ENG405 (Integrated Design Project): Critical Analysis

Tatyana Maltseva (**s257346**), Sakon Nadthayai (**s245002**),
and Shane Reynolds (**s262538**)

April 9, 2018

Contents

1 Introduction & Scope

The purpose of this report is to provide a critical analysis of proposed designs discussed in the statement of work. Three main designs were proposed in the statement of work: low complexity, moderate complexity, and high complexity. In order to assess each of these schemes critically, general autonomous wheeled robot (AWR) subsystem design must be discussed. There are many subsystems to consider for AWR design, however, in the interests of brevity this report will restrict focus to six main subsystems which AWR's principally rely - these include:

1. **Mobility Configuration:** the configuration of the robot's type of wheels and their placement on the chassis;
2. **Sensory System:** used to gather odometry data, and data about the environment - consists of two types of sensors, Proprioceptive sensors and Exteroceptive sensors;
3. **Locomotion Control:** controlling the robot's motors in order to execute an identified path through the environment;
4. **Map Building:** interpreting signals from sensors to obtain an understanding of the robot's environment and workspace;
5. **Localisation:** using sensor data to determine the position of the robot in some mapped environment;
6. **Exploration Strategy:** an autonomous navigation strategy which determines where the robot should move to gain new information - this includes path planning and obstacle avoidance;
7. **Hardware Environment:** the computational hardware and the power system employed to run devices

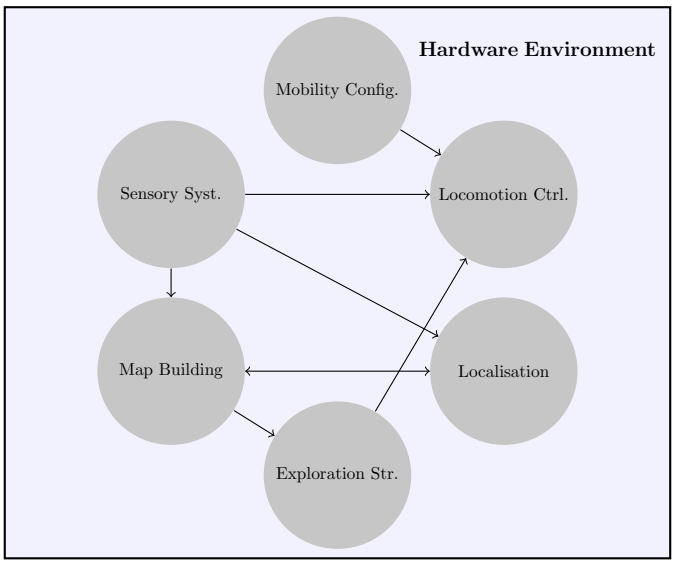

Figure 1: The interdependencies of each of the six main subsystems, encapsulated by the hardware environment.

Each of the subsystems outlined above do not operate in isolation, rather, they have interdependencies. This means design considerations made in one subsystem have an impact on design considerations in other subsystems. Moreover, each of these subsystems operates in a hardware environment consisting of the robotic chassis, the computational hardware, and the power system. The set of interdependencies can be seen in Figure 1 - the hardware environment is pictured as the rectangle encapsulating .

A critical analysis consists of three main components: detailing how each of the competing choices for a design element work; analysing the advantages and disadvantages of each; and evaluating the best choice for the design

task. This report is structured with six sections of the report which discuss competing design choices for each of the six subsystems outlined above. Each of these sections will culminate by looking at the advantages and disadvantages of each design choice. The next section of the report will briefly outline the design choices for the low complexity, moderate complexity, and high complexity designs. The final section of the report will evaluate each of the designs and provide a final recommendation on design choice.

2 Mobility Configuration

Mobility configuration refers to the type of wheel used, and their geometrical arrangement on the chassis. This will affect manoeuvrability of the robot, and kinematic complexity governing the robot's movement. Typically, the more complex the kinematics, the more difficult it is to control the robot (Siegwart & Nourbakhsh, 2004). The Pololu Dagu Rover 5 is the robot chassis specified by the client to be included in the final design, which is shown in Figure 2. The chassis has two principal configurations which are readily available:

- **Four motors, with Swedish 45 wheels:** each motor drives one of the 4 wheels directly (Swedish 45 wheels can be seen in Figure 3);
- **Two motors, with tracked wheels:** each motor drives one of the two wheels directly, and the remaining two wheels are each driven by one of the robot tracks (see Figure 2)

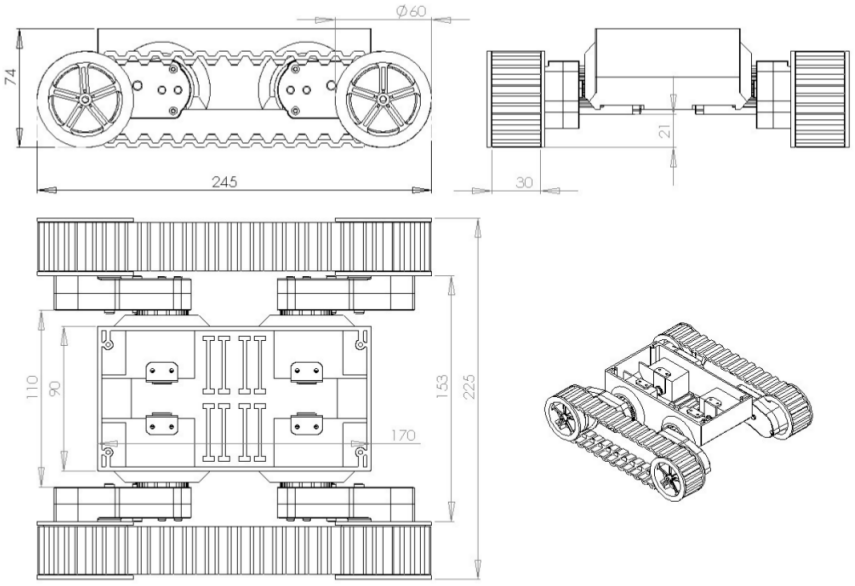

Figure 2: Detailed schematic of the Pololu Dagu Rover 5 chassis, set up with two motors and tracked wheels.

The design which uses four Swedish 45 wheels, shown in Figure 5, provides a high level of manoeuvrability. The design facilitates movement in any trajectory in the plane, however, this comes with increased kinematic complexity, making it more difficult to control. Tracked wheel design provides excellent robustness when dealing with significant floor discontinuity found in unstructured terrain. The design is a special implementation of a differential drive referred to as skid steering. The configuration intentionally relies on wheel slippage in order to make turns. This is problematic when relying on navigation schemes that utilise odometry for navigation (Borenstein, Everett, and Feng, 1996). A third configuration, which would require some chassis modification, can be seen in Figure 6. The configuration employs two motors, however, the tracks are removed from the chassis. Further, a passive spherical wheel (see Figure 4) would be mounted to the opposite end of the chassis. This configuration is widely used in AWR design, meaning that governing kinematics are simple and well documented, allowing for less complex control schemes. The main drawback to this design is no ready way to attach the spherical wheel to the chassis - a mount would have to be designed to allow for this. Table 1 provides a summary of the advantages and disadvantages for each of the three considered designs, and overall design evaluation can be found in Section 11.

Figure 3: Swedish 45 wheel, which has passive rollers around the wheel circumference, allowing movement along many different trajectories.

Figure 4: Passive spherical wheel, providing omnidirectional movement.

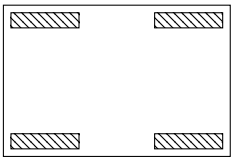

Figure 5: The geometrical arrangement of the four omnidirectional wheels, each driven by a separate motor, allows the robot to move any trajectory in the plane.

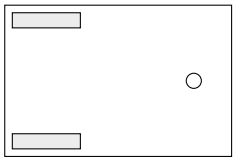

Figure 6: The geometrical arrangement shows two standard wheels, each driven by a separate motor, in a differential drive configuration. A passive omnidirectional spherical wheel mounted at the opposite end of the chassis.

Table 1: Summary of the advantages and disadvantages of the design considerations for the mobility configuration

	Four motors, with Swedish 45 wheels	**Two motors, with tracked wheels**	**Differential Drive, with passive omnidirectinal wheel**
Advantages	• The design has increased manoeuvrability • Inverse kinematics are similar to multi-dof robotic arm, since there are no non-holonomic constraints	• The design is already implemented on the Rover 5 chassis • The dynamics are based on a simple differential drive model so is easy to understand	• The design is one of the most widely used and system dynamics are well documented
Disadvantages	• The design requires more power to run the additional two motors • Mathematical description of system dynamics is complicated and would be harder to understand • Additional motors and enconders would need to be installed	• The design is based on a slippage model which is difficult to control • Odometry errors from slippage require sophisticated localisation techniques like SLAM for correction • Non-holonomic constraint means difficult to implement high level control	• Passive omnidirectional wheel has no readily available mounting • Non-holonomic constraint means difficult to implement high level control

3 Sensory System

The sensory system of a robot provides facility to capture data about the robot itself, and the robot's environment. This data is used to build maps, determine the robot's position (relative localisation), or control locomotion along some planned path - tasks which are of interest in the design of an AWR. Siegwart & Nourbakhsh (2004) classify sensors into two types: Proprioceptive, or Exteroceptive. Both sensor types will be important for design considerations.

3.1 Proprioceptive Sensors

Proprioceptive sensors are those sensors which provide internal information about the robot. Examples of this type of information are robot velocity, acceleration, wheel rotation, or the robot's heading (Leung, 2003). These types of sensors are typically used to estimate the change in robot position over time, sometimes referred to as relative localisation (Ippoliti, Jetto, & Longhi, 2005). There are two widely accepted ways of approaching this:

1. **Odometry:** uses encoders to measure wheel rotation, which is in turn used to evaluate relative position, heading, and vehicle velocity (Everett, 1995);
2. **Inertial Navigation:** uses accelerometers, together with gyroscopes, to yield position, velocity, acceleration, and heading (Borenstein, Everett, & Feng, 1995)

These types of sensors are also important for low level locomotion control. This subsection of the report will consider two types of Proprioceptive Sensors which could be used: optical encoders, and inertial measurement units.

3.1.1 Incremental Optical Rotary Encoders

Incremental optical rotary encoders are devices which will measure the angular velocity and infer relative position of some shaft, or one of the gears in a gear train. A black and white patterned disk, like the one shown in Figure 7, is attached to the shaft of interest. More segments implies higher encoder resolution. Infrared (IR) light is emitted onto the disk, and the reflected IR is detected with a sensor. IR light is reflected from lighter panels, and absorbed by darker panels. When the patterned wheel spins with the shaft of interest, the IR detector receives alternating high and low signals (Borenstein, Everett, and Feng, 1996).

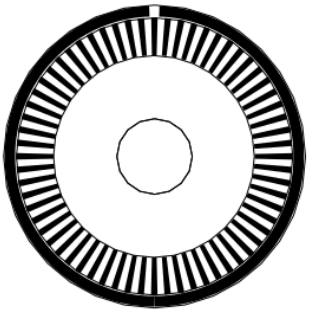

Figure 7: Black and white patterned disk on an optical encoder. White sections reflect IR light, and black absorb IR light.

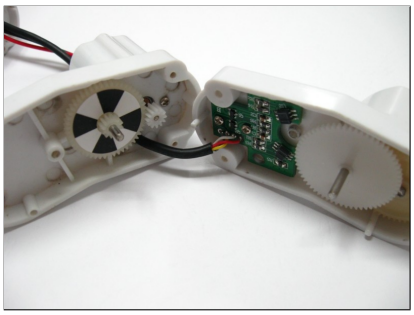

Figure 8: An incremental quadrature optical rotary encoder is provided on each of the two motors that come with the Rover 5 chassis.

A single IR detection sensor is unable to indicate the direction in which the shaft is spinning. Quadrature rotary encoders, shown in Figure 9, make use of two IR sensors. The sensor pair are mounted 90^o out of phase with each other. Spinning the encoder creates two high/low signal patterns which are 90^o out of phase, as shown in Figure 10. If the shaft is spinning in one direction the state sequence is $S_1S_2S_3S_4$, and the sequence is reversed if the shaft is spinning in the opposite direction. There are two Quadrature encoders equipped to the motors on the Rover 5 chassis, like the one shown in Figure 8. The resolution, which is measured in counts per revolution (CPR), tells us

how many pulse transitions to expect for one given wheel revolution. The encoders on the Rover 5 chassis undergo 1000 pulse transitions every 3 revolutions - the encoders are aptly named 333 CPR. This information can be used to calculate the distance travelled by a wheel for 1 pulse count. If the Rover 5 chassis has wheels equipped of R millimeters in diameter, then the distance travelled for a single revolution is the wheel circumference:

$$C = 2 \cdot \pi \cdot radius = \pi \cdot R = 1 \ \ \texttt{rev}$$

Hence, if there are 1000 pulse transitions for 3 revolutions implies that the angular displace for a single pulse, β_{pulse}, is given by:

$$\begin{aligned} 1000 \times \beta_{pulse} &= 3 \ \ [\texttt{rev}] \\ \beta_{pulse} &= \frac{3 \ \ [\texttt{rev}]}{1000} \times \frac{2 \cdot \pi \ \ [\text{rad}]}{1 \ \ [\texttt{rev}]} \\ \beta_{pulse} &= \frac{6 \cdot \pi}{1000} \ \ [\text{rad}] \end{aligned} \tag{1}$$

Assuming we have some sampling period, ΔT, and the pulse count is specified as N_K for discrete time k, then using equation (1) we can arrive at an estimate for angular velocity:

$$\omega_k = \frac{(N_k - N_{k-1}) \cdot 6 \cdot \pi}{1000} \ \ [\text{rad}/\,\text{sec}] \tag{2}$$

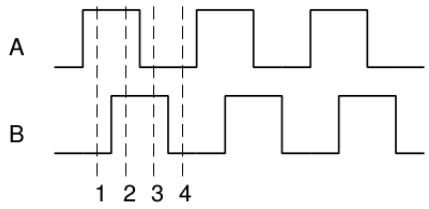

Figure 9: The two idealised signals generated by the IR sensor pair on a Quadrature Encoder. The two waves are 90^o out of phase.

A simple example of how *odometry* works can be seen by analysing straight-line motion, where the angular velocity of the wheels are equal: $\omega_{LW} = \omega_{RW}$. Consider Figure 7, where $X_I[k]$, and $Y_I[k]$ refer to the robots position in the global environment at some instant n, and the robot's current heading is given by θ. Suppose the robot travels at some linear velocity, $(R\omega_k)$, over the discrete time period $\Delta T = t_k - t_{k-1}$. Note that R is the wheel radius of the robot, and the angular velocity is measured by the rotary encoder. The global position of the robot at time t_k is given by:

$$\begin{aligned} X_I[k] &= X_I[k-1] + (R\omega_k) \cdot \Delta T \cdot \sin\theta \\ Y_I[k] &= Y_I[k-1] + (R\omega_k) \cdot \Delta T \cdot \cos\theta \end{aligned}$$

Table 3 provides a summary of the advantages and disadvantages of using an optical rotary encoder to calculate odometry.

Table 2: Four unique states from the two signals can be observed - the positions of these states are shown in Figure 10.

State	**Ch A**	**Ch B**
S_1	HIGH	LOW
S_2	HIGH	HIGH
S_3	LOW	HIGH
S_4	LOW	LOW

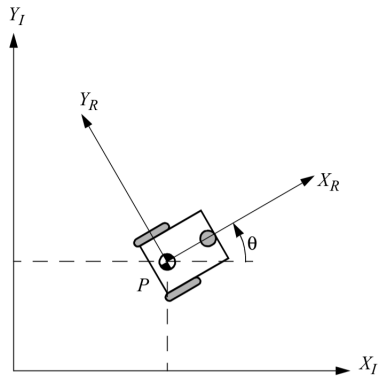

Figure 10: The robot in the global coordinate frame denoted by the subscript I. The coordinate frame attached to the robot is denoted with a subscript R, and allows us to define the robot heading, θ.

Table 3: Summary of the advantages and disadvantages of using optical rotary encoders to calculate odometry

Advantages	Disadvantages
• Two quadrature optical encoders come pre-installed with the Rover 5 chassis specified by the client • Easy to understand, and straightforward to implement • The quick brown fox jumped over the lazy dog	• There are errors present in the measurements provided by encoders given the discrete nature of their pulses • Odometry position is based on a cumulative method, meaning that over time errors grow without bound • Odometry error is particularly sensitive to changes in robot orientation

3.1.2 Accelerometer & Gyroscopes (IMU)

Accelerometers are used to measure external forces acting on the vehicle, which can be used to determine dynamic accelerations (Borenstein, Everett, & Feng, 1996). It must be noted that the sensor cannot distinguish between static gravitational force, and dynamic forces - this means care is required when interpreting sensor results. The on-board accelerometer provided with the myRIO utilises a microelectromechanical system (MEMS). The device has two terminals, with finger like arrangement acting as tiny capacitance plates, as shown in Figure 11. One of the terminals has mass attached to it, and when the accelerometer is moved along the axis of interest, the plates move, which changes the capacitance. Measured change in capacitance provides an estimate of the acceleration. Note that there are actually three accelerometers in the myRIO device, each one mounted on a different mutually orthogonal axis.

Gyroscopes are used to measure changes in the vehicle orientation. A MEMS gyroscopic sensor has a mass which oscillates at some velocity. When an angular velocity is applied to the sensor, the mass experiences a force due to the Coriolis effect. Similarly to the MEMS accelerometer, the MEMS gyroscope makes use of tiny capacitance plates. It is the change in capacitance that allows the estimation of the applied angular velocity. A single axis MEMS gyroscope can be seen in Figure 12. Typically, a MEMS gyroscopic sensor has three gyroscopes mounted on mutually orthogonal axes.

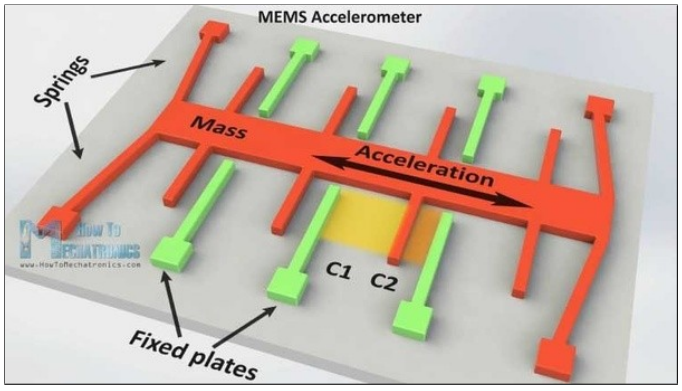

Figure 11: MEMS accelerometer. As the mass moves due to acceleration, the plates come closer together, or further apart, causing a measurable change in capacitance.

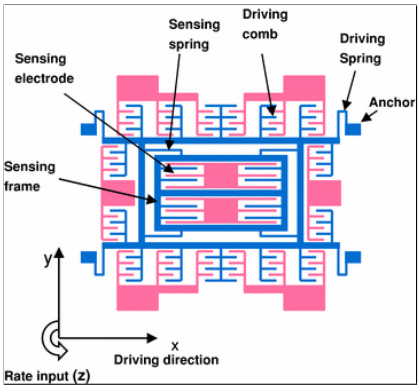

Figure 12: MEMS gyroscope. the plates come closer together, or further apart, causing a measurable change in capacitance.

An accelerometer and gyroscope used in conjunction is often referred to as an Inertial Measurement Unit (IMU). An IMU can be employed to determine relative localisation of an AWR - the process can be seen in Figure 13. In this instance, the robot is constrained to movement in the x-y plane, so we position the z-axis of the gyroscope orthogonal to this plane in order to detect rotational velocities about the z-axis. These rotational velocities are integrated and added to the initial heading to determine the robot's heading with respect to the Global frame. The x and y axes of the accelerometer are fixed on the robot and aligned with X_R and Y_R, respectively (as shown in

Figure 10). Detected accelerations in either the x or y axes can be integrated once to determine the relative velocity, or twice to determine the relative position. Combining these with the initial conditions allows the determination of the global velocity, and position. According to XXXX using accelerometers for inertial navigation is prone to error since the sensor measurements are integrated twice. Table 4 summarises the advantages and disadvantages of using accelerometers and gyroscopes for inertial navigation.

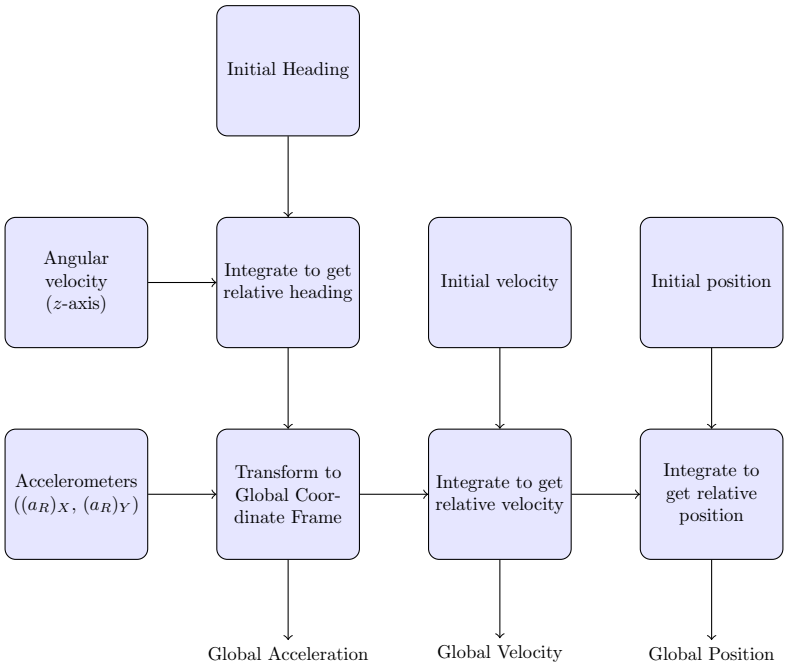

Figure 13: Computational flow diagram used for determining the global heading, acceleration, velocity, and position using inertial navigation.

Table 4: Summary of the advantages and disadvantages of using an IMU for inertial navigation

Advantages	**Disadvantages**
• The myRIO has a built in accelerometer • The sampling frequency is higher compared to optical encoders	• Integrating the reported acceleration from the sensors means that this type of navigation is sensitive to errors • Typically, due to propensity for error, these systems are used concurrently with absolute localisation such as SLAM, which is technically challenging to implement

3.2 Exteroceptive Sensors

Exteroceptive sensors are those sensors which acquire information from the robot's environment. Examples of this type of information include distance measurements, light intensity, and sound amplitude. The two main applications of these sensors for AWRs are map generation, and absolute localisation. The remainder of this subsection will take a closer look at the advantages and disadvantages of three of these sensors: infrared distance, time of flight ultrasonic, and an RBGD camera.

3.2.1 Infrared Distance Sensor

The basic operating principle of the IR sensors is based on infrared light. By looking at where the light hits the detector, it is possible to calculate the angle of the light and from that angle, derive the distance to the object. Figures 14 and 15 illustrates the basic operating principle of the ranging technique.

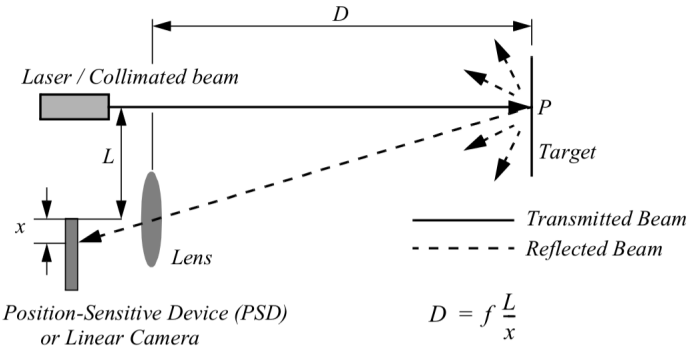

Figure 14: The 1D IR ranging sensor sends IR light which reflects from the target surface, entering the IR sensor lens at some angle. Triangulation is used to determine the distance to the target.

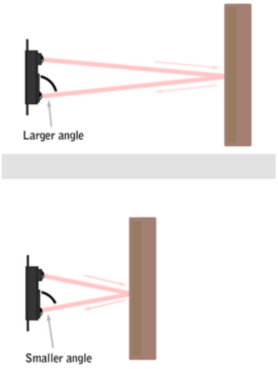

Figure 15: An example showing how the angle of reflected IR light varies on the IR sensor array for different target distances.

Two different 1D IR ranging sensors were selected, based on their different ranging properties. The SHARP GP2YD120 has an accurate measurement range of 40 millimetres to 300 millimetres, and the SHARP GP2Y0A21YK0F has an accurate measurement range of 100 millimetres to 800 millimetres. The SHARP GP2Y0A21YK0F sensor can be seen in Figure 16. In both SHARP IR sensors the dependence between voltage output and corresponding distance is not linear, as shown in the Figures 16 and 17 below. There are 2 approaches to calibrating a sensor model for use on a mobile robotics platform. One way is to focus on a linear region of operation, and determine a simple linear model using regression techniques. This linearised model can then be implemented programatically, however, the result will carry some error since the linear model is only an approximation for the non-linear behaviour of the sensor. Another method simply relies on extracting voltage and distance values at a reasonably high resolution, and programatically implementing these as a discrete lookup table. This allows the sensor to be used over a broader range of operation, but also carries error given the discrete nature of measurements. Table 7 summarises the advantages and disadvantages of this sensor.

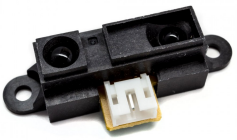

Figure 16: A picture of the SHARP GP2YD120 sensor.

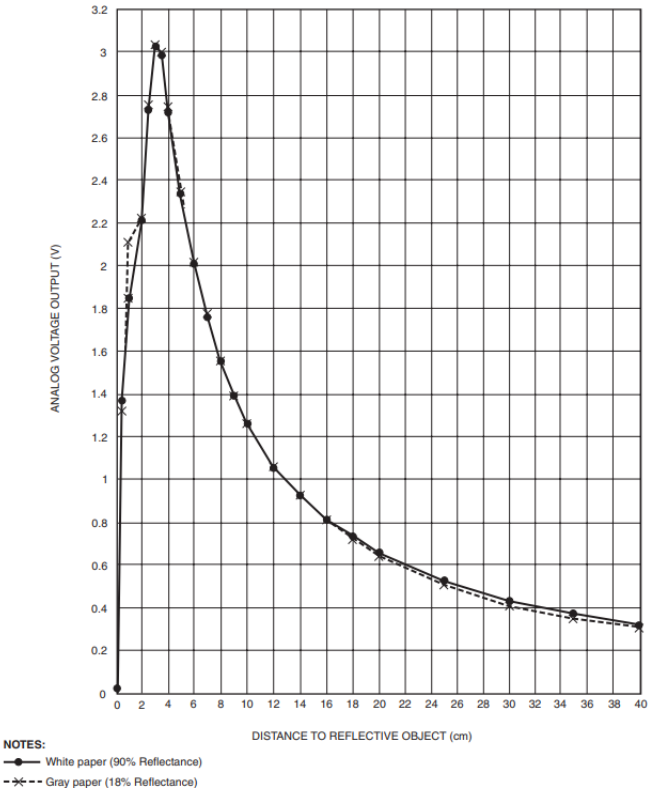

Figure 17: Voltage and ranged distance profile for the SHARP GP2YD120 sensor

Table 5: Values corresponding to the voltage and ranged distance profile for the SHARP GP2Y0A21YK0F sensor

Voltage Output (V)	**Distance** (cm)
3	3.5
2.8	4
1.6	8
1.1	12
0.85	16
0.7	20
0.6	24
0.5	28
0.4	32
0.35	36
0.3	40

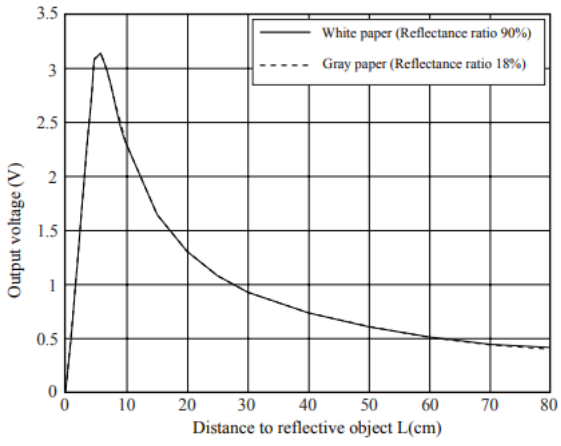

Figure 18: Voltage and ranged distance profile for the SHARP GP2Y0A21YK0F sensor

Table 6: Values corresponding to the voltage and ranged distance profile for the SHARP GP2Y0A21YK0F sensor

Voltage Output (V)	**Distance** (cm)
3.1	5
3.2	6
3	7
2.7	8
2.3	10
1.6	15
1.25	20
1.1	25
0.8	30
0.75	40
0.6	50
0.4	80

Table 7: Summary of the advantages and disadvantages of the 1D IR ranging sensors

Advantages	Disadvantages
• Low cost • Low power • Reasonable accuracy for close objects	• Poor accuracy for objects at a distance - limited by geometry

3.2.2 Ultrasonic Time of Flight Sensor

Like the 1D IR ranging sensors, an ultrasonic sensor provides distance measurement. The basic operating principle is illustrated in the Figure 19. It works by transmitting an ultrasonic (well above human hearing range) burst and providing an output pulse that corresponds to the time required for the burst echo to return to the sensor. By measuring the echo pulse width, the distance to target can easily be calculated.

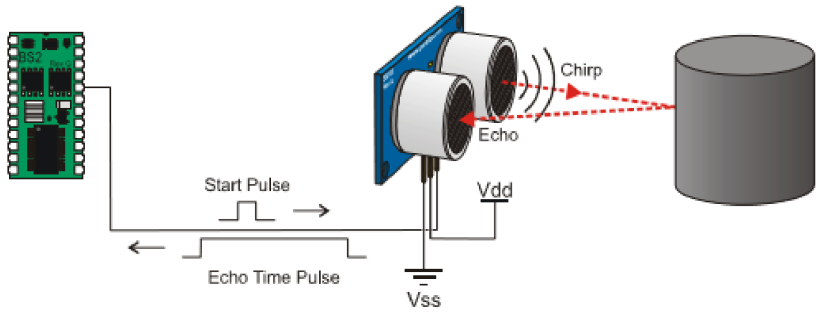

Figure 19: Typically an ultrasonic sensor will emit a sonic burst and trigger an ECHO pulse. The ECHO pulse is triggered again when the sonic burst returns to the sensor. The time duration of the ECHO pulse is used to measure the distance to the target.

The distance can be calculated by multiplying the speed of sound, ν, by the total time of travel, Δt. Distance to target, L, can be determined as follows:

$$L = \frac{\Delta t \cdot \nu}{2} [\text{cm}]$$

Note that the 2 in the denominator represents the fact that ultrasonic wave crosses over the unknown distance twice. Noting that the speed of sound is 343 $\text{m}\,\text{s}^{-1}$ at 20°C. Hence,

$$L = \frac{\Delta t \cdot 343}{2} = 171.5 \times \Delta t \;\; [\text{m}\,\text{s}^{-1}]$$

Principal factors affecting the range ability are ambietn temperature, T_C. The speed of sound is very sensitive to variations in temperature, hence it is important to calibrate the sensor in a similar temperature to the operating environment. The equation below describes the sensitivity level of the sensor in response to the temperature variations:

$$\Delta\nu = 331.5 + (0.6 \times T_C)$$

The advantages and disadvantages of this sensor are summarised in Table 8.

Table 8: Summary of the advantages and disadvantages of the ultrasonic time of flight sensor

Advantages	Disadvantages
• Simple to implement • Low power • Reasonably accurate over a range of distances	• Accuracy of the sensor is based on the acoustic reflective properties of the target • Sensor can give incorrect reading due to cross talk from other sensors, or reflected acoustic waves • Targets with non-orthogonal surfaces may produce errors

3.2.3 RGBD Sensor

An RGBD sensor captures a 2D pixel array on red, green, and blue channels using a monocular camera. Further, the sensor captures depth information by measuring the deformation of reflections from structured infra-red (IR) light emitted into the environment (Ganganath, & Leung, 2012). The Microsoft Kinect, shown in Figure 20, is a low cost RGBD camera which is readily employed in robotics projects requiring high fidelity environmental information.

Figure 20: The Microsoft Kinect sensor, which comes with an optical RGB camera, IR structured light emitter, and IR sensor. The sensor combination can be used to create 3D point clouds of 640 × 480 resolution, with 11-bit depth resolution.

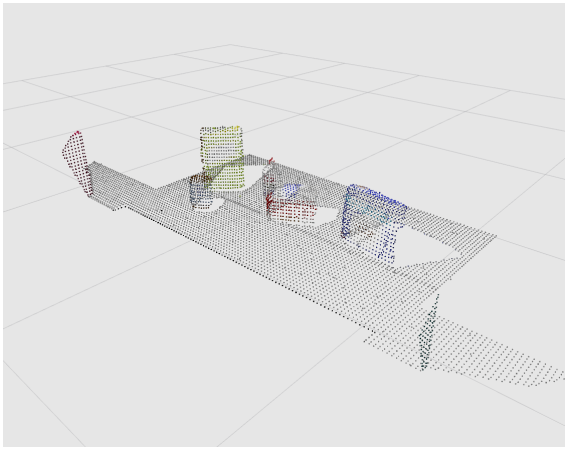

Figure 21: Sensor data taken from a Microsoft Kinect and used to create a 3D point cloud using the `pcl` library. The signal noise was removed using a statistical outlier filter, and the number of points in the point cloud reduced using Voxel downsampling. Image taken from Reynolds (2018).

The Kinect depth sensor, and optical sensor, operate at a resolution of 640 × 480 pixels. Combining the optical RGB and IR depth information, allows the development of a point cloud consisting of approximately 300000 points (Kamarudin, Mamduh, Shakaff, & Zakaria, 2014). Typically, some of the points are removed to reduce the amount required computational resources needed to process a point cloud. This is done using a process called Voxel downsamping (Reynolds, 2018). An example of a downsampled point cloud, generated from Kinect sensor data, can be seen in Figure 21. Point cloud representations of environments are desirable as they allow for the a number of sophisticated solutions to difficult problems associated with designing an AWR.

Table 9: Summary of the advantages and disadvantages of Kinect sensor

Advantages	Disadvantages
• Sensor provides access to 3D point cloud which can be used for advanced computational schemes such as SLAM • Optical RGB sensor could be utilised to detect the red exit panel using an Hue, Saturation, Value (HSV) image segmentation filter	• Technically challenging to implement many of the computational schemes that take advantage of the sensor • Since AWR navigation is in 2D, additional information from 3D point clouds are not fully utilised • Sensor is expensive • Sensor requires a 12 V power supply

4 Locomotion Control

Locomotion, according to the Concise Oxford Dictionary, is the motion from place to place. In the context of a mobile robot, locomotion is the process by which the robot moves. This requires an application of force (Dudek, & Jenkin, 2010), which in this instance is provided by two DC brushed motors, driving 86.8:1 gearboxes. The motors come pre-equipped with the Rover 5 chassis - the full motor specifications can be found in Table 1.

Locomotion control is an important design consideration when building a mobile robot. The problem can be divided up into two categories: Low Level Control, and High Level Control. This section of the report discusses the design options for low level control strategies, and then provides some background information for high level control strategies.

Figure 22: Specifications of the motors which come equipped with the Rover 5 chassis

Characteristic	Description
Voltage	7.2 V
Unloaded Current	210 mA
Stall Current	2.4 A
Max Speed	25 $\mathrm{cm\,s^{-1}}$

4.1 Low Level Control

Low level control strategies are concerned with controlling the actual output of the motors - angular velocity (or angular position). The simplest way of accomplishing this is using open loop control. The basic control structure can be seen in Figure 23. Basically, the desired reference value, r, is fed to the controller, which send a control signal, u, to the Open loop control systems are those in which actual system output has no effect on the control action (Ogata, 2010). The advantages of this type of control are that it is simple to implement with software.

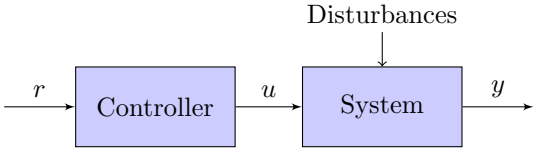

Figure 23: Open loop control structure sees a reference signal, r, sent to the controller, which sends a control signal, u, to the system in order to control the desired output, y. The output does not need to be measured by sensors in this instance.

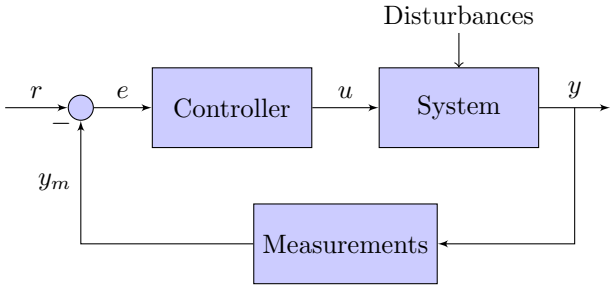

Figure 24: Closed loop control measures the output variable y using sensors, and feeds this back for comparison with the reference signal. The difference between the output and the reference is used as input to the controller, meaning that a small error results in a small control action.

The principal drawback is that the control system is sensitive to unexpected loads on the shaft, or other disturbances which will cause the device to deviate from expected operation. Moreover, this control strategy is sensitive to changes in plant variables such as the system mass, or shaft loading. The alternative to open loop control is closed loop control. Ogata notes that the defining characteristic of these control systems is the monitoring of the output, which is used in a feedback loop to determine the error between the output and the desired reference signal. It is this calculated error value which is used as input to the controller. The closed loop control structure can be seen in Figure 24. The main draw card of closed loop control is that it avoids the problems encountered with the open loop controller, that is, closed loop control:

- is insulated against stochastic perturbations which are not modelled in the system dynamics; and
- is insensitive to internal variations in system parameters.

There are many different types of closed loop controllers that are available, however, one of the most commonly used control mechanisms is the PID controller (Dudek & Jenkin, 2010). PID stands for proportional integral derivative controller. The controller receives error $r(t) - y(t)$, denoted as e, as input and outputs the control signal, which is sent to the system. The control signal, $u(t)$, is expressed as:

$$u(t) = K_p \cdot e(t) + K_i \cdot \int e(\tau)d\tau + K_d \cdot \frac{de(t)}{dt}$$

There are three different parameters which can be adjusted to tune the control to the required system: K_p, K_i, and K_d. The ubiquity of PID controllers means that although some effort is required to appropriately tune them, implementation in a digital system is relatively well documented and straight forward. Table 10 summarises the advantages and disadvantages of using either open loop control, or closed loop PID control.

Table 10: Summary of the advantages and disadvantages of open loop control, and closed loop control

	Open Loop Control	**Closed Loop Control**
Advantages	• Simple, quick and cheap to implement • Reasonable control for systems with no variance on internal parameters	• Widely used in industry - well documented implementation in digital systems • Will correct for system disturbances, or stochastic perturbations of inputs • Insensitive to variation in system parameters
Disadvantages	• Does not correct for system disturbances or stochastic perturbation to input • Controller is sensitive to changes to internal system parameters, such as unexpected shaft loadings	• Additional computational load on the processor • Takes time to implement and then tune the controller for the desired application

4.2 High Level Control

According to Siegwart & Nourbakhsh (2004), a commonly used architecture for map-based navigation involves a planning phase and a motion control phase, as shown in Figure 25. The planning phase is comprised of two main tasks: exploration strategies (discussed in Section 7), and path planning. For the purposes of this report, high level control has been used as a term to encapsulate path planning, and motion control. Path planning is highly dependent on a number of different factors such as the type of map that is used, the obstacle avoidance strategies employed, and the mobility configuration of the robot. Typically, once a path has been determined, it is broken up into a set of discrete poses (x, y, θ) and this is passed as an argument to the motion control phase. It is beyond the scope of this report to provide detail on path planning strategies, however, some path planning schemes will be outlined in Section 9 which discusses robot operation for 3 nominated designs.

The other main aspect of high level control is concerned with moving the robot from one pose in the environment to another pose in the environment - referred to as motion control in Figure 25. In the case of wheeled robots, a pose is made up of position in 2D space, (x, y), and heading, θ. A previously mentioned, a planned path is typically decomposed into a list of intermediate poses along the path. Executing the planned path is simply a matter of ensuring that the robot passes through each of these intermediate points.

If the pose parameters (x, y, θ) could be controlled directly, then the solution to this control problem would be trivial. Unfortunately, robotic systems do not normally have control of these parameters, rather, they have control over position, and velocity of motors (or actuators). This is often referred to as having control over the Joint Space. Hence, the problem can be stated as follows:

In order to achieve some final pose, given some initial pose, how should the robot's joints be manipulated?

This problem is often referred to as the Inverse Kinematic problem. Indeed, for many different robots this is a solved problem - especially for multiple degree of freedom industrial robot arms. It must be noted, however, that for many simple AWR mobility configurations, the inverse kinematic problem is difficult to solve due to non-holonomic constraints.

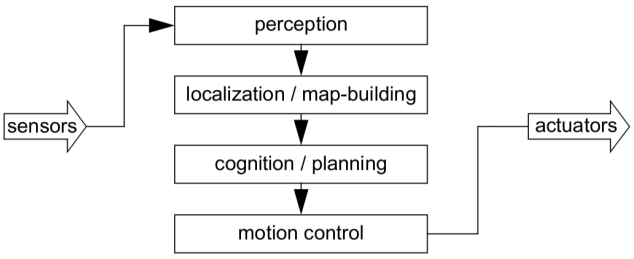

Figure 25: A common architecture which is employed for map-based navigation. This architecture is comprised of 4 phases: perception, localisation/map building, path planning, and motion control. (Siegwart & Nourbakhsh, 2004).

5 Map Building

Complicated tasks such as exploration and object avoidance can be undertaken without an internal representation of the robot's environment, however, some tasks such as absolute localisation or path planning become increasingly difficult to solve. According to XXXX the most natural representation of a robot's environment is a map. This section of the critical analysis compares the advantages and disadvantages of using a mapping scheme, or not mapping scheme.

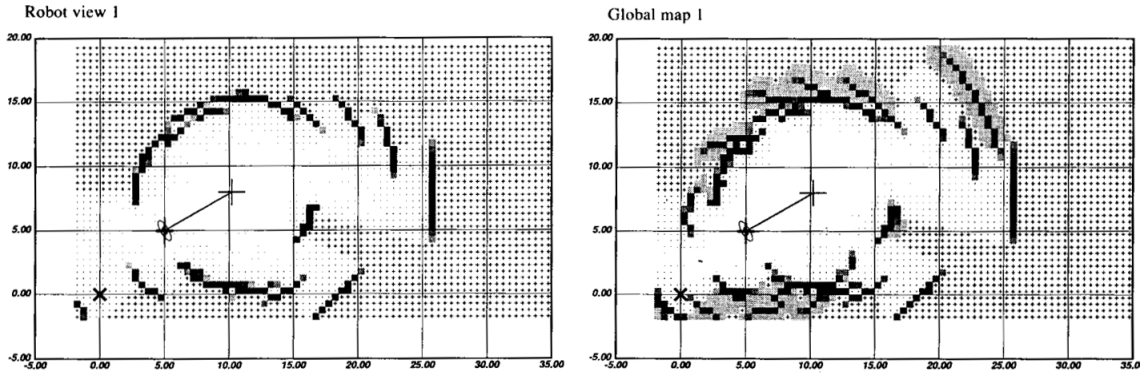

Figure 26: An example of a mobile robot mapping an unknown environment using an occupancy grid, taken from Alberto Elfes' 1989 paper *Using Occupancy Grids for Mobile Robot Perception and Navigation*

One of the single most common approaches to mapping is a fixed decomposition of an environment. This method was famously pioneered by Alberto Elfes in 1989, who referred to this representation as an *occupancy grid*. Elfes

builds an *occupancy grid* by overlaying the environment using a set of discrete cells, as shown in Figure 26. Each of the grid's cells is either filled (by part of an obstacle) or is empty (part of free space). *Occupancy grid* mapping is attractive as it allows the user to build a representation of the environment using data from multiple sensors of different type - referred to as sensor fusion in the literature. Given the stochastic nature of the sensor data, Elfes employed a probabilistic approach to determine whether a cell was occupied or not. This method turned out to computationally expensive, and in 1991, Borenstein & Koren improved on Elfes method for updating cells using a *Vector Field Histogram*.

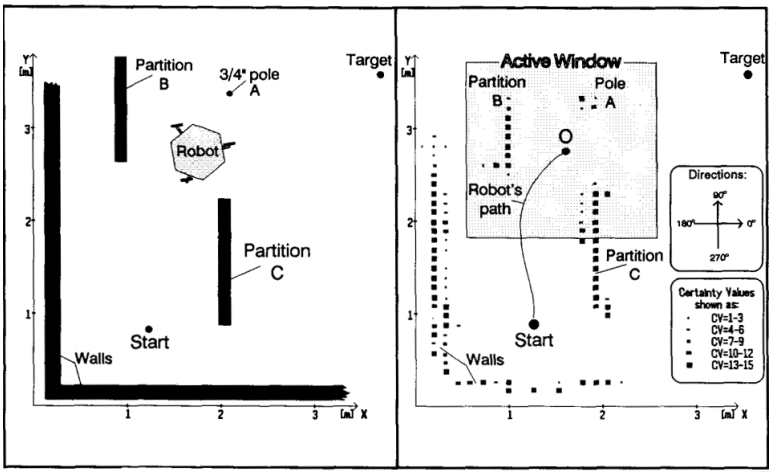

Figure 27: This picture is taken from Borenstein & Koren's paper titled *The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots*. The right hand panel shows the robot's environment, and the left hand panel shows the occupancy grid representation of that environment, which has been built using a *Vector Field Histogram*

Essentially, Borenstein & Koren updated the occupancy of cells in the robot's active window at the sensor sampling frequency. The microprocessor would receive sensor data, and if the sensor detected an obstacle, then the cell value corresponding to the obstacle location was incremented by 1. Simultaneously, every cell value between the detected obstacle, and the robot was decremented by 1. The high frequency update of this approach represented a probability distribution similar to that of a histogram. Cell values were allowed to vary from 0 which represented free space, up to 15 which represented a high certainty that the cell contained an obstacle. Figure 27 shows an example environment on the left, and a corresponding histogram representation on the right. The advantages of Borenstein & Koren's approach was that it achieved similar results as Elfes' *occupancy grid*, but was significantly less expensive from a computational standpoint. Table 11 provides a summary of the advantages and disadvantages of not mapping an environment compared with mapping an environment.

Table 11: Summary on the advantages and disadvantages of not mapping, and mapping using a occupancy grid approach

	No Mapping	**Mapping with an Occupancy Grid**
Advantages	• Simple to implement • Does not require memory • Computationally fast	• Provides the robot with access to more sophisticated navigation, path planning, and obstacle avoidance strategies
Disadvantages	• Leaves robot with limited options for navigation strategies, and obstacle avoidance • The full environment may never be explored since the robot has not internal construct of what it has and has not explored	• There is a memory requirement to store the map • Higher levels of map fidelity require finer map discretisation resulting in greater memory requirements • Technically challenging to implement

6 Absolute Localisation

Gutmann & Fox (1998) define localisation as the task of estimating position and orientation of a mobile robot in some environment. Goel, Roumeliotis, & Sukhatme (1998) provide further distinction using two subcategories: relative localisation and absolute localisation. They define these subcategories as follows:

- **Relative localisation:** evaluating the position and orientation using information provided by various on-board sensors, such as optical rotary encoders
- **Absolute localisation:** obtaining the absolute position using environmental landmarks, or beacons

Why is absolute localisation important? As previously noted, techniques which calculate relative localisation, such as odometry, are imperfect - they are prone to the accumulation of error. This is due to approximations used in dynamic equations governing robot motion, and errors from the sensor readings that odometry depends on. This can lead to the robot's internal position belief to diverge from its actual position over time. Odometry is particularly sensitive to errors in orientation - even small orientation errors can lead to large errors in position belief, as shown in Figure 28. Furthermore, this can corrupt the robot's mapping system which path planning and navigation more difficult. An example of this can be seen in Figure 29. The left panel shows the exact environment map, indicated with thick dark lines. The desired robot trajectory is shown by the light grey line.

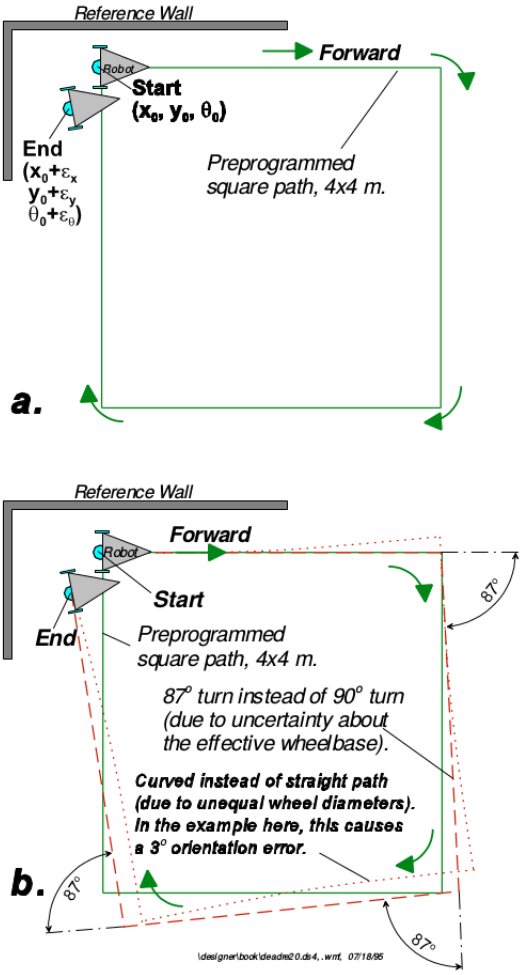

Figure 28: An example of basic odometry errors taken from Borenstein's book *Where am I now?*

The middle panel shows the exact robot environment, and the trajectory which has been corrupted by odometry error. The panel on the right shows the robots internalised map of the environment, which has been corrupted by odometry error. It must be noted that if a robot does not have prior information about the environment, that is the robot is not provided with a map, then an additional problem arises. The robot cannot accurately map if it does not know where it is in the environment, and at the same time the robot cannot know where it truly is if it does not have an accurate map. This problem, in the literature, is known as the simultaneous location and mapping problem (SLAM). According to Durrant-Whyte & Bailey (2006) the SLAM problem has been solved both theoretically, and for practical implementations. These types of solutions represent state-of-the-art in AWR design, and the implementations of these strategies would take considerable time and cost. The description of how these these methods work are mathematically involved and beyond the scope of this report, however, it must be noted that successful implementation would see increased performance in AWR navigation, obstacle avoidance, and autonomous exploration. Table 12 provides a summary of the advantages and disadvantages associated with having no localisation strategy, and using SLAM.

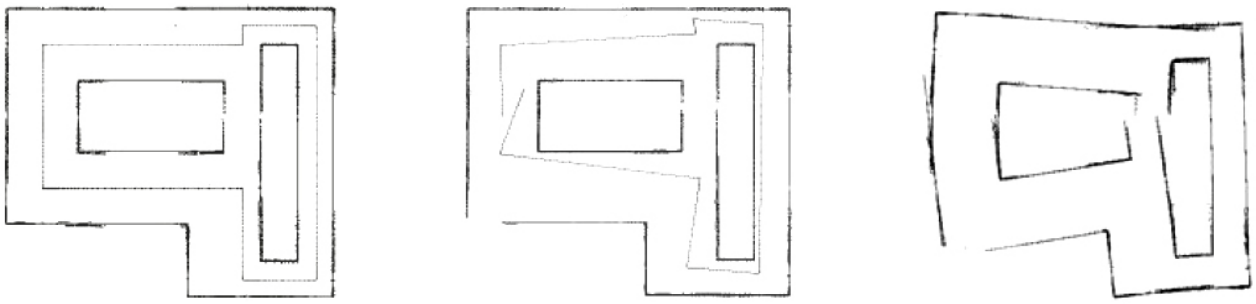

Figure 29: The image on the left shows the exact model of the robot environment (dark line), and the desired trajectory around the environment (light line). The middle image shows the exact robot environment, and the trajectory which has been corrupted by odometry error. The image on the right shows the robot's internalised map of the environment, which has been corrupted by odometry error. Images taken from Okamoto & Guizilini (2010).

Table 12: Summary of the advantages of no absolute localisation implementation, compared to a SLAM impementation

	No Localisation	**Simultaneous Localisation & Mapping (SLAM)**
Advantages	•	•
	•	
	•	
Disadvantages	•	•
	•	
	•	

7 Exploration Strategy

Exploration is defined by Yamauchi (1997) as the act of "moving through an unknown environment". An exploration strategy is implemented in the robot's software, and is a critical design consideration for successful autonomous navigation. This section of the report will look at the advantages and disadvantages of having a basic stochastic exploration strategy, compared to a strategy which is more computationally involved.

A stochastic exploration strategy, sometimes called reactive navigation, implements the AWR with with a set of behaviours which are triggered when the robot encounters some known environmental structure. A simple example of would be software which tells the robot to drive straight until it's sensors locate an obstacle in front of the robot. The detected obstacle would trigger the robot to stop and initiate a 90^o turn to the right (or left). The robot would then continue in a straight line until it reached another obstacle. The strategy is referred to as stochastic because the robot bounces around the environment in a randomised way.

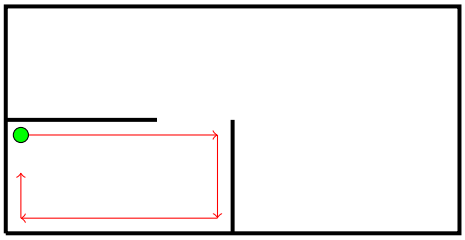

Figure 30: A simple example of environment design which would cause a robot with simple stochastic exploration strategy to fail. This figure shows the robot (a green circle) becoming trapped in an environment structure forever.

A stochastic exploration strategy is desirable because it is simple to implement and only requires minimal computational resources to complete the task. The main disadvantages lie in the fact that the robot may traverse an area of the environment multiple times before it explores all of the environment. Moreover, environmental topologies exist in which the robot could get stuck, referred to as *local minima* in the literature (XXXX, XXXX). Figure 30 demonstrates the basic exploration strategy described above, and shows an environmental topology in which the robot becomes trapped in a *local minima*. Yamauchi cites the central question to developing a good exploration strategy as:

Given what you know about the world, where should you move to gain as much new information as possible?

Yamauchi goes on to posit that an exploration strategy for gaining the most new information is to move the AWR to a boundary between obstacle free open space and uncharted territory. This strategy is referred to as frontier based navigation. The strategy relies on the use of an occupancy grid with additional representation for unexplored cells. At the start of robot navigation each cell is populated with a value, such as -1, indicating that the cell is unknown. When a cell has been explored, using Borenstein and Koren's *Vector Field Histogram* scheme outlined in Section 5, the cell is populated with a value ranging from 0 to 15. If a cell is below a certain threshold value (say 2) then it is considered free space.

The exploration strategy employs a process analogous to edge detection to find the boundaries between open space and unknown space - any obstacle free cell adjacent to an uncharted cell is labelled as a frontier edge. Frontier edge cells are grouped into frontier regions, and if the frontier region is big enough the centroid of this region is marked as a location that the robot will explore. An example of this process can be seen in Figure 31, which was taken from Yamauchi's 1997 paper *A Frontier-Based Approach for Autonomous Exploration*. The main advantages of this exploration strategy is that it guarantees full environment exploration, given that all areas of the environment are accessible. The principal drawbacks are that the software implementation is not straight forward, and requires more computational resources meaning that a more expensive microprocessor (like the myRIO) is required. Table 13 provides a summary on the advantages and disadvantages of stochastic exploration and frontier based exploration.

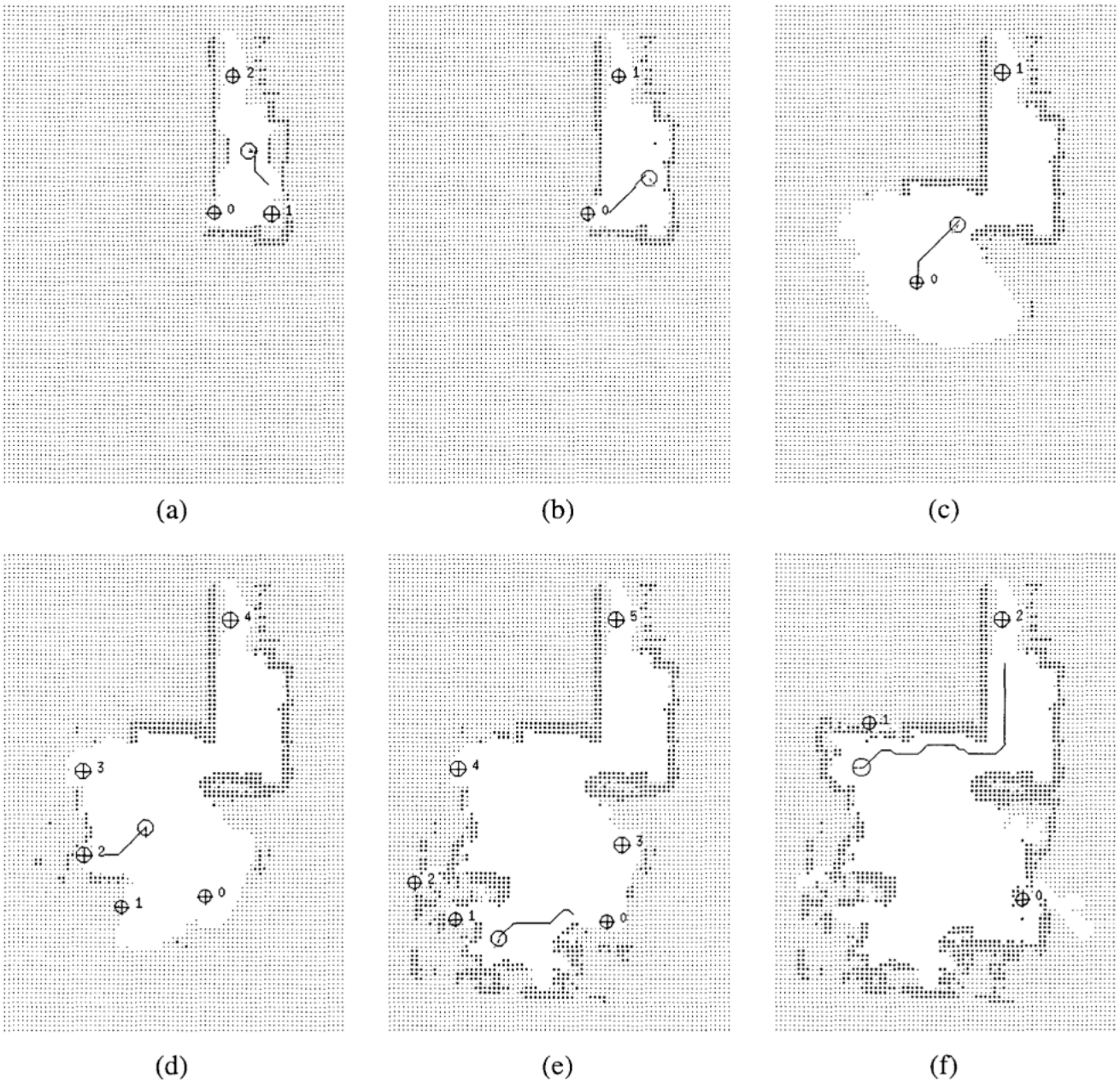

Figure 31: A picture illustrating Frontier-Based autonomous exploration taken from Yamauchi's 1991 paper titled *A Frontier-Based Approach for Autonomous Exploration.* Panel (a) depicts the AWR identifying 3 frontier regions for exploration, and panel (b) shows the AWR navigating to region 0.

Table 13: Summary on the advantages and disadvantages of using a stochastic exploration strategy, or a frontier-based exploration strategy

	Stochastic Exploration Strategy	**Frontier-Based Exploration Strategy**
Advantages	• Simple to implement • Computationally inexpensive	• Guaranteed to explore entire map, provided all map areas are deemed accessible from grid resolution on robot map • More likely that robot will converge on exit panel
Disadvantages	• May lead to robot getting trapped in local minima • Does not guarantee that entire map will be explored • Considerable risk that robot will not converge on exit panel	• Computationally expensive • Technically challenging to implement

8 Hardware Environment

8.1 Computational Device

There are two computational devices which are permitted to be used for the AWR:

- **Arduino:** 8-bit AVR microcontroller wqith 32KB flash memory, 1024B EEPROM. Low cost, low performance.
- **myRio:** Xilinx Z-7010 667 MHz processor, 512MB non-volatile, 256MB DDR3. High cost, high performance.

8.2 Power

There are three main devices that require a power supply namely, the myRIO, the motor controller to drive the motors and the Kinect device. The robots motors are rated at 7.2 V, and the myRIO requires a power supply between 6 and 16 V (14W consumption). The Kinect device requires a 12 V supply. To run all of the devices, some power arrangement is necessary. It must be noted that there are two different voltage levels:

1. 7.2V to run motors
2. 12V to run myRIO and Kinect devices

Each device requires an individual minimum current value, hence appropriate voltage and current supply requirements are critically important. Table 14 summarises the supply requirements for each device.

Table 14: Summary of the main power requirements key components of the system.

Device/Equipment	**Voltage** (V)	**Current** (A)
myRIO	12.00	1.20
Kinect	12.00	2.00
Motors	7.2	<2.50

On a top of the different power supply requirements, there is another concern for electrical supply design. The problem will possibly arise if one supply is used to supply power to all the devices. Brushed DC motors are notorious for introducing EMI (electro-magnetic interference). This interference/noise from the commutator brushes induces noise into the power line. It can interfere with the sensors, and can possibly damage myRIO by causing voltage dips. The motor draws a very high current at the start-up which can cause ripple voltage at the power input to myRIO

device. To eliminate the potential problems with the noise, the decision has been made to use two power supplies i.e 9V battery to power robots motors and 12V battery to power both myRIO and Kinect device.

9 Proposed Design Options

There were three main design options proposed in the statement of work: Low Complexity (Design Option 1), Moderate Complexity (Design Option 2), and High Complexity (Design Option 3). This section provides a list of the hardware elements and design choices for each design, in addition to providing a brief overview of how each system operates.

9.1 Design Option 1 (Low Complexity)

9.1.1 Hardware & Design Choices

Table 15: Summary of the hardware utilised in design option 1

Hardware	**Specification**	**Design Element**	**Specification**
Chassis	*Rover 5*	Mobility Configuration	*Differential Drive with omni-directional passive wheel*
Proprioceptive Sensors	*Optical rotary encoders*	Locomotion Control	*PID*
Exteroceptive Sensors	*SHARP GP2Y0A21YKOF*	Mapping	*None*
Computational Hardware	*Arduino*	Localisation	*None*
		Exploration Strategy	*Stochastic*

9.1.2 Operation & Software Description

This design is low complexity, easy to be constructed and required the lowest cost compared to other 2 options. It is operated by using a finite state machine and four sensors; one IR sensor on the front used for checking an obstacle-free path forward, two on the left and right side of the robot to assure the robot will not collide with walls and a final sensor will be employed on rear of the robot chassis used to detect if the robot is currently placed on the red exit panel. There are two types of IR sensor used for the design option 1 which are SHARP GP2D120 for the side of the robot and SHARP GP2Y0A21YKOF for the front of the robot. Operation diagram of the design option 1 is shown as below. Initially, the robot is placed at the starting point as the stop state. The sensors will check through the robot path if there are no obstacles present in the robot path the robot will moves forward using a constant velocity controlled by PID control. If the sensors detect that the path was blocked the robot will change state to the turn state.

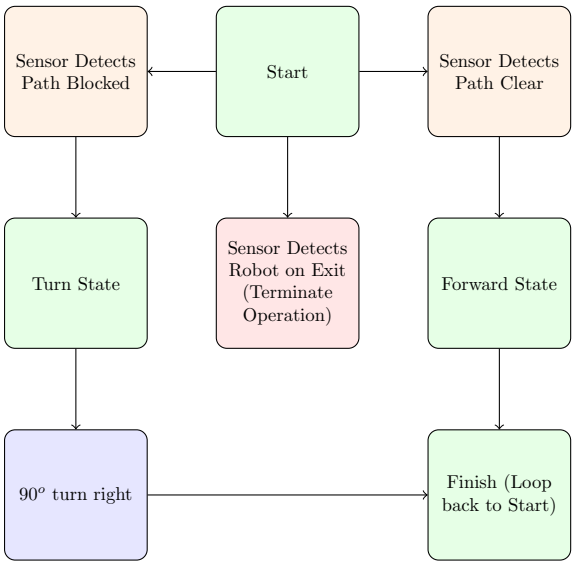

Figure 32: Flow diagram demonstrating the basic operation of the software for Design Option 1.

The robot performs to turn, for left turn the left motor will move forward by 90-degree pivot turn. Similarly, for right turn the right motor will perform reverse by 90-degree turn. Once the robot has completed turning, the robot will automatically transition state back to the stop state. Finally, if the robots sensors detect that it is on the red exit panel then the robot will alter state to the finish sate.

9.2 Design Option 2 (Moderate Complexity)

9.2.1 Hardware & Design Choices

Table 16: Summary of the hardware utilised in design option 1

Hardware	**Specification**	**Design Element**	**Specification**
Chassis	*Rover 5*	Mobility Configuration	*Differential Drive with omni-directional passive wheel*
Proprioceptive Sensors	*Optical rotary encoders*	Locomotion Control	*PID*
Exteroceptive Sensors		Mapping	*Vector Field Histogram*
Computational Hardware	*myRio*	Localisation	*None*
		Exploration Strategy	*Frontier-Based Exploration*

9.2.2 Operation & Software Description

Given the nature of the environment where the robot motion is limited to the predetermined terrain boundaries, the localization and mapping can be represented by a grid of uniform cells and the movement of the robotic vehicle is bounded to the valid steps from one cell to another. This way the navigation map can be reduced to the logic matrix with coordinate system. Using the inputs from mapping and the localization, the vehicle path planning algorithm allows robotic vehicle to follow a suitable path to reach the final destination. In line with the project purpose, the logical navigation rules for this design option are to go straight towards the final destination and, in case of obstacles obstructing the way, avoid it. The block diagram of the obstacle detection and path planning is shown below in Figure 33.

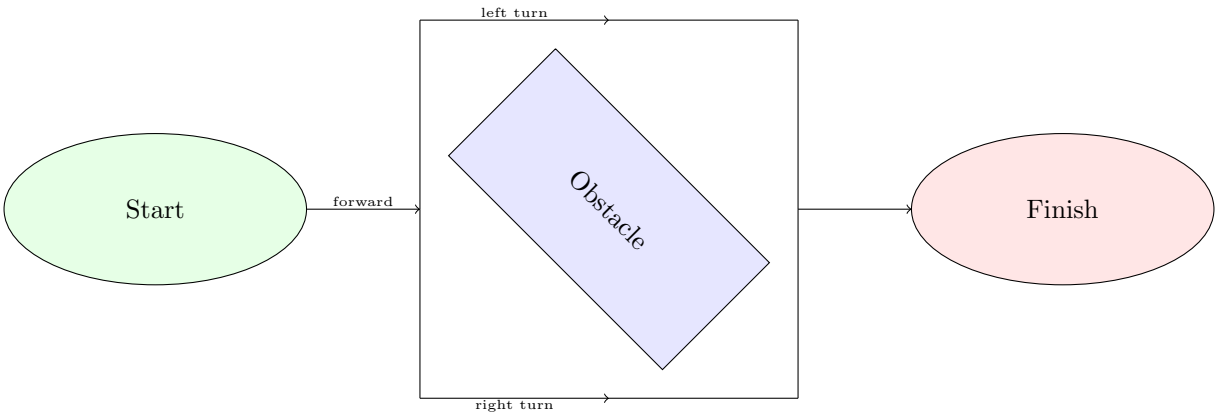

Figure 33: Navigation strategy for the robotic system.

As the robot travels through the environment (grid), it records measurements from its IR and Sonic sensors, the presence of an obstacle inside the angle covered by the sensors sweep area is indicated if the recorded measures from

The logic diagram of grid exploration can be seen in Figure 34. Firstly the robot checks the surrounding spaces for clearance and moves forward to the next spot, if the robot has finished mapping the maze, it stops its motors. If the robot has not finished mapping the maze, then it will check to see whether there are obstacles in front, to the left, and to the right, which is called a dead end. If the robot identifies itself at a dead end it moves backwards until there is a new spot to check. For instance, if the robot identifies free movement forward and left, the map stores 3, the value for forwards, in position 5 of the array.

As the robot continues its travelling through the maze it maps every path in the maze. If the robot identifies itself in a position that was already visited it reorients itself to the next available path from the previous spot. The travelling through the maze continues until the robot reaches its final destination identified by the red square on the terrain. The final destination is detected by the sensor located on the bottom of the robot.

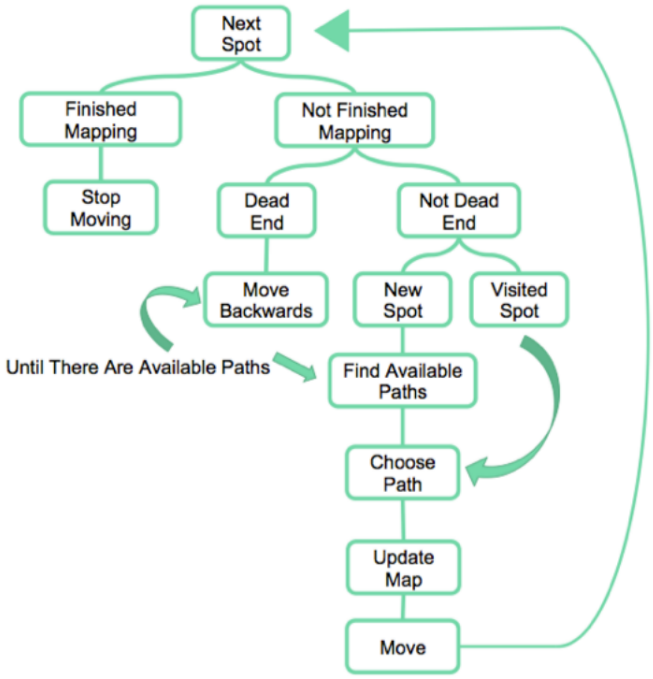

Figure 34: The logic diagram for the grid exploration used for Design Option 2.

the sensors reach a certain threshold. The robot presence on a place indicates that this place is open to be travelled and empty of obstacles. As the robot travels through the grid it stores the information of the previously visited places as to avoid unnecessarily revisiting locations it has already seen, allowing it to reach the goal faster.

9.3 Design Option 3 (High Complexity)

9.3.1 Hardware & Design Choices

Table 17: Summary of the hardware utilised in Design Option 3

Hardware	**Specification**	**Design Element**	**Specification**
Chassis	*Rover 5*	Mobility Configuration	*Differential Drive with omni-directional passive wheel*
Proprioceptive Sensors	*Optical rotary encoders*	Locomotion Control	*PID*
Exteroceptive Sensors		Mapping	
Computational Hardware	*myRio*	Localisation	
		Exploration Strategy	

9.3.2 Operation & Software Description

This design operates in the exact same way as design option 2, however, the Microsoft Kinect that is equipped provides two additional functions:

- The additional data captured by the Kinect is used in a sensor fusion arrangement with the IR and ultrasonic sensors to provide higher fidelity mapping - more sensors contributing to the *Vector-Field Histogram* map, means the map is more accurate;
- The Kinect will provide a rich enough dataset to implement an absolute localisation strategy (SLAM), which will be used to correct errors in *odometry*

As the robot moves around the map using the Frontier-Based exploration strategy, encoder information is used to predict the measurement of the robot's position, and sensor data is used to create a map of the environment using *Vector-Field Histogram* mapping. The Kinect sensor readings are used to develop a 3D point cloud, which the SLAM algorithm uses determine where the robot is located. The *odometry* position and SLAM estimate position are compared. If the positions match, then the robot continues as normal. If the positions don't match, then the robot checks to see if the the observations are expected, or unexpected. If the observations are expected then the robot will correct the odometry position. If the results are unexpected then the robot will reduce the credibility of the current map, and continue as normal.

Figure 35 illustrates the basic operation of the robotic platform, and acts as a high level flow diagram for the implementation of the software.

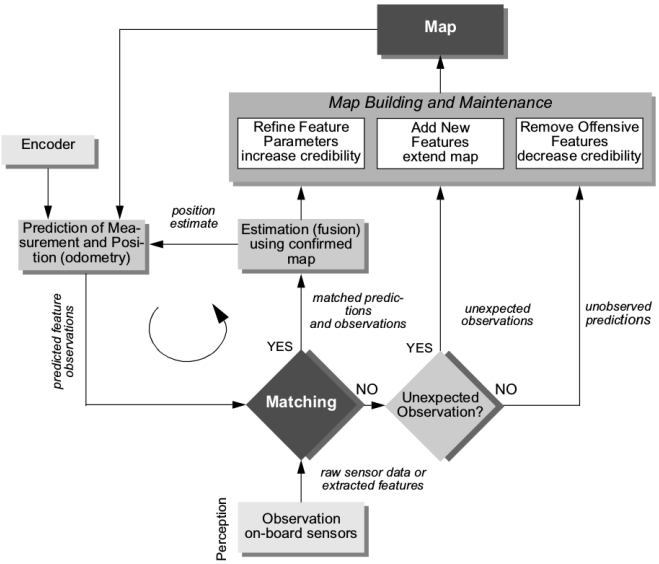

Figure 35: Flow diagram demonstrating how SLAM corrects the errors in *odometry* information

10 Budgets & Timeframes

Each of the three design options require the use of the Rover 5 chassis, the 4 channel motor controller, and the NI myRio Xilinx processor. The main differentiation in terms of cost comes down to the sensors that will be employed and the estimated man hours involved with the implementation. Table 18 provides the estimated budget for each option. The range of costs for the proposed options start at $1018.56 for the cheapest option, up to $1464.16 for the state of the art solution. The estimated man hours spent on each of the design options are as follows:

Table 18: Estimated time for each design option

Design Option	**Time Spent**
Design Option 1	60 hours
Design Option 2	100 hours
Design Option 3	120 hours

The costs for these products were sourced from the the following online retailers:

- **Mouser:** `https://au.mouser.com`
- **National Instruments** `https://www.ni.com`

Project time frame estimations are depicted using a Gantt chart which can be found on page 31.

Table 19: Budgeted costs for the sunk costs, and each design option

Sunk Costs		**Low Complexity**	
Rover 5 chassis	$68.81	3 × SHARP GP2D120	$47.85
myRio 1900	$814.00	1 × SHARP GP2Y0A21YK0F	$15.95
4 channel motor controller	$21.95		
Prototyping equipment (breadboard, wiring, etc)	$50.00		
Total	$ 954.76	Total	$63.80

Moderate Complexity		**High Complexity**	
3 × SHARP GP2D120	$47.85	2 × SHARP GP2D120	$32.90
1 × SHARP GP2Y0A21YK0F	$15.95	Intel RealSense R200 Camera	$127.50
1 × PING))) Ultrasonic distance sensor	$29.99	(optional) Scanse Lidar Sweep Scanner	$349.00
Total	$93.79	Total	$509.40

11 Critical Analysis Evaluation & Recommended Option

The evaluation of final design can be broken up into two categories:

- Elements which do not vary with each of the 3 design options (**Design Option Invariant Elements**); and
- Elements which do vary with each of the 3 design options (**Design Option Variant Elements**)

Every design option has its advantages and disadvantages; hence its important to weight each attribute according to how this element performs against the criteria and requirements set by the client. This report uses five main characteristics to assess all design choices - these are outlined as follows:

- **Speed (25% weighting):** one of the main requirement for this design project is speed, the vehicle can operate perfectly and execute the tasks but if it fails to execute it within 3 minutes, the task is deemed as failed. It is one of the most important factors.
- **Risk of failure (25% weighting):** risk of not executing the key milestones that make up the task
- **Time to build (25% weighting):** the longer it takes to build the product the less time is left for testing and removal of errors.
- **Complexity (15% weighting):** complex design puts at risk the execution of the task due to limited time-frame.
- **Cost (10% weighting):** its always least desirable to spend extra budget on the design. For the client the least spending is the best, but the trade-off is the quality. In this design task, the budget is not a main constraint, hence is not as important as the technical factors and functional compliance.

11.1 Evaluation of Design Option Invariant Elements

This section of the report provides an evaluation of those elements of the design which do not vary with each of the design options. Notably, these are the Mobility Configuration, the choice of Proprioceptive Sensors, and the Low Level Control Scheme. The evaluation is undertaken in the table below, with the most desirable design elements highlighted in green. A final design recommendation is provided in Section 11.3.

Mobility Configuration (Section 2)						
	25%	25%	25%	15%	10%	
Design Choice	**Speed**	**Risk of Failure**	**Time to Build**	**Complexity**	**Cost**	**Score**
Four motors, with Swedish 45 wheels	90	80	20	20	20	52.50
Two motors, with tracked wheels	60	50	80	80	90	68.50
Two wheels, with one passive spherical wheel	80	80	60	70	70	72.50
Proprioceptive Sensor (Section 3.1)						
	25%	25%	25%	15%	10%	
Design Choice	**Speed**	**Risk of Failure**	**Time to Build**	**Complexity**	**Cost**	**Score**
Optical Rotary Encoders (*Odometry*)	N/A	70	90	80	90	61.00
IMU (*Inertial Navigation*)	N/A	50	60	60	60	42.50
Low Level Locomotion Control (Section 4.1)						
	25%	25%	25%	15%	10%	
Design Choice	**Speed**	**Risk of Failure**	**Time to Build**	**Complexity**	**Cost**	**Score**
Open Loop Control	N/A	40	70	70	N/A	38.00
Closed Loop Control (PID)	N/A	70	50	60	N/A	39.00

11.2 Evaluation of Design Option Variant Elements

This section of the report provides an evaluation of those elements of the design which vary with each of the design options. Notably, these are the Computational Hardware, Exteroceptive Sensors, Mapping Strategy, Localisation Strategy, and Exploration Strategy. The evaluation is undertaken in two parts. The first table determines a score for each of the design elements, for each of the three design options. The set of scores for each of design options are totalled in the second table, and the most desirable design option is highlighted in green. Note that these tables can be found on page 28.

11.3 Final Recommendation

Summarising the findings from Sections 11.1 and 11.2 the final design should utilise two wheels, with one passive omnidirectional spherical wheel for it's mobility configuration. Optical encoders should be used to calculate the *odometry* of the robot, and closed loop control (preferably PID) should be employed for the low level control of the motors. Design Option 2 (Moderate Complexity) is the most desirable design option. This would see final design employ the myRIO as the computational hardware. Further, the design would employ 4 × IR ranging sensors: 1 × SHARP GP2D120 at the front of the robot, and 2 × SHARP GP2Y0A21YKOF at either side of the robot, and 1 × SHARP GP2Y0A21YKOF mounted underneath the robot to detect the red exit panel. An ultrasonic sensor would also be employed at the front of the robot. The sensor most likely to be utilised is the Matbotix MB1230 ultrasonic sensor. Both IR and ultrasonic sensors would be used in fusion arrangement to provide information to the robot's mapping system which is based on a modified *occupancy grid* called a *Vector-Field Histogram*. The robot would not employ an absolute localisation strategy, however, would use a *Frontier-Based* exploration strategy to identify areas of interest on the map.

Computational Hardware (Section 8.1)						
	25%	25%	25%	15%	10%	
Design Choice	**Speed**	**Risk of Failure**	**Time to Build**	**Complexity**	**Cost**	**Score**
Design Option 1 (*Arduino*)	N/A	N/A	60	70	70	32.50
Design Option 2 (*myRIO*)	N/A	N/A	50	50	30	23.00
Design Option 3 (*myRIO*)	N/A	N/A	50	50	30	23.00
Exteroceptive Sensor (Section 3.2)						
	25%	25%	25%	15%	10%	
Design Choice	**Speed**	**Risk of Failure**	**Time to Build**	**Complexity**	**Cost**	**Score**
Design Option 1 (*SHARP IR Ranging*)	N/A	N/A	60	80	90	36.00
Design Option 2 (*Fusion of IR and Ultrasonic Ranging*)	N/A	N/A	50	70	70	30.00
Design Option 3 (*Fusion of IR, Ultrasonic, & RGBD camera*)	N/A	N/A	20	30	40	13.50
Mapping (Section 5)						
	25%	25%	25%	15%	10%	
Design Choice	**Speed**	**Risk of Failure**	**Time to Build**	**Complexity**	**Cost**	**Score**
Design Option 1 (*No Mapping*)	20	20	90	80	90	53.50
Design Option 2 (*Vector Field Hist.*)	80	80	50	50	60	66.00
Design Option 3 (*Vector Field Hist.*)	80	80	50	50	60	66.00
Localisation (Section 6)						
	25%	25%	25%	15%	10%	
Design Choice	**Speed**	**Risk of Failure**	**Time to Build**	**Complexity**	**Cost**	**Score**
Design Option 1 (*None*)	N/A	50	90	90	60	54.50
Design Option 2 (*None*)	N/A	50	90	90	60	54.50
Design Option 3 (*None*)	N/A	90	20	10	40	33.00
Exploration Strategy (Section 7)						
	25%	25%	25%	15%	10%	
Design Choice	**Speed**	**Risk of Failure**	**Time to Build**	**Complexity**	**Cost**	**Score**
Design Option 1 (*Stochastic*)	10	10	90	80	80	47.50
Design Option 2 (*Frontier-Based*)	90	80	20	20	20	52.50
Design Option 3 (*Frontier-Based*)	90	80	20	20	20	52.50

Final Evaluation						
	Comp Hardware	**Exteroceptive Sn**	**Mapping**	**Localisation**	**Exploration Stg**	**Total**
Design Option 1	32.50	36.00	53.50	54.50	47.50	224.00
Design Option 2	23.00	30.00	66.00	54.50	52.50	226.00
Design Option 3	23.00	13.50	66.00	33.00	52.50	188.00

12 References

Barshan, B., Durrant-Whyte, H. (1993). An Inertial Navigation System for a Mobile Robot. *IEEE Conference on Intelligent Robots and Systems*, Yokohama, Japan, July 26-30, 1993

Borenstein, J., Everett, H., & Feng, L. (1996). Where am i? *University of Michigan for ORNL D&D program.*
Borenstein, J. & Koren, Y. (1991). Histogramic Mapping for Mobile Robot Obstacle Avoidance. *IEEE Transactions on Robotics and Automation*, 7 (4) pp 535-539

Dudek, G., Jenkin, M. (2010). Computational Principals of Mobile Robotics. *Cambridge University Press*

Durrant-Whyte, H., Bailey, T. (2006). Simultaneous Localization and Mapping: Part 1. *IEEE Robotics & Automation Magazine*, pp 99-108

Kapoor, S., Mahesh, K., Ruzinov, D., Battipaglia, J. (2017). Autonomously Solving Mazes with Robots. Retrieved from `http://soe.rutgers.edu/sites/default/files/imce/gov2017/Autonomously%20Solving%20Mazes%20with%20Robots.pdf`

Siegwart, R., Nourbakhsh, I. (2004). Autonomous Mobile Robots. *The MIT Press Cambridge, Massachusetts.*

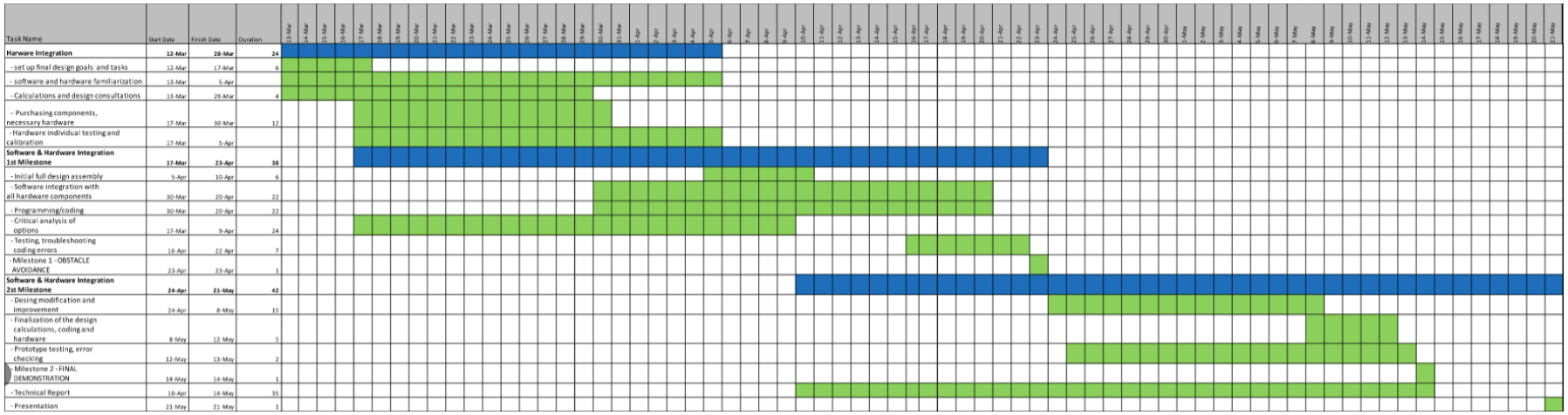

Figure 36: A gantt chart breaking down the phases of the project and detailing the labour assignment through the project life cycle.