol. 54, no. 9, pp. 740–749, 2006.

[19] Pololu, "Pololu - romi chassis kits," November 2018.

[20] T. Bräunl, *Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems*. Springer Science & Business Media, 2013.

[21] Pololu, "Pololu - motor driver and power distribution board for romi chassis," November 2018.

[22] M. Hebert, "Active and passive range sensing for robotics," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 1, pp. 102–110, 02 2000.

[23] InvenSense, *MPU-6000 and MPU-6050 Product Specification*, 06 2013. Rev. 3.4.

[24] Pololu, "Pololu - romi encoder pair kit, 12 cpr, 3.5-18v," November 2018.

[25] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *et al.*, "Fastslam: A factored solution to the simultaneous localization and mapping problem," *Aaai/iaai*, vol. 593598, 2002.

[26] M. Montemerlo and S. Thrun, "Fastslam 2.0," *FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics*, pp. 63–90, 2007.

[27] S. A. K. Tareen and Z. Saleem, "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK," in *Computing, Mathematics and Engineering Technologies (iCoMET), 2018 International Conference on*, pp. 1–10, IEEE, 2018.