SCHOOL OF ENGINEERING
ENGG341 Individual Project

Autonomous Navigation of Mobile Robots in Unknown Environments

Final Report

Sunprit Singh Jandu

201044722

Aerospace Engineering with Pilot Studies

April 2017

Supervisor: Dr. Mike Jump

183/1

School of Engineering
University of Liverpool
Brownlow Hill, Liverpool, L69 3GH

Summary

The aim of the project was to develop a program which enables the robotic platform Dr. Jaguar 4x4 Wheel to follow a minimum cost path between two points autonomously while avoiding obstacles in a dynamic unknown environment. Dr. Jaguar 4x4 Wheel is a skid steered rover with 4 powerful DC motors, one connected to each wheel. This thesis first presents the background research prior to beginning the program development. The theory of the chosen methods from the research is described subsequent to the research. The design approach and implementation are shown in the later stages of the thesis. MATLAB/Simulink was used to write and model the program. The modelling of the program could be divided into three sections: path planning, a physical representation of Dr. Jaguar 4x4 Wheel, and a control strategy for guiding the robotic platform over the planned path.

The D* algorithm was chosen for the path planning and was acquired from the Robotics Toolbox developed by Peter Corke[1]. It is developed in MATLAB and therefore easy to integrate with the physical system as well as the control strategy. The physical system was represented as realistically as possible by modelling each DC motor and the dynamic system of the robotic platform. The dynamic system consists of the wheel model and a representation of the robotic platform frame. The Coulomb Friction Model was used in the wheel model. The core idea of follow-the-carrot method was used to develop the control algorithm.

The program was successfully developed; however, it was not tested on the actual robotic platform. Thus, the results only include data from simulations run on MATLAB/Simulink. The program enables the physical system to track down the planned path with great precision and accuracy with the guidance of the developed control strategy.

[1] http://petercorke.com/wordpress/ The toolbox can be obtained from this link.

Index to Chapters

Acknoledgements

I would like to thank the University for giving me the opportunity to research an area which I was really interested in. I would like to thank all my professors as I have learned so much from them in the past 3 years. Furthermore, I would like to thank Dr. Paolo Paoletti and Dr. Mike Jump who gave me a lot of knowledge and advice throughout the project. I would also like to give special thanks to Ihusan Adam for taking his time to meet me, and providing me with useful information and tips about the project approach. The project would not be possible without the help of these people. Finally, I would like to thank my family and friends for supporting and encouraging me to achieve my goals.

Table of Notation

Notation	Description	Units
J	Moment of inertia of the rotor	kgm^2
B	Motor viscous friction constant	Nms
K_e	Electromotive force constant	$\mathrm{V \cdot s/rad}$
K_t	Motor torque constant	$\mathrm{Nm/Amp}$
R	Electric resistance	Ω
L	Electric inductance	H
ω	Angular velocity of the DC motor	rad/s
v^F	Longitudinal velocity of a wheel axle	$\mathrm{m/s}$
v^S	Lateral wheel velocity of a wheel axle	$\mathrm{m/s}$
v_S	Longitudinal slipping velocity	$\mathrm{m/s}$
v_f	Total velocity of slipping	$\mathrm{m/s}$
a	Wheel base	m
b	Wheel track	m
p	CG position with respect to GC specified in percentage <0,100>	-
W	Vehicle weight	kg
I	Moment of inertia of the robotic platform	kgm^2
F	Longitudinal force between tire and surface due to slip	N
S	Lateral force between tire and surface due to skid	N
φ	Heading angle of vehicle	rad
θ	Required heading angle to change vehicle course	rad
θ_{orient}	Desired pose angle of vehicle on reaching its target	rad
θ_e	Difference between heading and desired angle	rad
C	Tire stiffness	N/m
μ_0	Coefficient of friction	-

1.0 Introduction

1.1 Project Background

Autonomy in general means to make a logical decision based on available information. Autonomous robots could therefore be defined as machines that have free will; the choice to act based on applicable information (Clough, 2002). On the other hand, the choice to act based on the relevant information may be limited as it depends on the robot-program. For instance, robots working in assembly plants are fully autonomous but work in a fixed pattern (Autonomy Levels, 1985). The opposite of autonomous is automatic, which means a system has no choice (work exactly as programmed) (Clough, 2002). A more suitable definition of an autonomous robot is therefore: a machine that makes logical decisions and carries out work without human interaction, based on the way its program evaluates the information that is processed from its surroundings.

The development of autonomous mobile robots has made it possible to carry out work that would be dangerous, ineffective or too expensive if executed by humans. Autonomous mobile robots do not have the same understanding as humans but they have an ability to share detailed information. This information can be applied to overcome barriers faced by for instance fire fighters to develop better solutions faster. In the case of firefighters, a team of autonomous quadcopters could display the environmental information to the firemen over a display. This would navigate the firefighters which would result in better solutions and faster work (Schechter, 2014).

The 2011 nuclear disaster in Japan underlines the importance to research autonomous mobile robots. On March 11, 2011, the earthquake in the northern part of Japan recorded magnitude 9.0 and immediately caused damage to the nuclear reactors of Fukushima Daiichi nuclear power plant. Furthermore, the incident triggered the leakage of radioactive- and chemical material (World Nuclear Association, 2016). An autonomous mobile robot is useful in a situation like this to map the place and record detailed information. This information can be used by scientists and engineers to minimize the leakage without being exposed to nuclear radiation. These limits are the reason why autonomous mobile robots are increasingly being employed in applications such as mining, surveillance, search and rescue and hazardous site inspection. These applications require robots to travel across unprepared and rugged terrain. The conditions of an environment effect a mobile robot's mobility. Unknown environments require robots to rely on on-board sensors for navigation and control. These sensors will have uncertainty and measurement errors (Iagnemma & Dubowsky, 2004). The effects of unknown terrain and sensor errors make the motion planning and navigation challenging.

It can be argued that motion is the most important aspect of the development of these machines. The execution of path planning in terms of distance and computational time determines the capability of a robot (Devaurs, Simeon, & Cortes, 2015). Pathfinding is the route a robot must follow to move from point A to point B. On the other hand, motion planning is what makes the robot move that distance. A good interaction between the control system and sensors is required to successfully change the robot's motion accordingly to the environment. Pathfinding and motion planning are two sides of the same coin and one is incomplete without the other.

The project is divided into three subprojects: mapping, image processing and motion planning. This report deals with the motion planning, which includes the path planning, the physical system and the control of the physical system. Project in this report refers to motion planning unless otherwise stated.

1.2 Aims & Objectives

The aim of this project is to develop a program which enables the Dr. Jaguar 4x4 Wheel to follow a minimum cost path between two points autonomously avoiding obstacles in a dynamic unknown environment. To accomplish the aim, it is important to meet the following objectives:

1. To demonstrate and implement an efficient path planning algorithm for unknown dynamic environments in a two-dimensional state space.
2. Research and understand mathematical motion models that can be used to simulate the robotic platform used in this project.
3. Research and understand factors that are important for developing an all-terrain robotic platform and steps needed to develop an adaptive control systems with the ability to adjust tuning parameters as the robot moves between different unknown environments.
4. Develop a program for autonomous navigation in MATLAB/Simulink and learn how the path planning algorithm and control strategy can be used to control the rover over the planned path.
5. Ensure the program works by simulating it in different scenarios. Integrate the program with other team members' projects and implement the combined program in the actual rover.

1.3 Project Approach

All work was done using MATLAB and Simulink. The latest version (10) of the robotic toolbox developed by Peter Corke and Robotics System Toolbox were downloaded and installed on MATLAB. The project was supervised by Dr. Mike Jump at the University of Liverpool and a lot of help was received from both him and Dr. Paolo Paoletti. The Gantt chart proposed at the beginning of the project is shown in Appendix III.

2.0 Literature Review

The project was broken down into 3 sections: path planning, physical system and control system. This chapter presents previous research in these areas as a summary from various research papers used prior to beginning and throughout the project work. Each section describes multiple techniques and algorithms that can be implemented in this project.

2.1 Path Planning

The aim of a path planning algorithm is to find the minimum cost path that connects the starting position of the rover to the goal while avoiding obstacles. To determine the minimum cost path physical distance, time delay and energy consumption must be considered. Path planning can be classified into two categories; reactive navigation and map-based navigation. Reactive navigation does not include a map and the robot is guided towards a goal by for instance following a light on the ground or moving through a maze by following a wall. This type of navigation is used in small tasks such as vacuum cleaners. Map based path planning is more complex and is used for more advanced tasks such as emergency work (Corke, 2011). Three methods were understood and learnt in detail, each one was based on map based path planning.

The virtual force field method uses the general idea behind a ball rolling down the hill. The desired destination would be a low potential region whilst the obstacles would be high potential regions. Consequently, the robot should always move to a lower potential state. This is made possible by creating a virtual force field where the goal has an attractive potential and the obstacles have a repulsive potential. Applying mathematics to the field, a gradient can be obtained. The robot will just follow the gradient towards the goal (Kuipers, 2005). However, there unresolvable problems related to this method (Koren, Borenstein, & Arbor, 1991). The most severe problem is identified during robot movement in a narrow corridor (Borenstein & Koren, 1989). A robot moving on the centreline between two walls in a narrow corridor will experience an equal repulsive force from both walls causing an equilibrium and thereby a straight movement on the centreline. However, the robot will oscillate if the robot is slightly on either side of the centreline. This is due to a greater repulsive force from the closer wall and a decreased force from the wall farther away. In addition to this the robot will get stuck if it is on the centreline but the goal is near one of the walls in a corridor. As a matter of fact, the method does not guarantee tracking down all path points thus this method will not be implemented.

The A* algorithm is the most used technique for pathfinding (Choset, Carnegie Mellon University, 2007). The final path is determined only after analysing all possible paths. It uses a cost-effective method and guarantees the shortest path between two points. A map divided into nodes is required to execute the algorithm therefore it is usual that a grid based search is applied to run this algorithm. The drawback of the A* algorithm is that it must replan from scratch if the position of obstacles changes. The D* algorithm, also called Dynamic A*, is a solution to this disadvantage. It can replan faster than the A* because it modifies its previous search results locally (Choset, Carnegie Mellon University, 2007). The D* lite is another version of this algorithm. It is not based on the original D* algorithm but implements the same behaviour, however D* lite can be coded in fewer lines hence the name ‘lite’ (Choset, Carnegie Mellon University, 2007). It would be interesting to analyse how these algorithms evolved and what challenges need to be tackled to enhance the performance of autonomous mobile robots in future.

2.2 Motion Model

The differential drive is the most common mechanism used on robotic platforms. It is easy and highly manoeuvrable to control in indoor environments and therefore is suitable for beginners (Mondal, 2014). It has two electrically driven wheels on either side of the robot body. Thus, the rover direction

can be steered by varying the rotational speed of the wheels (Mandow, et al., 2007). Rotating the wheels at the same speed on each side causes it to move in a straight line, whilst rotating them at opposite speeds with equal magnitude causes the robot to rotate about the central point of the axis. The rover turns toward the side with less speed, e.g. it turns left when left side speed is less than right side speed and vice-versa (Mandow, et al., 2007). Additional free rotating wheels may be added to increase body balance. Skid-steered rovers use the same principle as differential drive rovers (Mandow, et al., 2007). However, at least two wheels on each side are driven independently by a power source. This result in wheel slippage while turning the rover. Skid-steered robots have high manoeuvrability which makes them suitable for all-terrain missions (Mandow, et al., 2007). Nevertheless, the most comprehensive wheeled robot platform on unknown terrain is the rocker-bogie suspension mechanism, which is currently the favoured design by NASA (Lee & Miller, 1998). The mechanism incorporates three wheels on both sides that are each driven by an independent power source (Lee & Miller, 1998). It is unique because it can climb over obstacles that are twice the diameter of the wheel while all wheels stay in contact with the ground (Lee & Miller, 1998). All the mechanisms mentioned above share the ability to perform a zero-radius turn making them an attractive choice in the robotic society.

FIGURE 1 THE ROBOTIC PLATFORM DR. JAGUAR 4X4 WHEEL

The Jaguar 4x4-wheel mobile robotic platform is used in this project. It has two wheels on each side of the robot body and is driven by four powerful motors, one for each wheel thus classified as a skid-steered vehicle. The robotic platform is strong yet simple, designed for rough conditions. It weighs 20 kg and can carry an additional weight of 30 kg or drag a weight of 50 kg on a flat surface (Dr Robot, 2014). The robotic platform is also capable of running over a vertical step of 155 mm and climbing stairs up to 110 mm step (Dr Robot, 2014). The maximum speed goes up to 3 m/s with a ground clearance of 88 mm (Dr Robot, 2014). It integrates outdoor GPS and DOF IMO for autonomous navigation. Furthermore, the platform is compact, weather and water resistant, and is therefore suitable for all-terrain missions.

A skid-steered rover is simple to drive and easy to control in tight space. However, the drawbacks include poor odometry due to slippage, and large energy usage while turning (Solc & Sembera, 2008). A skid-steered vehicle experiences both lateral and longitudinal slip. Consequently, the wheel slip is critical because it provides a connection between the wheel rotation velocity and linear motion of the rover platform. A kinematic model is easy to implement but it is based on pure rolling and no slip assumptions. It can still be used represent a skid-steered vehicle by determining slip ratios by comparing the rotational velocity of the wheels and the linear motion of the robotic platform (Solc & Sembera, 2008). However, this must be done experimentally and limits the slip values to the tested surface. Therefore, dynamic modelling must be opted for to develop a software for autonomous navigation of mobile rovers in unknown terrain. The main difference between a kinematic- and dynamic model is the modelling of the tire (Amtoft & Jensen, 2011). Two tyre models that calculate lateral and longitudinal slipping using different methods are discussed below.

2.2.1 Model A

The basic idea of slip calculation for model A is based on calculating the adhesion force in the direction of tyre motion. Dispersive- and electrostatic adhesion are two of the five forms of the adhesion force

and can be applied to dynamic contact between tyres and the surface (Raymond, 2006). Electrostatic adhesion is where materials pass electrons to form a difference in electrical charge at the contact area (Raymond, 2006). This creates an attractive electrostatic force between materials. It is known that tyres can gain electrostatic charges, but Kummer assumes electrostatic adhesion to be negligible (Kummer, 1996). The reason being is that electrostatic force is sensitive in humidity which is an insignificant factor for tyre friction (Raymond, 2006). This leaves dispersive adhesion, which is due to two materials being held together by van der Waals force. A van der Waals force is an attraction caused by opposite charged molecules. However, dispersive adhesion is negligible as well because the gravity force is dominant. This is because dispersive adhesion is proportional to diameter d, whilst gravity is proportional to diameter d^3 for objects in contact (Raymond, 2006). The dispersive force can however be significant in comparison to gravity for microparticles. Since the surface roughness reduces the dispersive force drastically even on the micro-level, the concept of adhesion is complex and is unlikely to have a significant role in the development of tyre-pavement traction forces (Raymond, 2006). Although, the resistance to sliding can be described as a constant frictional force at the contact surface. This is known as the Coulomb model of friction. It predicts the direction and magnitude of friction force between two bodies with dry surface in contact. Assumed static and kinetic friction are both calculated prior to predicting the case (static or kinetic) that arises (Raymond, 2006).

2.2.2 Model B

The Pacejka's magic wheel formula is called Model B. It calculates longitudinal- and lateral slip force based on data gathered from experimental tests on real tyres. The acquired data was used to develop the magic equation. It can predict the tyre behaviour with great precision (Edy, n.d.). Longitudinal slip is calculated in percentage considering the wheel speeds and longitudinal vehicle speed, while lateral slip is calculated in radians considering lateral vehicle speed and wheel speeds. The longitudinal force and lateral force is calculated using the calculated slip based on B, C, D, E (Edy, n.d.). These constants have physically no meaning, but are obtained by performing tests on real tires.

2.2.3 Tests

Model A and Model B have been tested and compared by Pompa and Suarez at the Polytechnic University of Milan in 2016 (Pompa & Suarez, 2016). Model B proved to be more realistic as expected since it considers forces present for small slip angles, something Model A fails to predict. Despite this, the results have a small absolute error and Model A was therefore verified for a skid-steered robot. Overall, Model A is simple to implement and fairly accurate empirically for robot platforms, while Model B is complex and preferred for larger vehicles. A motion model using tyre Model A has been designed earlier by (Solc & Sembera, 2008), (Amtoft & Jensen, 2011) and (Adam, 2016).

2.3 Control Strategy

The mobility of the rover as a function is the crux; it is the essential capability for complex achievement. The control strategy is what executes the planned path. It can be divided into two sections: path tracking and motion trigger. The main purpose of path tracking is to divide the path into temporary targets and chase them until the rover reaches the final goal. Path tracking connects the planned path to the motion trigger which generates wheel velocities. Designing a control for skid steered vehicles that can provide all terrain manoeuvrability in an unknown environment is a challenge. In fact, it is a need to work towards adaptive control laws in the future, with the ability to autonomously tune parameters based on characteristics such as wheel rotations, turn rates and throttle values (Clarke & Blanchard, 2010). This would allow autonomous vehicles to move between different unknown surfaces while maximising their performance. This principle of adaptive control law can be applied to optimise turning abilities and path following accuracy of a rover on indoor floors as the surface characteristics do not change much. The path following accuracy could significantly improve because highly tuned

parameters increases manoeuvrability but reduces motion smoothness, and vice-versa for parameters with low values. A common problem linked to control strategy is mentioned in the research paper by (Clarke & Blanchard, 2010). The motion trigger consists of a throttle and steering. The DC motors take time to adjust and therefore quick throttle movements could have little effect on the rover movement. Therefore, correct DC motor tuning and a control design that takes into account the overshoot of the rover due to time needed for DC motors to slow down during navigation process is vital to develop a precise and accurate control strategy.

A simple yet effective algorithm for path tracking is called follow-the-carrot (Lundgren, 2003). A perpendicular line is drawn from the current vehicle position towards the next target coordinate. The vehicle is steered towards the target coordinate by the motion trigger. The velocity induced by the motion trigger is proportional to the distance between the current and next point. The vehicle velocity is the highest during the beginning of the movement between these points, and is reduced to zero upon reaching the target coordinate until a next target coordinate is set. A drawback of follow-the-carrot is that the vehicle could oscillate about the path if it misses the target coordinate, which is likely to happen during small distances between two points if the speed is high. To fix this problem an error radius is usually set (Roy, 2016). The vehicle then turns towards the next point upon reaching the space within the set error radius. This solution results in the vehicle cutting corners, which is another disadvantage (Lundgren, 2003). However, setting a small error radius and controlling the acceleration of the vehicle, combined with an adaptive control system can fix the problem.

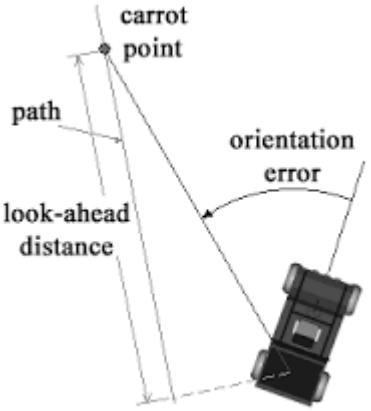


FIGURE 2 FOLLOW-THE-CARROT DIAGRAM

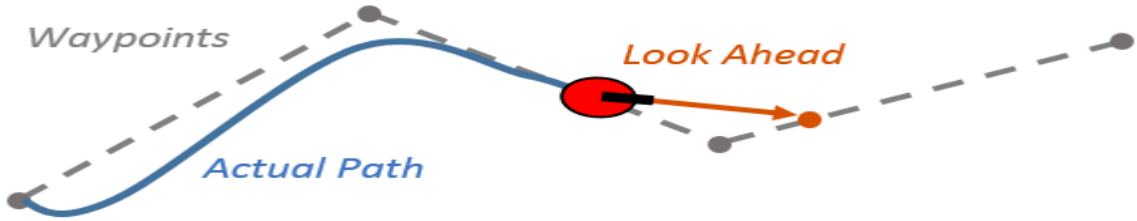


FIGURE 3 LOOK AHEAD

The pure pursuit method is another algorithm that tracks a path using temporary target points in the same way as follow-the-carrot algorithm. It is often compared to how humans look ahead and focus on a certain point while driving a car (Lundgren, 2003). The car movement between the current position and the focus point is not perpendicular but curved. Since the vehicle is moved towards the goal with the calculated curvature, the motion is more robust than follow-the-carrot method. Other path tracking algorithms may be optimised in other manoeuvrability areas, but this is overall yet the best available algorithm available today because it results in less oscillation and a smoother path track (Lundgren, 2003). The CF pursuit is a new version of the pure pursuit method introduced by Shan (Samuel, Hussein, & Mohamad, 2016). In their method, the curvature is replaced by clothiod[2] C curve to reduce fitting errors. A fuzzy logic is used to consider the best fitting curvature. It is developed based on real world tests of human drivers as well as their own research tests. It has been proven to be an efficient path tracking algorithm for self-driving cars (Samuel, Hussein, & Mohamad, 2016).

[2] A plane curve whose curvature at any point is proportional to distance along it. [http://www.engyes.com/en/dic-content/clothiod]

The vector pursuit is another example of a geometric path tracking algorithm that works well for vehicles with skid-steered mechanism (Samuel, Hussein, & Mohamad, 2016). The vector pursuit uses the same technique of having temporary targets as pure pursuit- and follow-the-carrot algorithm. The difference is how the look-ahead distance is used to determine the desired motion of the vehicle. In addition to looking ahead at the location of the target coordinate, the vector pursuit considers the desired orientation the vehicle should have upon reaching the target point. The vector pursuit is based on the screw theory basics developed by Sir Robert Stawell Ball (Lundgren, 2003). A screw is used to explain the instantaneous movement of a relative body about a given coordinate system. The velocity of the rigid body depends on the velocity due to rotation and the translation velocity caused by the pitch of the screw (Wit J. S., 2000). The rotation and translation velocities are both called instantaneous screws (Wit, Crane, & Armstrong). The first screw considers the translation from the current vehicle position to the look-ahead target point, while the second screw considers the rotation from the current vehicle orientation to the desired orientation at the target point (Lundgren, 2003). These two screws are added together to form a path on which the vehicle is steered.

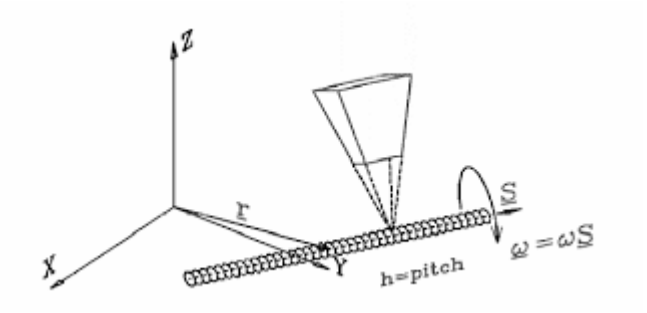


FIGURE 4 SCREW THEORY FOR VECTOR PURSUIT ALGORITHM

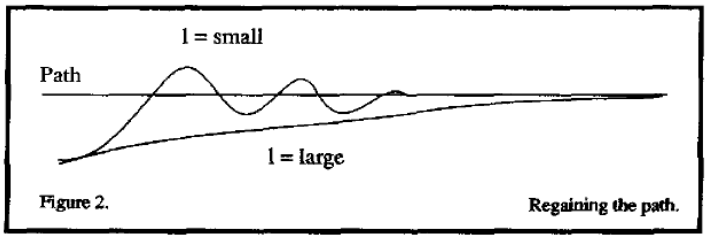


FIGURE 5 REGAINING A PATH

Common to all the path tracking methods mentioned above is the use of look-ahead distance. There are three factors that needs to be considered before choosing an algorithm; i) regaining a path, ii) maintaining a path and iii) vehicle speed (Lundgren, 2003). A large value of the look-ahead distance causes the vehicle to converge smoothly with less oscillation onto the required path, whilst a small value required to keep the vehicle running smoothly on the path (Lundgren, 2003). It is important to notice that the look-ahead distance is equivalent to the controlled vehicle speed, because a higher controlled speed requires the vehicle to turn at an earlier stage (Clarke & Blanchard, 2010).

2.4 Closure

It was decided to implement the D* algorithm for path planning, model A for the physical system and follow-the-carrot as the core principle for the control strategy. The D* algorithm is a more advanced and a common path planning method used for autonomous navigation of rovers in unknown environments. The D* lite is a shorter code than the D* and performs equally well. Despite that, the D* was chosen because it was already developed and available in the robotic toolbox by Peter Corke. Therefore, this was the best choice because it saved time in addition to being the preferred algorithm. It was decided to use grid based mapping on all subprojects, because the D* uses a grid based approach to plan a path. Model A was used for representing the physical system because the results were similar to Model B and it was easier verify model A as it could be compared against previously recreated models. Lastly, the control strategy was built based on the follow-the-carrot method. This method was easy to understand and simple to implement. It could be enhanced and possibly match the performance of the best choice which would be the pure pursuit method.

3.0 Theory

The theory consists of three sections: path planner, physical system and the controller. The path planner section explains how the D* algorithm works. The DC motors and the Dr. Jaguar 4x4 make the physical system. The controller connects the path planner and the physical system. It uses the path information to guide the physical system along the planned path by the D* algorithm.

3.1 Path Planning

Each node is 1 m^2 where the length (x) and width (y) are 1 meter each. Hence the diagonal distance (d) can be defined using Pythagoras theorem d = $\sqrt{x^2 + y^2} \approx$ 1.4 m^2.

Initially, the D* algorithm plans a path using the best first approach. The h values represent the shortest path cost (distance) from the goal. Nodes that are being evaluated are placed on the OPEN list. The nodes are ranked based on k values; smaller k values are given priority. The k value of a node identifies the node either as a RAISE or a LOWER state. A RAISE state is when k<h, which is an indication of increased path cost (Choset, Carnegie Mellon University, 2007). A LOWER state is when k≥ h and indicates that there may have been a path cost reduction (Choset, Carnegie Mellon University, 2007). Below is a sample map that will be used to explain the D* algorithm during movement of a robot where the yellow node represents a gate. The gate is initially assumed as free path but the robot will realise the gate is too small to pass through on reaching closer to it and therefore will reconsider the gate as an obstacle.

FIGURE 6 DSTAR AREA MAP (LEFT) AND BEGINNING OF THE PATH SEARCH (RIGHT)

Since all nodes are being evaluated for the first time in the beginning, therefore the k and h values are the same. For obstacles, the k and h values are set to a high number, because ultimately the values work as a gradient. This basically means that the rover would move towards a node with a lower gradient.

The nodes up until the start are expanded before the starting node is expanded itself (shown on figure 6 right part). The start is the last one to be expanded. Nodes with red crosses have been evaluated.

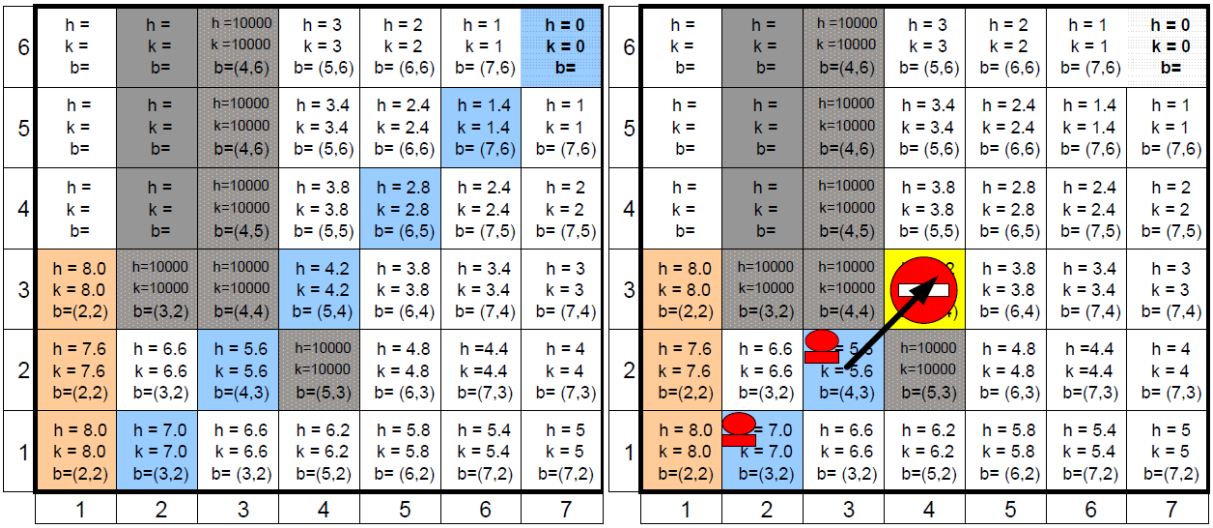


FIGURE 7 INITIALLY PLANNED PATH (LEFT) AND DETECTION OF AN OBSTACLE ON THE PLANNED PATH (RIGHT)

The blue nodes form the final path. The route is towards the lower h value, basically from high to low. Therefore, the high h values ensure there is no movement towards the obstacles. On reaching point (3.2) (shown on figure 7 right part) the rover realises the gate is an obstacle and therefore a new path must be computed.

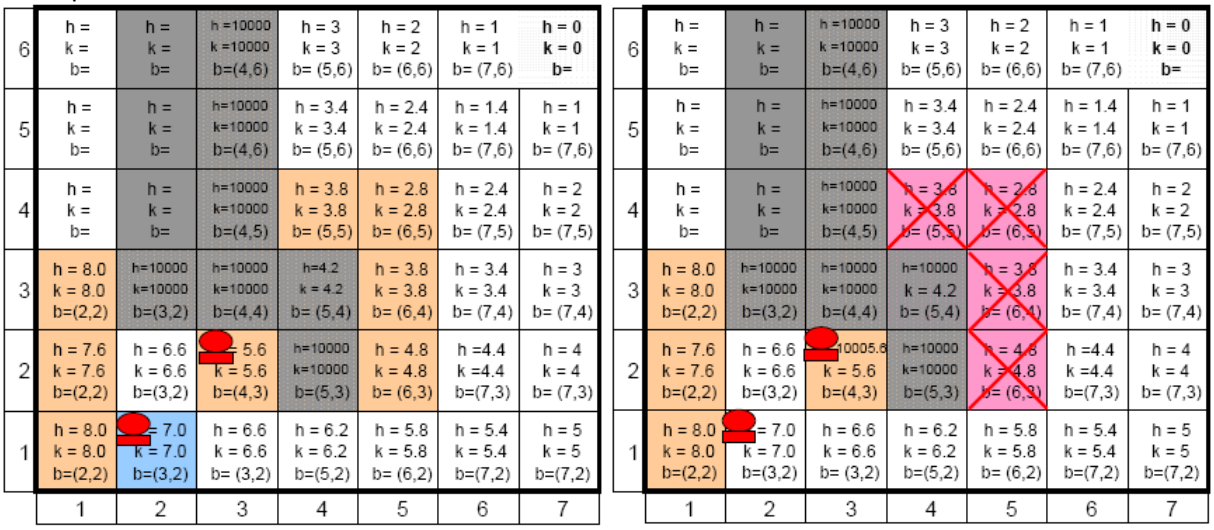


FIGURE 8 RE-PLANNING THE PATH TOWARDS GOAL STEP 1 AND STEP 2

First, all neighbours of (4.3) are added on the OPEN list. The nodes with the smaller k values are prioritized first. The h value is now updated to a high number for node (4.3). The pink nodes are crossed out and no other nodes are affected, because there are no new pixels in the surroundings and the h values are correct. Through the current path node (3.2) would have a h value of 10005.6, all high numbers are assumed to be equal, therefore these are set to 10000.

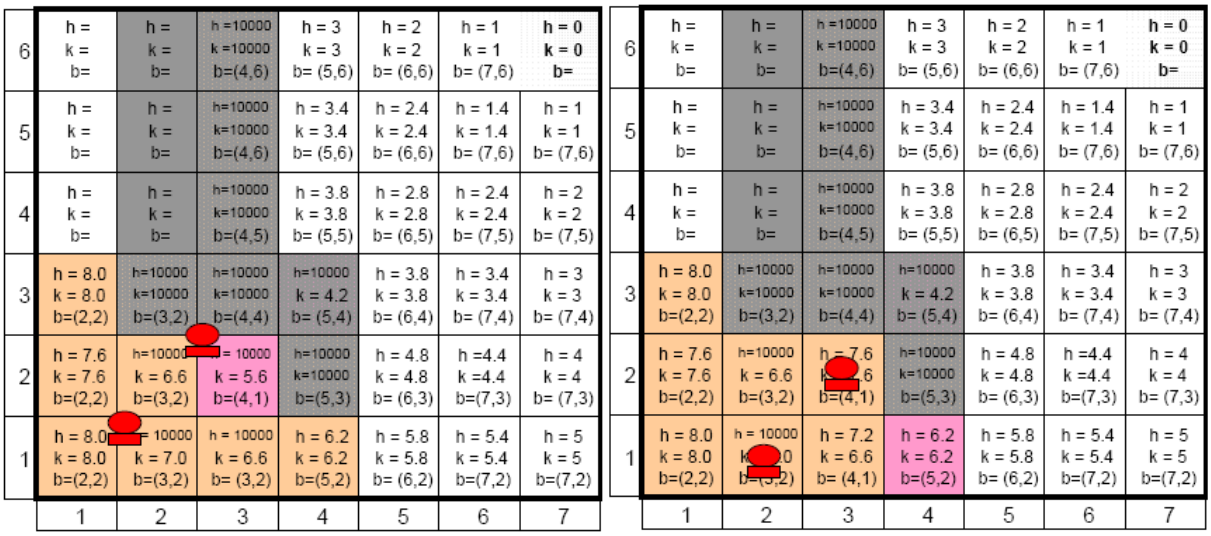


FIGURE 9 RE-PLANNING THE PATH TOWARDS GOAL STEP 3 AND STEP 4

(3.2) is now a RAISE state. No neighbours have a lower k value than this node and hence this node is given priority. (4.1) has a lower h value than (3.2) and therefore could lead to a lower path cost. (4.1) is expanded and it updates the h values of (3.2) and (3.1). (3.1) cannot reduce the cost path because it is a RAISE state.

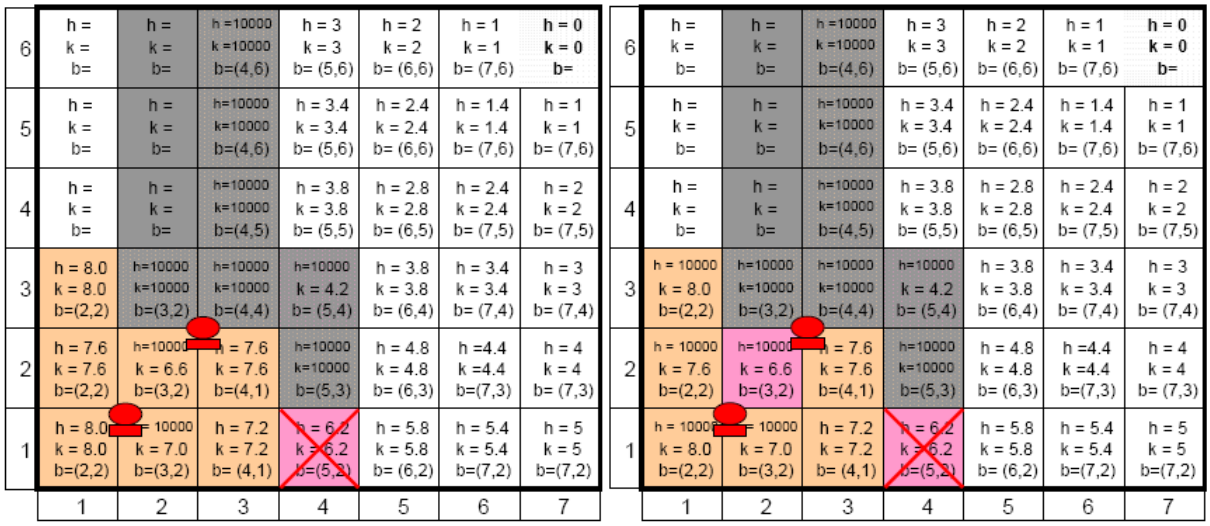


FIGURE 10 RE-PLANNING THE PATH TOWARDS GOAL STEP 5 AND STEP 6

However, (3.1) can be used to form a reduced cost path for its neighbours. The k values are set equal to the minimum of the new and old h value. Thus, the node is in a LOWER state.

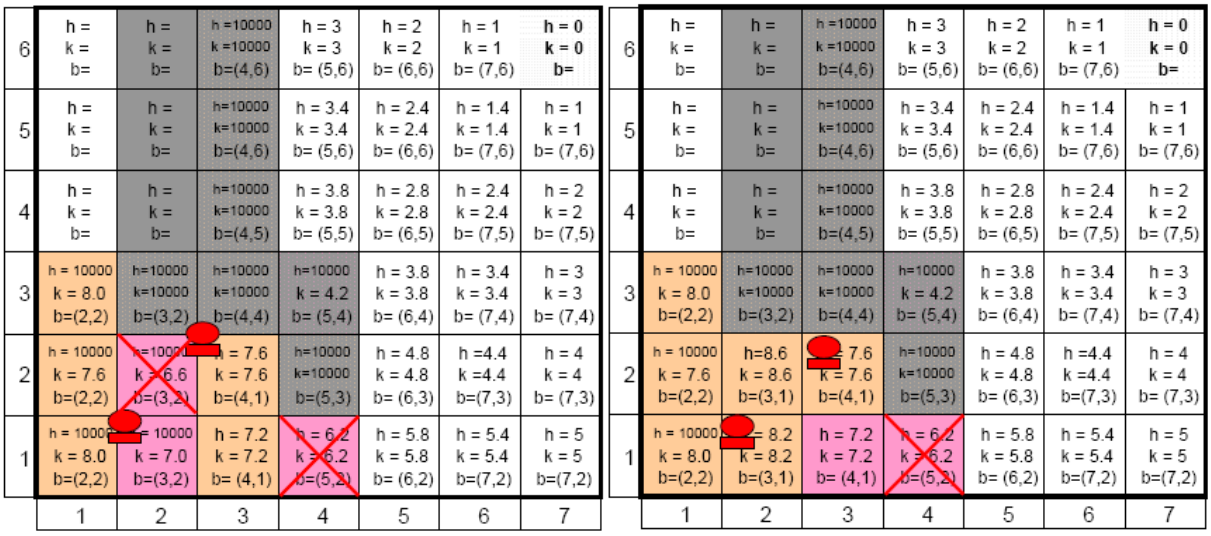


FIGURE 11 RE-PLANNING THE PATH TOWARDS GOAL STEP 7 AND STEP 8

Node (2.2) and (2.1) both have k<h and is therefore a RAISE state, therefore it cannot reduce the cost of any of its neighbours. Therefore, these nodes have no effect and are updated to h==k.

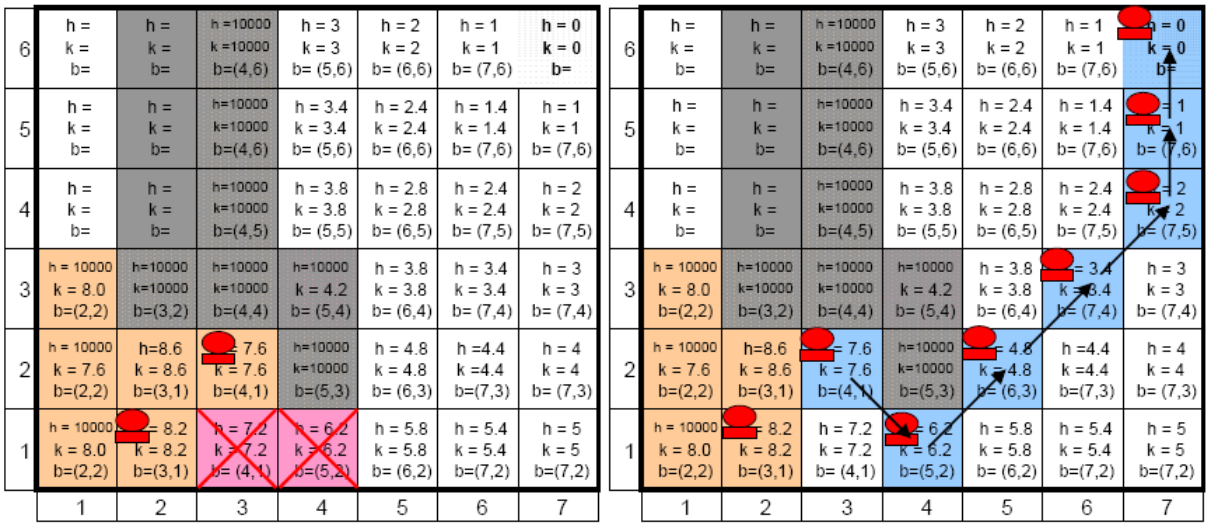


FIGURE 12 RE-PLANNING THE PATH TOWARDS GOAL STEP 9 (LEFT) AND ROVER MOVEMENT ON RE-PLANNED PATH (RIGHT)

The search ends here because no node has a lower h value than the current node the rover is placed on. The rover moves towards a node with a lower gradient until it reaches the goal as shown on figure 12 right part.

3.2 Skid Steered Motion Model

The skid steered rover is driven by four powerful motors, one connected to each wheel. The vehicle body is represented using a dynamic model. The following assumptions are considered:

- Each side's two wheels rotate at the same speed (Mafrica, 2012).
- There is always contact between the wheels and the ground (Mafrica, 2012).
- The rolling resistant force is negligible (Mafrica, 2012).
- The centre of mass of the rover is located at the geometric centre of the body (Mafrica, 2012).
- The normal forces are equally distributed among four wheels during motion (Mafrica, 2012).

- The rover is running on a flat surface (Mafrica, 2012).

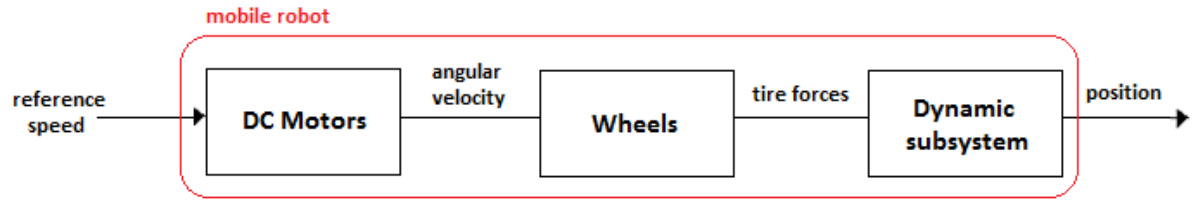


FIGURE 13 UPPER LEVEL BLOCK DIAGRAM OF THE PHYSICAL SYSTEM (SKID STEERED MOTION MODEL)

The wheels and dynamic subsystem is recreated from the research paper by (Solc & Sembera, 2008).

3.2.1 DC Motors

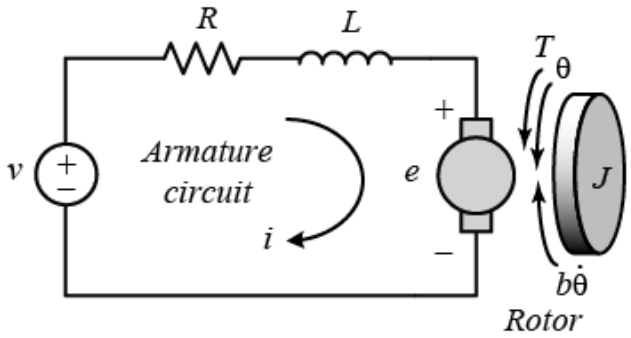


FIGURE 14 ELECTRIC CIRCUIT OF THE ARMATURE

R represents the resistance of the coil, L is the inductance of the coil, T is the motor torque and v is the motor voltage. It is assumed the magnetic field is fixed. Therefore, the torque is proportional to armature current by the torque constant K_t.

$$T = Ki \tag{1}$$

This is therefore an armature-controlled motor. The back emf e is proportional to angular velocity of the shaft by a constant factor K_e. The motor torque and back emf are equal, $K_t = K_e$ and is therefore represented as K throughout the paper.

$$e = K\omega = K\frac{\partial\theta}{\partial t} \tag{2}$$

Equations 3 and 4 are obtained by applying Newton's and Kirchhoff's law to the figure above.

$$J\frac{\partial^2\theta}{\partial t} + B\frac{\partial\theta}{\partial t} = Ki \tag{3}$$

$$L + B\frac{\partial i}{\partial t} + Ri = V - K\frac{\partial\theta}{\partial t} \tag{4}$$

By applying Laplace transformation to (3) and (4), they can be written as shown below.

$$Js^2\theta(s) + Bs\theta(s) = KI(s) \tag{5}$$

$$LsI(s) + RI(s) = V(s) - Ks\theta(s) \tag{6}$$

Solving 5 and 6 gives

$$Ga(s) = \frac{\theta(s)}{V(s)} = \frac{K}{s((R+Ls)(Js+B)+K^2)} \tag{7}$$

In terms of velocity it can be represented as

$$Gv(s) = \frac{\omega(s)}{V(s)} = \frac{K}{(R + Ls)(Js + B) + K^2} \tag{8}$$

This equation was used to draw a block diagram to build the model in Simulink. Furthermore, a PID feedback controller was created to control the velocity of the DC motor[3]. The following equation is used for modelling the speed controller:

$$C(s) = \frac{U(s)}{E(s)} = \frac{K_d s^2 + K_p s + K_i}{s} \tag{9}$$

3.2.2 Wheel Speeds

Four equations are needed to describe the longitudinal velocities of each wheel. These are reduced to two equations by assuming that each side's two wheels rotate at the same speed. Similarly, four equations are needed to describe the lateral velocities of each wheel, but these are reduced to two equations by assuming the lateral frontal velocities to be the same and the lateral rear velocities to be the same.

The following four equations can be derived from the free body diagram that describe the different velocities of each wheel:

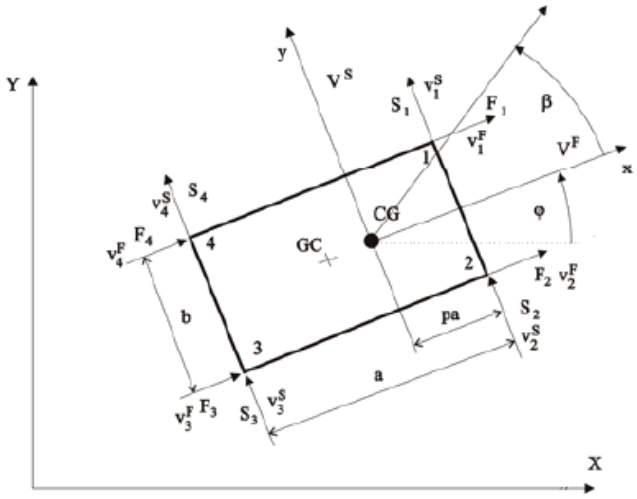


FIGURE 15 FREE BODY DIAGRAM OF THE ROBOTIC PLATFORM DESCRIBING THE VELOCITIES AND TRIGONOMETRIC VEHICLE MOVEMENT

$$v_1^F = v_4^F = \dot{x}\cos\varphi + \dot{y}\sin\varphi - \frac{b}{2}\dot{\varphi} \tag{10}$$

$$v_2^F = v_3^F = \dot{x}\cos\varphi + \dot{y}\sin\varphi + \frac{b}{2}\dot{\varphi} \tag{11}$$

$$v_1^S = v_2^S = -\dot{x}\sin\varphi + \dot{y}\cos\varphi + pa\dot{\varphi} \tag{12}$$

$$v_3^S = v_4^S = -\dot{x}\sin\varphi + \dot{y}\cos\varphi - (1-p)a\dot{\varphi} \tag{13}$$

These equations describe the trigonometric relationship of the whole vehicle movement and the velocities of each wheel. The first two equations are used to obtain the longitudinal velocities of left

[3] http://ctms.engin.umich.edu/CTMS/index.php?example=MotorSpeed&section=SystemModeling Modelling of a DC motor tutorial.

and right wheels respectively. The last two equations are used to obtain the lateral velocities of front and rear wheels respectively.

3.2.2 Tire Forces

The wheel model is based on the Coulomb model of friction. The basic idea is to calculate the force of adhesion in the direction of the tyre motion. Hence, the lateral and longitudinal forces components can be obtained. F is the longitudinal force while S is the lateral force.

$$F = P\frac{v_s}{v_f} \tag{14}$$

$$S = -P\frac{v^S}{v_f} \tag{15}$$

v_s is the longitudinal velocity and can be expressed as:

$$v_s = v - v^F \tag{16}$$

v_f is the total slipping velocities including the lateral slip velocity, and can be expressed as:

$$v_f = \sqrt{v_s^2 + (V^S)^2} \tag{17}$$

Using Coulomb model of friction yields

$$P = Cv_f \text{ for } v_f \leq \frac{\mu_0 W}{C} \tag{18}$$

$$P = \mu_0 W \text{ for } v_f > \frac{\mu_0 W}{C} \tag{19}$$

where C is the tire stiffness and μ_0 is coefficient of friction.

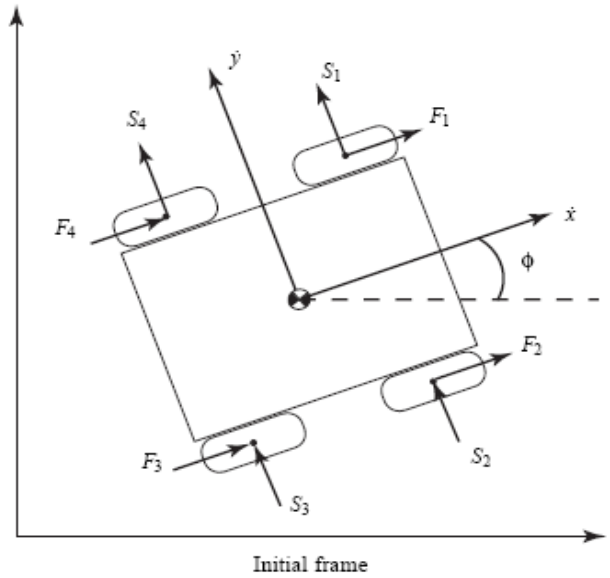


FIGURE 16 FREE BODY DIAGRAM SHOWING THE LONGITUDINAL AND LATERAL FORCES ON THE TYRES

3.2.3 Dynamic Subsystem

The dynamic subsystem is needed for a skid steered vehicle because slipping and skidding is involved. Slipping and skidding can only be calculated if forces and masses are known. Using Newton's second law to describe the movement about the centre of gravity of the vehicle gives equation (20) and (21).

$$M\ddot{x} = \sum_{i=1}^{i=4} F\cos\varphi - \sum_{i=1}^{i=4} S\sin\varphi \tag{20}$$

$$M\ddot{y} = \sum_{i=1}^{i=4} F\sin\varphi + \sum_{i=1}^{i=4} S\cos\varphi \tag{21}$$

$$J\ddot{\varphi} = (-F_1 + F_2 + F_3 - F_4)\frac{b}{2} + (S_1 + S_2)pa - (S_3 + S_4)(1-p)a \tag{22}$$

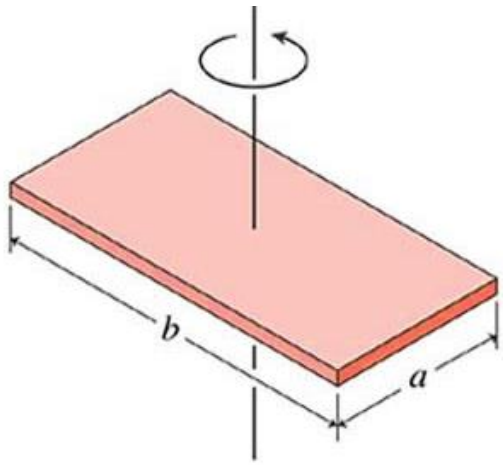


FIGURE 17 RECTANGULAR FRAME

The vehicle acceleration along each direction is equal to the force acting on the vehicle in that direction, divided by the mass. The rotational force in equation (22) is calculated taking the moments around centre of gravity, the first term is the longitudinal force, the second term is the lateral force of front wheels and the last term is the lateral force of rear wheels. The moment of inertia for a rectangular robotic platform can be estimated by the following equation:

$$I = \frac{1}{12} M(length^2 + width^2) \tag{23}$$

The accelerations can be calculated if the longitudinal and lateral forces of each wheel are known. Integrating the accelerations would give the velocity and final position of the vehicle.

3.3 Control Strategy

This is an overview over how the control strategy works. If the rover is moving towards a waypoint and the heading difference is greater than the defined limit in heading error, the rover performs a zero-skid radius and aligns itself with the correct heading angle. If the rover is within the set radius of a waypoint, the next waypoint is considered. The navigation task is finished when there are no more waypoints left.

A detailed description of the theory is given in this subchapter. The control strategy is divided into trajectory controller, motion controller and proportional velocity command reduction. The motion controller can further be divided into a cruise control, heading control and a move to pose control.

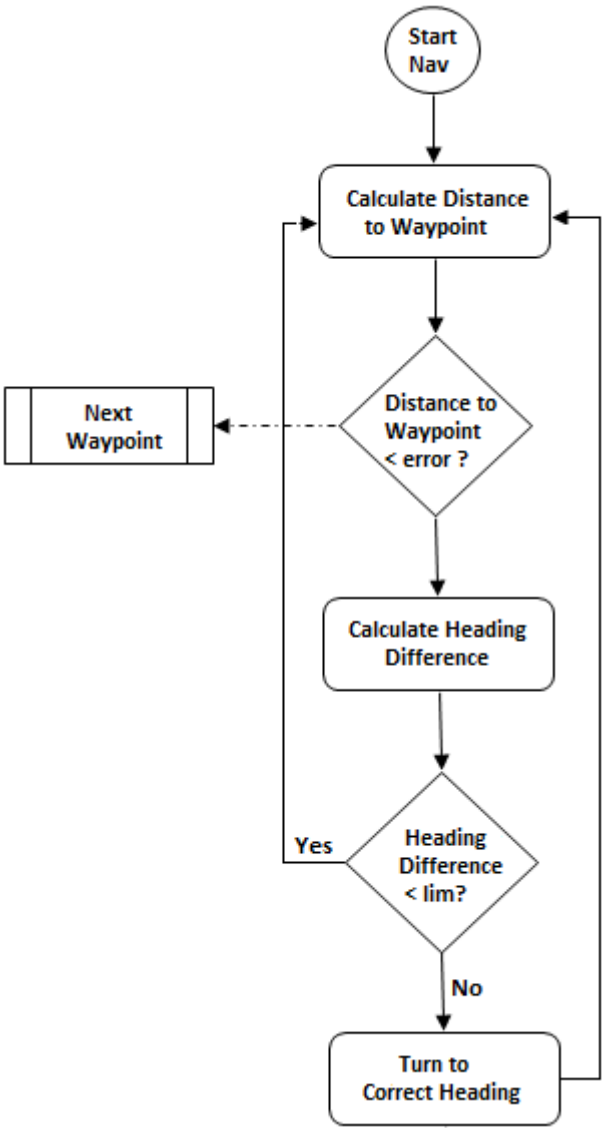


FIGURE 18 FLOW-CHART OVERVIEW OF CONTROL STRATEGY

3.3.1 Trajectory Controller

The trajectory controller takes the current rover position and a temporary target position, and calculates the distance between these positions. When the distance (D) difference is zero, next waypoint (temporary target position) is automatically set.

$$D = \sqrt{(x_1 - x)^2 + (y_1 - y)^2} \tag{24}$$

3.3.2 Motion Controller

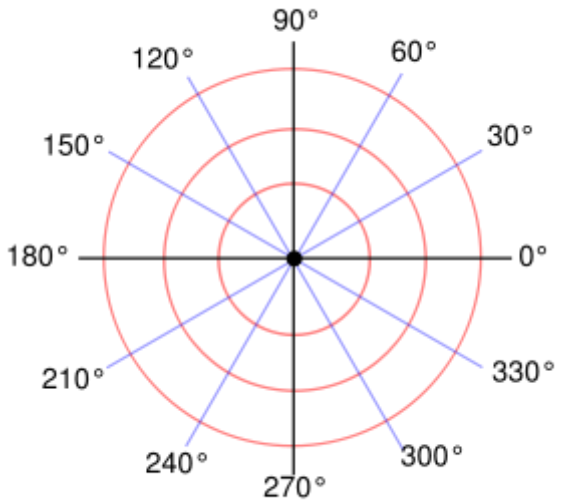


FIGURE 19 ANGLES DEFINED FROM THE ORIGIN OF A COORDINATE SYSTEM

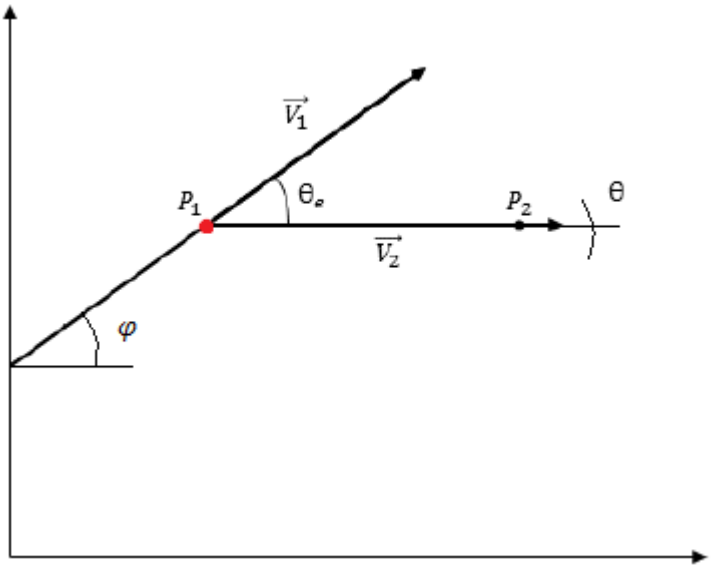


FIGURE 20 SHOWING THE ROVER (RED DOT) CHANGING COURSE

The angles shown in figure 19 go from 0 to 360 degrees, but the controller limits the angle range from -180 to 180 degrees. Therefore, the angles re-start from zero but in negative direction (0 to -180).

3.3.2.1 Cruise Control

Cruise control considers the rover moving to a point. It controls speed- and steering command (sets the required angle to change heading). The velocity of the rover is controlled to be proportional to its distance from goal when moving towards a goal (x1, y1) in the plane:

$$v = K_v\sqrt{(x_1 - x)^2 + (y_1 - y)^2} * reached \tag{25}$$

Reached is equal to 1 while the navigation is carried out, it changes to 0 on reaching the final goal. The red dot on figure represents the rover. The angle, φ, is the current heading angle of the rover. The angle, θ, is the required angle to steer the rover towards position, P_2. The error angle, θ_e, is the difference between the required and current heading angle. If the error angle is zero, the rover will move in a straight line relative to its current heading angle. In other words, equal speeds on both sides' causes the rover to move in a straight line.

$$\theta = \tan^{-1}\frac{y_2 - y_1}{x_2 - x_1} \tag{26}$$

3.3.2.2 Heading Control

The part of the controller that turns right or left is called the heading control. If the difference between the required angle to change course and current heading angle is positive then the controller steers it towards left, if the angle difference is negative then it is steered towards right. The rover turns towards the side which has the lower velocity. The higher the velocity difference is between the right and the left side, the larger is the skid radius (Amtoft & Jensen, 2011). The angle difference is limited to π and $-\pi$.

$$\theta_e = (\theta - \varphi) * reached \tag{27}$$

$$v_L = v - K_r\theta_e \tag{28}$$

$$v_R = v + K_r\theta_e \tag{29}$$

3.3.3 Move to Pose

The rover will be aligned relative to the path angle when it reaches the goal. A desired pose different to the path angle can be achieved by a zero-radius skid on the spot. This is achieved by generating opposite speeds of same magnitude on left and right wheels. If reached is equal to 1 the move to pose control will remain inactive, the moment reached is 0 the cruise control will shut down and move to pose control be activated. The equations below show how this is achieved.

$$v_{left} = v_L + v_{L2} \tag{30}$$

$$v_{right} = v_R + v_{R2} \tag{31}$$

where

$$v_{L2} = -z(reached - 1)(\theta_{orient} - \varphi) \tag{32}$$

$$v_{R2} = z(reached - 1)(\theta_{orient} - \varphi) \tag{33}$$

3.3.4 Proportional Velocity Command Reduction

The purpose of the PVCR is to increase motion smoothness especially during tight turns (Hieema, 2013). The triggered velocity is high when the next waypoint is received since the velocity is proportional to the distance between current- and next coordinate. Hence, the rover overshoots and motion smoothness is decreased. This is fixed by proportionally accelerating the speed of both sides at a slow pace.

The PVCR takes the absolute velocity of the controlled speed for left and right wheels. If the total velocity is higher than a defined limit, the controlled speed of the wheels is accelerated proportionally. However, if the total velocity is less or equal to the limit the modified velocity remains the same as the controlled speed.

$$v_{total} = \frac{\left|v_{left} + v_{right}\right|}{2} \tag{34}$$

$$\begin{aligned} mod\_v_{left} &= v_{left} \\ mod\_v_{right} &= v_{right} \end{aligned} \quad \text{for } v_{lim} >= v_{total} \tag{35}$$

$$mod\_v_{left} = v_{left}\sqrt{\frac{v_{lim}}{v_{total}}} \qquad \text{for } v_{lim} < v_{total} \tag{36}$$

$$mod\_v_{right} = v_{right}\sqrt{\frac{v_{lim}}{v_{total}}}$$

4.0 Set up & Implementation

4.1 Path Planning

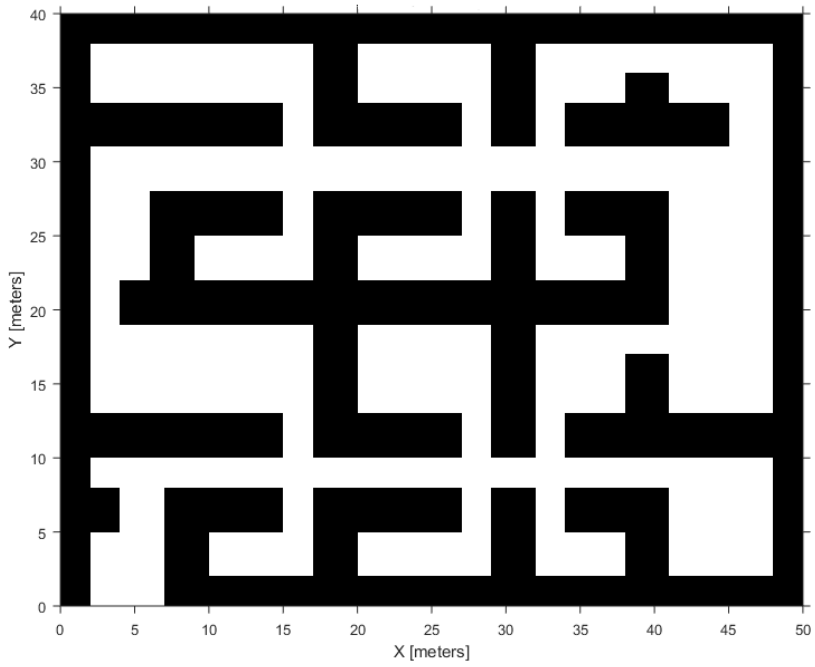


FIGURE 21 MAP 1 (BLACK REPRESENTS WALL WHILE WHITE IS FREE SPACE)

MATLAB Command Window	load map
	map = flip(map)
	map = robotics.BinaryOccupancyGrid(P, Resolution)

P is a matrix used to create a map with 1s and 0s, where 1s are space covered by obstacles and 0s are free space. The map is divided into square nodes, 50x40 is equal to 2000 nodes. Resolution specifies cells per square meter.

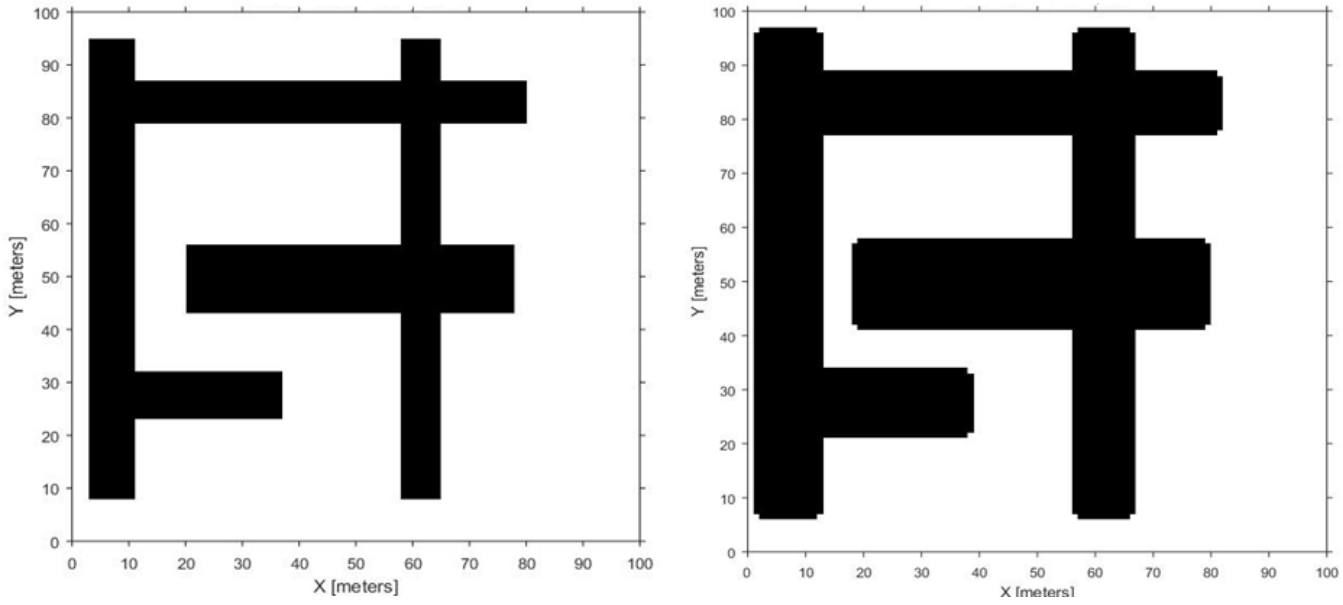


FIGURE 22 MAP 2 (LEFT) AND INFLATED MAP 2 (RIGHT)

Figure 22 shows the same map on both left and right plots but the figure to the right has been inflated by a radius of 1.5. This is a useful technique because the size of the rover can be considered while plotting the planned path.

MATLAB Command Window

inflate(map, radius)

Updated_map = occupancy grid

Updated_map returns a matrix that represent the inflated map in binary values. This map is then loaded into the Dstar MATLAB script before the minimum cost path is calculated. A goal coordinate and a start coordinate must be set before the path is calculated. The cost can be manually increased over a rough terrain; this is useful because the rover may take longer time to move on a particular surface. See Appendix I for the integrated path planning script. The Dstar script should have been able to inflate the map without integrating it with the technique used as described in this section. The toolbox describes the method to inflate the map using the Dstar script, however, it did not run using the latest version of the toolbox. Therefore, an alternative method to inflate the map has been integrated with the original script. It requires the Robotic Systems Toolbox to be installed on MATLAB[4].

4.1 Skid Steered Motion Model

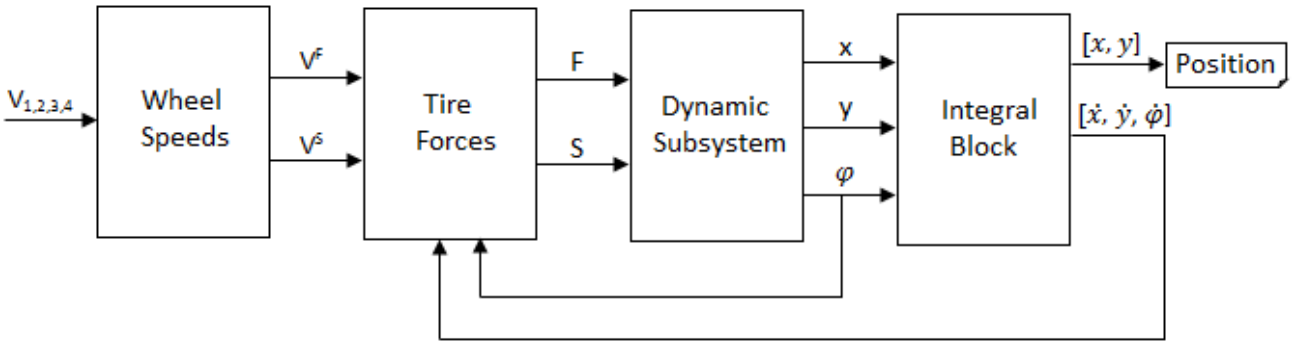


FIGURE 23 SHOWS THE IMPLEMENTED BLOCK DIAGRAM OF THE ROBOTIC PLATFORM

The angular velocities of the DC motors are converted to linear velocities and fed into the wheel block. The wheel block shown in the theory section is made up of wheel speeds and tire forces.

Table 1: Physical System

Definition	Symbol	Value
Wheel base	a	0.615 m
Wheel track	b1	0.573 m
Weight	W	20 kg
Relative position of CG	p	0.5 (50%)
Coefficient of friction	μ_0	0.61
Moment of Inertia	J	1.2 kgm^2
Stiffness of tires	C	5000 Nm^{-1}

[4] https://uk.mathworks.com/help/robotics/ug/install-robotics-system-toolbox-support-packages.html

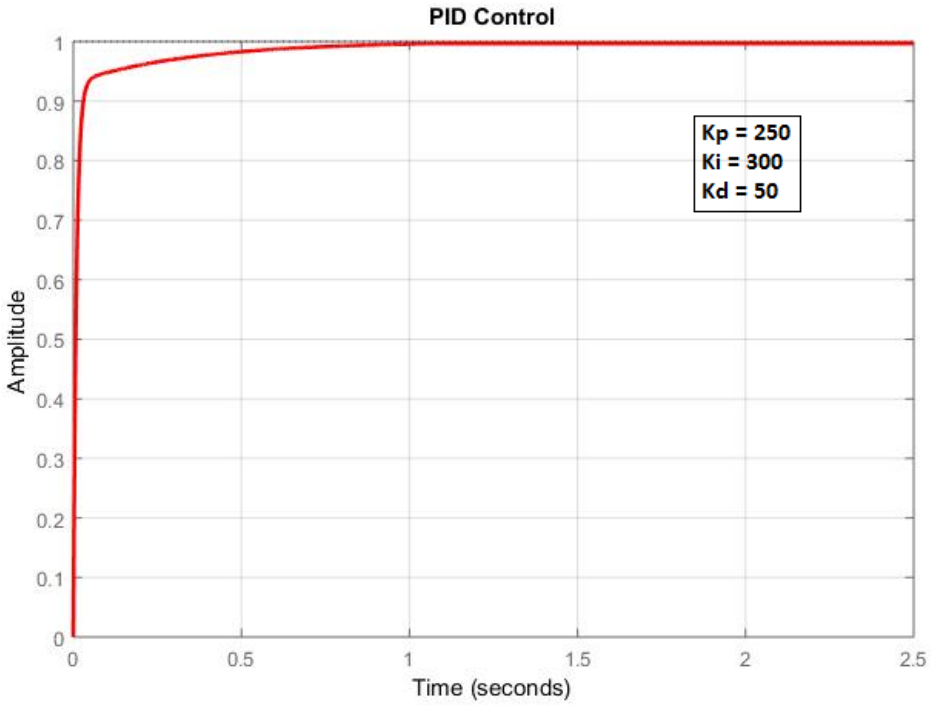


FIGURE 24 PID CONTROL TUNING OF DC MOTOR (TIME VS AMPLITUDE)

The motor should rotate at the desired speed (settling time less than 2 seconds) and reach it as soon as possible (steady state error less than 1%). A speed faster than the reference may damage the equipment and hence an overshoot less than 5% is required. The current settings give fast rise time, no overshoot and no steady state error and therefore meet the design requirements.

Table 2: DC Motor Parameters

Definition	Symbol	Value
Moment of inertia of the rotor	J	0.01 kgm^2
Motor viscous friction constant	b	0.1 Nms
Electromotive force constant	K_e	0.01 V/rad/s
Motor torque constant	K_t	0.01 Nm/Amp
Electric resistance	R	1 Ohm
Electric inductance	L	0.5 H

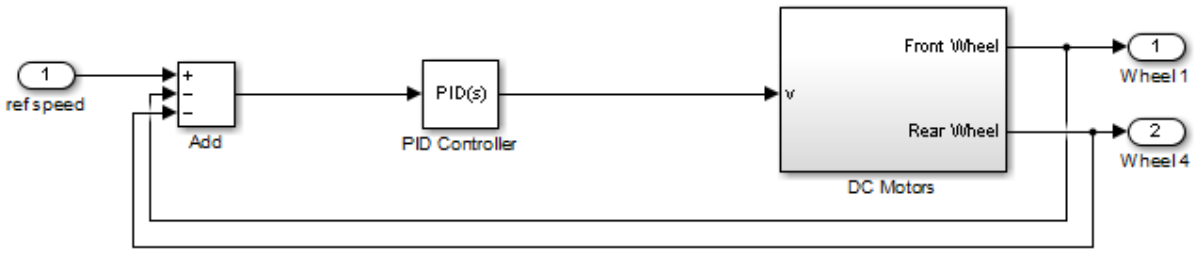


FIGURE 25 SHOWS A BLOCK DIAGRAM OF THE IMPLEMENTATION OF THE PID CONTROLLER AND DC MOTORS

A parallel connection of the DC motors on each side enables one PID controller to control the speed of two wheels.

4.2 Control Strategy

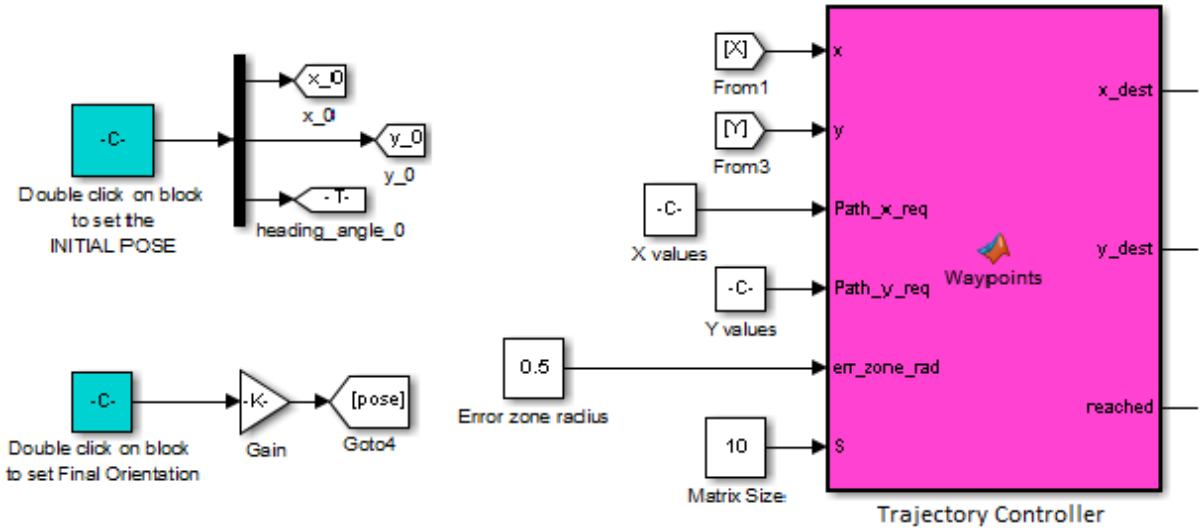


FIGURE 26 SHOWS THE SIMULINK IMPLEMENTATION OF THE TRAJECTORY CONTROLLER

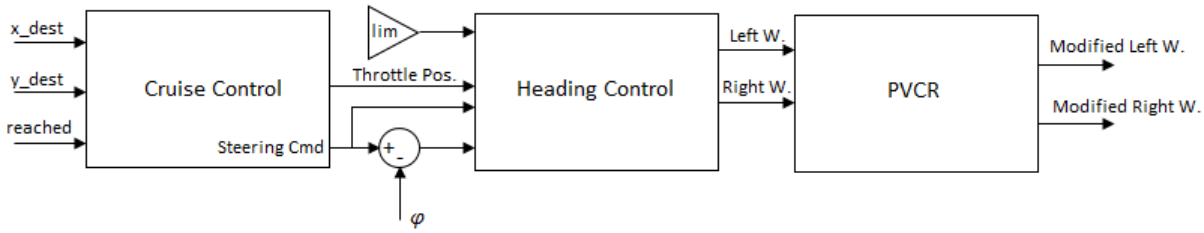


FIGURE 27 SHOWS THE BLOCK DIAGRAM OF THE IMPLEMENTED CONTROL STRATEGY

Table 3: Control Settings

Definition	Parameter	Value
Maximum heading error before a zero-radius-turn	lim	5 deg.
Initial Pose	-	[0; 0; 0]
Final Orientation	orient	90 deg
Matrix Size	S	x path coordinates
Error zone radius	erad	0.5 m

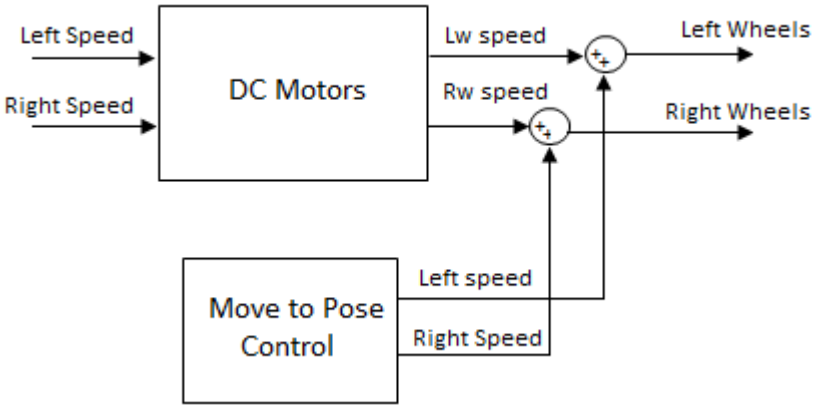


FIGURE 28 SHOWS THE IMPLEMENTATION OF MOVE TO POSE CONTROL

The model of the Move to Pose Control uses the same idea shown in the theory section, but is implemented differently. The DC motors take time to stop but an instant stop is required to send a signal to the Trajectory Controller that the navigation is finished. This was the reason for a different implementation approach. The speed generated by the DC motors are now being multiplied by reached as well as the controlled velocity. Reached equal to zero activates the left and right speeds of the Move to Pose Control. Opposite speeds are generated until the rover yaw (current heading) angle is aligned with the desired final orientation defined at the beginning of the simulation. Further breakdown and code script of the Move to Pose Control is shown in Appendix I.

4.4 Complete Model

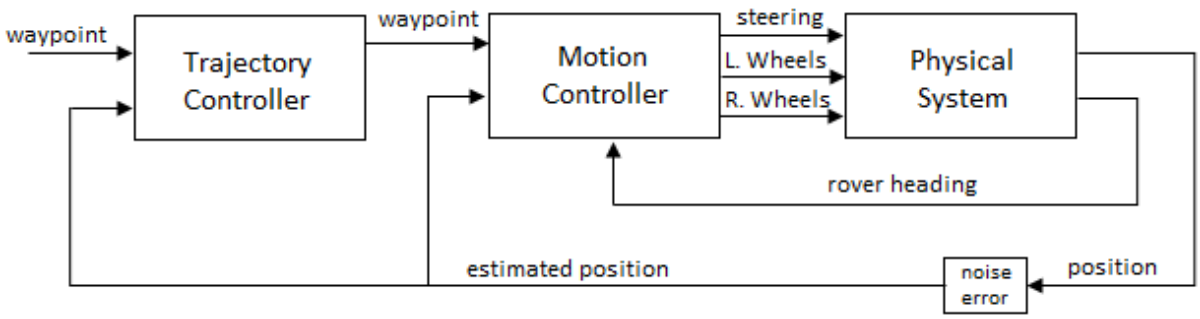


FIGURE 29 UPPER LEVEL BLOCK DIAGRAM OF THE IMPLEMENTED FINAL MODEL

Figure 29 shows an upper level block diagram of the complete model consisting of the trajectory controller, motion controller and the physical system. Noise error of the motor encoders has been added as well. To sum up, the trajectory block controls the look-ahead and sends the current point the rover is heading towards. The motion controller sends speed signals to the physical system and guides it around the planned path.

5.0 Results & Discussion

5.1 Path Planning

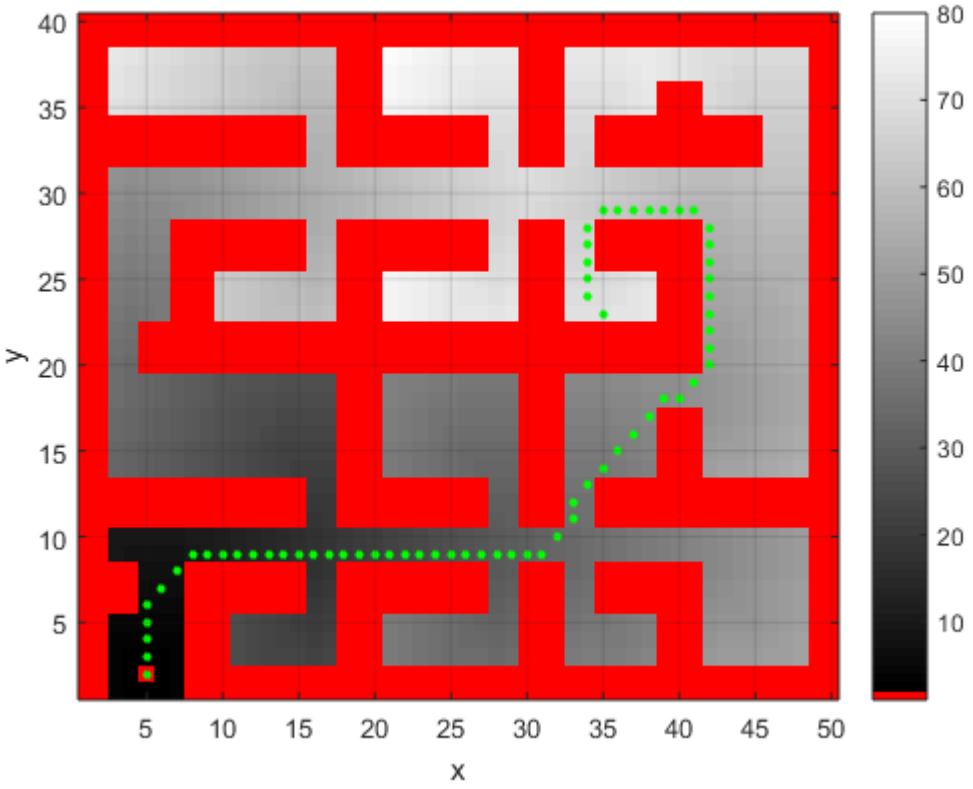


FIGURE 30 SHORTEST PHYSICAL DISTANCE COMPUTED BY DSTAR FROM [35,23] TO [5,2]

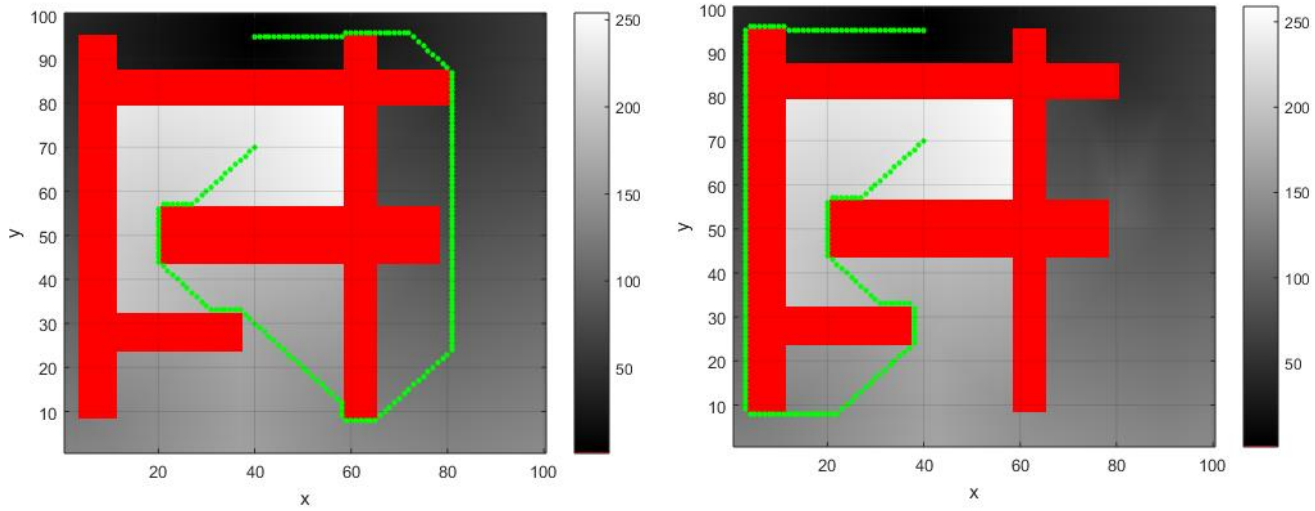


FIGURE 31 BOTH PLOTS SHOW A PATH PLANNED FROM [40,70] TO [40,95]. THE CELL COST IS HOWEVER INCREASED ON THE RIGHT PLOT ON THE WHOLE AREA THAT STRETCHES FROM X=60 TO X=100 FROM Y=0 TO Y=100.

5.2 Model Verification

5.2.1 Simulation 1

Table 4: Results from Simulation 1

Model	p-value	$\pm$ Wheel Velocities v_i [m/s]	Angular Velocity $\dot{\varphi}$ [rad/s]	Velocity of COG [m/s]	Turning radius R [m]
Solc & Sembera [Original]	0.54	0.12	-0.30	0.007	0.025
Solc & Sembera [Measured]	0.54	Not given	-0.31	0	0
Amtoft & Jensen [Recreated]	0.54	0.12	-0.233	Not given	0.020
Adam [Recreated]	0.54	0.12	-0.234	0.005	0.020
Motion Model [Recreated]	0.54	0.12	-0.45	0.0043	0.020
Motion Model [Tuned]	0.5	0.12	-0.19	0	0

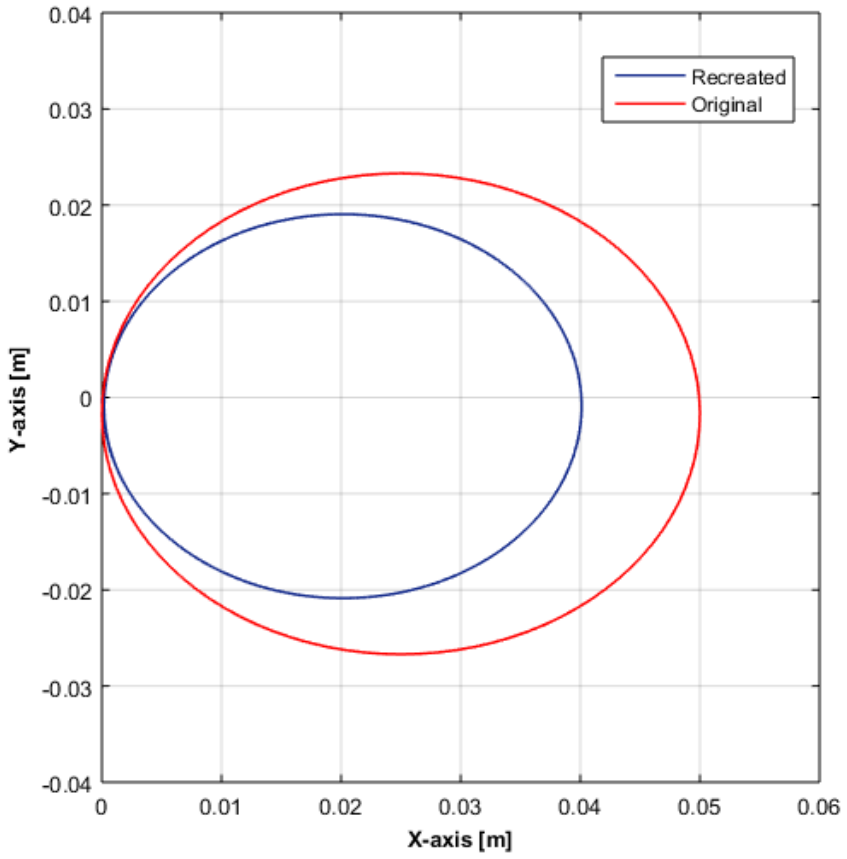


FIGURE 32 SKID-RADIUS OF THE ROVER AROUND ITS OWN AXIS (SIMULATION 1)

5.2.2 Simulation 2

Table 5: Resuls from Simulation 2

Model	p-value	Right Wheel Velocity v_R [m/s]	Angular Velocity $\dot{\varphi}$ [rad/s]	Velocity of COG [m/s]	Turning radius R [m]
Solc & Sembera [Original]	0.54	0.12	-0.14	0.06	0.4
Solc & Sembera [Measured]	0.54	Not given	-0.15	0.07	0.45
Amtoft & Jensen [Recreated]	0.54	0.12	-0.117	Not given	0.51
Adam [Recreated]	0.54	0.12	-0.12	0.06	0.51
Motion Model [Recreated]	0.54	0.12	-0.135	0.066	0.51
Motion Model [Tuned]	0.5	0.12	-0.097	0.06	0.62

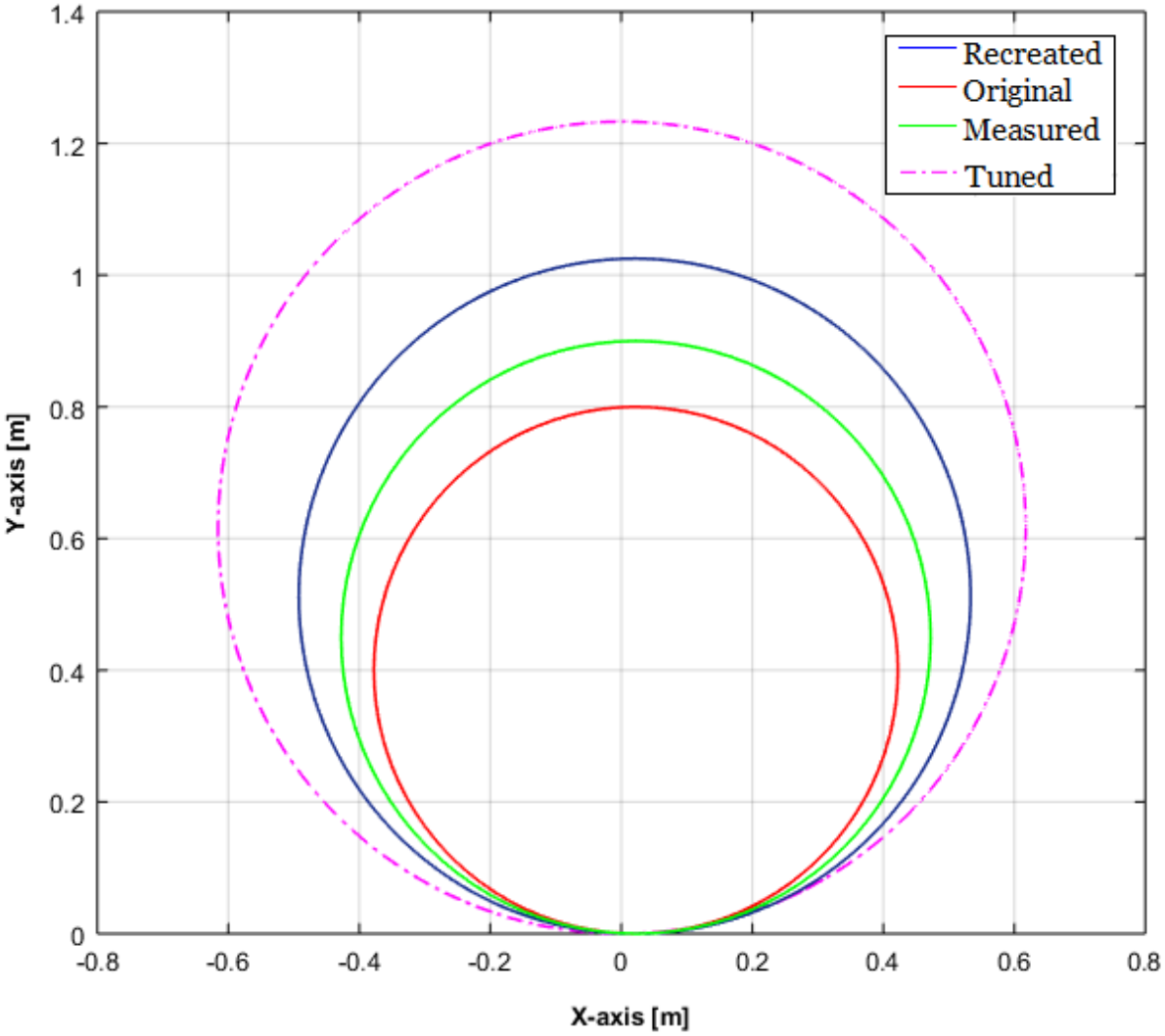


FIGURE 33 SKID-RADIUS OF THE ROVER AROUND ITS OWN AXIS (SIMULATION 2)

5.3 Control Verification

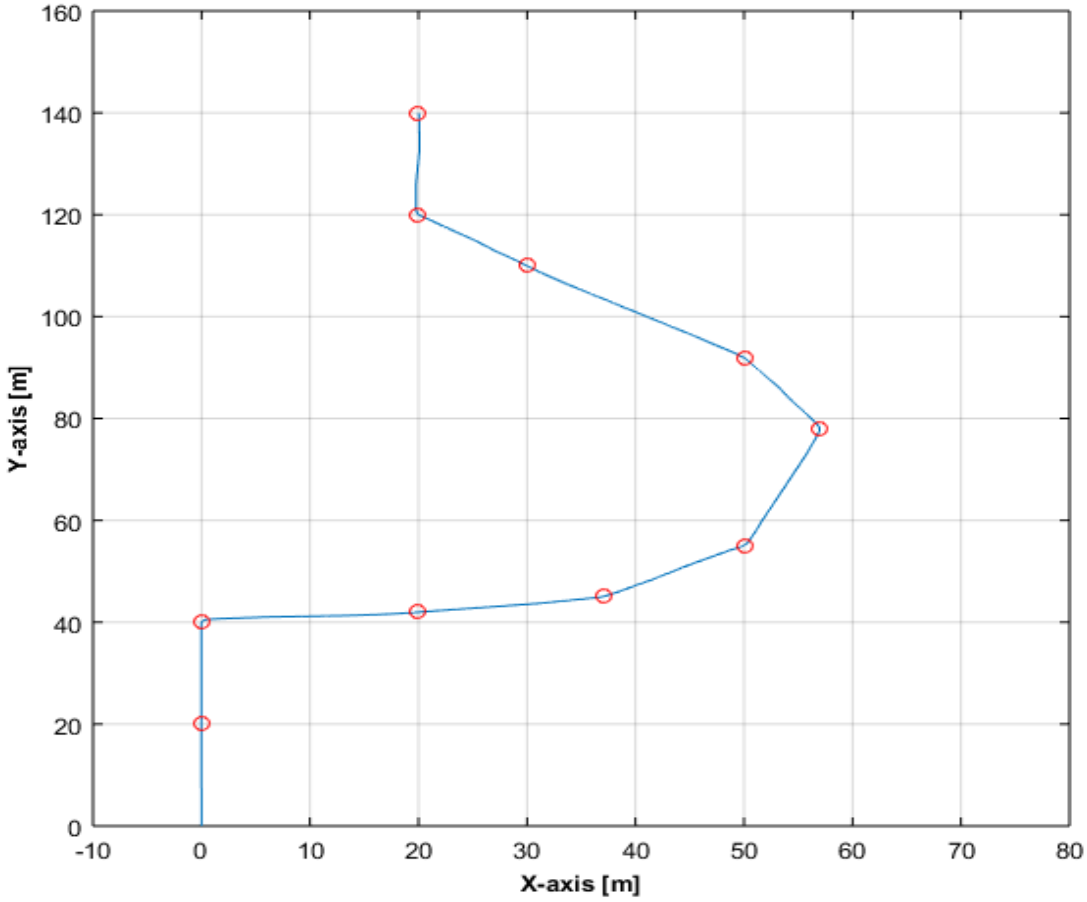


FIGURE 34 SHOWS ROVER MOVEMENT THROUGH WAYPOINTS

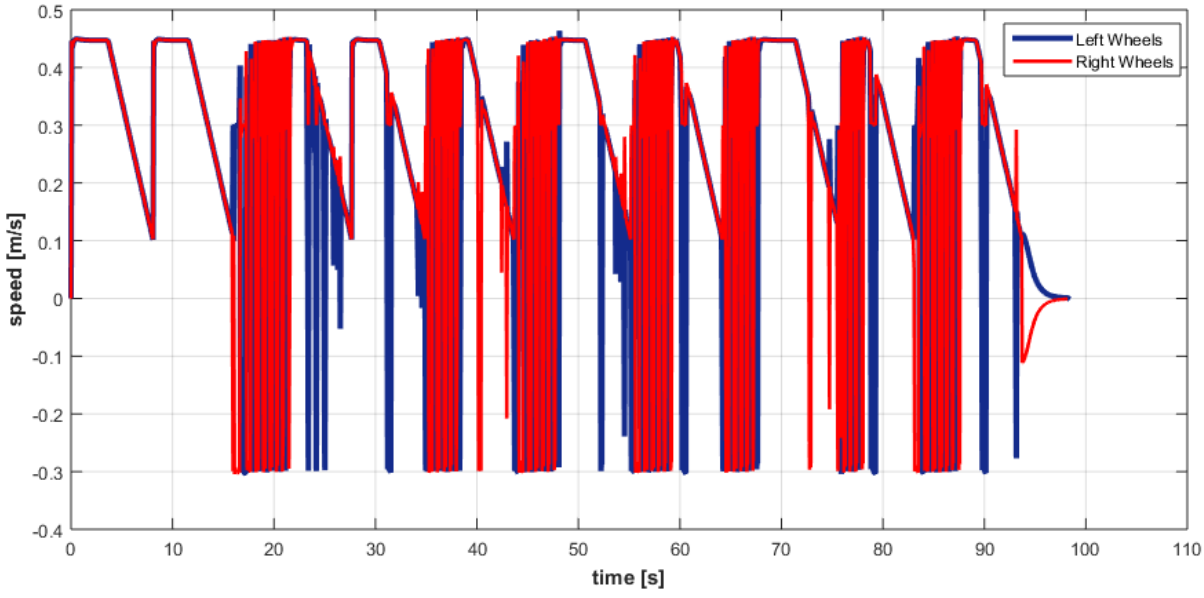


FIGURE 35 LEFT AND RIGHT SPEEDS DURING MOVEMENT ON THE PATH SHOWN IN FIGURE 34

5.4 Final Result

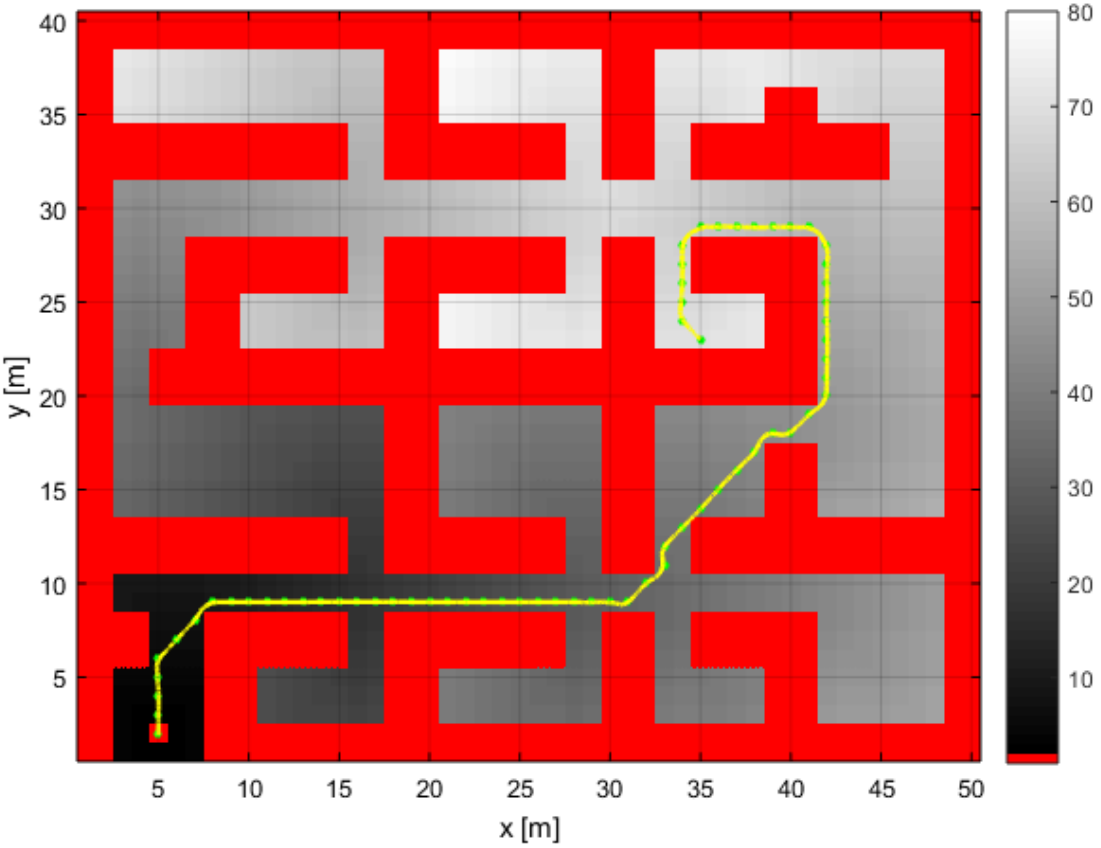


FIGURE 36 THE YELLOW LINE SHOWS ROVER MOVEMENT WHILE THE GREEN DOTS ARE THE WAYPOINTS

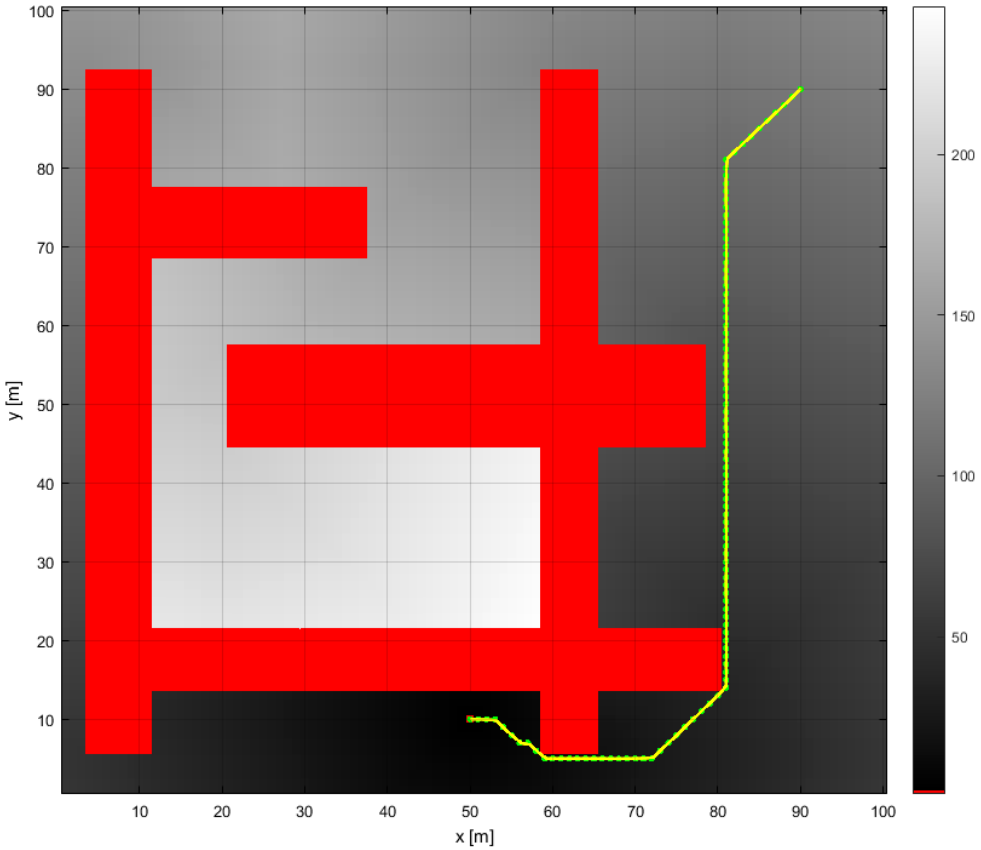


FIGURE 37 ROVER TRACKS DOWN A LONGER PATH WHILE CARRYING AN ADDITIONAL 30 KG TO ITS OWN WEIGHT

5.5 Discussion

Figure 30 shows the calculation of the shortest path between two points, where all nodes have the same cost. This basically means that moving vertically or horizontally through a node has a cost of 1 and moving diagonally through the node has a cost of 1.4 as described in chapter 3 section 3.1. Figure 31 has two plots with the same starting position and goal, but different cell costs. The left plot in figure 31 has an equally divided cost for all cells, hence each node movement will give similar cell cost as on figure 30 (1 for vertical or horizontal and 1.4 for diagonal movement). Therefore, figure 31 left plot shows plotting of the shortest moveable physical distance between the starting position and the goal. On figure 31 right plot the cell costs have been modified, hence the shortest physical distance is no longer the cheapest path. An increased cost could be a representation of a surface with higher friction or higher energy usage due to an uphill road. To conclude, the path planning considers the cheapest path based on defined cell costs. The cell cost is a representation of energy consumption and time needed to move through a cell. Therefore, the algorithm calculates the cheapest path based on cost distance. It is efficient and can be used to plan a path in real environment scenarios. The challenge will be for rover sensors to detect and classify an uneven (not flat, with changing roughness) surface.

The skid radius from Simulation 1 is identical to three recreated models. Since this model was recreated from the original report it is a high possibility that the original skid steered rover model is faulty. However, the angular velocity of the robotic platform around its own axis differs from all the previous data. It is not known which method was used to calculate/measure the velocities in the previous models. In the recreated model in this project, the average speed was taken during one full skid radius from the calculation by Simulink. The results for the tuned and originally measured model are not plotted in figure 32 because it has a skid radius of 0 m. Data from Simulation 2 shows a speed closer to the original model whilst a skid radius identical to the recreated models. Overall, the skid radius of the recreated model in this project is similar to results from previously recreated models. The radius is the same despite a difference in angular velocity.

The control is verified in section 5.3 as it shows accurate tracking of all waypoints. Despite an error radius of 0.5 m, the waypoints are tracked close to the middle of each waypoint radius. Hence, the combination of PVCR, the tuning of the DC motors and the controller proves to reduce the setbacks of follow-the-carrot method. The pure pursuit method is a superior method as it produces natural curved movements like a car driven by a human. In comparison, follow-the-carrot method produces movement in straight lines rather than a curvature movement. This is a downside as well as a plus point. It is a plus point in tightly navigated areas where a straight-line motion is better than a curvature movement to reduce the possibility of colliding in obstacles. Figure 35 is a representation of the speed of right and left wheels during the path movement shown in figure 34. It can be observed from figure 35 that the rover performs zero skid turns during the simulation. The zero skid radius turns are performed during opposite speeds on each sides (approximately 0.3 and -0.3 m/s in this simulation). A problem faced during the development of the control strategy was making the rover coming to a full stop at the final target. It was necessary to make the rover stop in the simulation without terminating the simulation because stopping it by a termination would prevent the opportunity of moving the rover into the desired pose. The rover would move past the final target because the DC motors cannot come to an instant stop although the throttle is closed. It would eventually stop but only after moving past the final target. Hence, the rover would regain speed due to the nature of the cruise control speed (the speed is proportional to the distance to goal). As a result, the rover would oscillate around the final target without stopping. This problem was solved as shown in section 4.2. On figure 35 it is seen that after 94 seconds the lines for left and right speeds merge slowly together as both sides end up with a speed of zero. This indicates the rover moving into position (desired final orientation was set to 30 degrees). Therefore, it can be concluded that the move to pose control is successfully implemented.

In reality, the rover will move slightly past the target, however, the control will still be able to move the rover into desired pose because the simulated data is what controls the move to pose control. On the simulated data, the overshoot is not shown. This is an important part of the controller design to optimise its performance.

Finally, Figure 36 and 37 show a great path tracking ability of the physical system with the guidance of the controller even under a longer path in a tight area with many obstacles. Hence, this program is efficient and accurate and ready to be combined with other subprojects.

6.0 Conclusion & Future Work

6.1 Conclusion

The D* algorithm has been understood, acquired, modified and implemented. This algorithm is efficient and considers physical distance, computational time and energy consumption by modifying the cost over an area. In other words, it can be set to consider changing surface characteristics as the rover moves through an environment. Its ability to replan a path during obstacle movement without having to restart the whole calculation process is the main advantage. Moreover, it considers the rover size as well while computing a path. These two qualities give this algorithm the edge over the others mentioned in the literature review and is therefore widely used in the robotic society.

A mathematical motion model has been developed and compared against previous work. Furthermore, the motion model was tuned to match the characteristics of the robotic platform Dr. Jaguar 4x4 Wheel. However, it was not implemented in the actual robotic platform and therefore actual performance is not taken into consideration.

The project deals with some aspects of adaptive law for control strategies, the idea behind adaptive control law was used to tune and enhance the performance of the rover on a single surface. This was done by programming a script that takes the rover position and compares it to planned position to alter acceleration limits and control parameters. The control strategy is built around the idea of follow-the-carrot method. The algorithm is robust enough to track down all waypoints in both concave and convex turns. In addition to guiding the rover over the planned path, the control strategy can move the rover into a desired position on reaching the target.

An efficient program for autonomous navigation has been developed and ensured it works by simulating it in different scenarios with different weight categories of the rover. However, the program was not integrated with other subprojects thereby not implementing it in the actual rover. Therefore, the aim and all objectives listed in chapter 1 were met apart from the last objective which was partially met as the program was not implemented in the actual rover.

6.2 Future Work

While this thesis has presented the development and implementation of an efficient and robust program which enables a skid steered robotic platform to be guided in an unknown environment while avoiding obstacles, many opportunities for extending the scope of the project remains. This section describes some of these directions.

First and foremost, looking at the limitations of the developed program the following areas can be worked on:

- The developed motion model could be tested against the actual robotic platform.
- The developed skid steered motion model is limited mainly because it is assumed that the rover runs on a flat surface and that its wheels always have ground contact. This can be improved by modelling a motion model which considers the rover climbing stairs of the size suitable for Dr. Jaguar 4x4 Wheel.
- The path tracking method could be changed to increase the accuracy of waypoint tracking.
- The developed navigation program could be integrated with mapping and image processing.

Secondly, the scope of the project can be extended by applying the developed program to a specified task. Referring to the introduction where the scenario of a team of quadcopters is described; a network of quadcopters interacting with each other could increase the capability of firefighters. Similarly, this program could be edited to suit a specified mission. It can further be developed to enable a team of autonomous mobile robots to navigate and interact with each other to perform a task together.

References

Adam, I. (2016). *Autonomous Navigation of Mobile Robots in Unknown Environments.* Liverpool: University of Liverpool.

Amtoft, K., & Jensen, S. B. (2011). *Mark 2.* Aalborg: Aalborg University. Retrieved from http://projekter.aau.dk/projekter/files/52660463/MARKII.pdf

Autonomy Levels. (1985). In J. G. Brookshear, *Computer Science: An Overview* (p. 325).

Borenstein, J., & Koren, Y. (1989). Real-time Obstacle Avoidance for Fast Mobile Robots. *IEEE Transactions on Systems, Man, and Cybernetics* (pp. 1179-1187). IEEE.

Choset, H. (2007, September 19). *Carnegie Mellon University.* Retrieved from Motion Planning: https://www.cs.cmu.edu/~motionplanning/lecture/AppH-astar-dstar_howie.pdf

Choset, H. (2007). *Robotic Motion Planning.* Boston: Robotics Institute 16-735. Retrieved from https://www.cs.cmu.edu/~motionplanning/lecture/AppH-astar-dstar_howie.pdf

Clarke, M., & Blanchard, T. (2010). Development of a Control System for a Skid-Steer Amphibious Vehicle. (p. 6). Aberystwyth: Intelligent Robotics Group.

Clough, B. T. (2002). Metrics, Schmetrics! How The Heck Do You Determine A UAV's Autonomy Anyway. *Air Force Research Laboratory,Wright-Patterson AFB,OH,45433* (pp. 1-8). Washington: U.S. Department of Defense. Retrieved from http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA515926

Corke, P. (2011). *Robotics, Vision and Control.* Brisbane: Springer.

Devaurs, D., Simeon, T., & Cortes, J. (2015). *Optimal Path Planning in Complex Cost Spaces with Sampling-based Algorithms.* Karaviklab. Retrieved from http://www.kavrakilab.org/publications/devaurs-15-tase.pdf

Dr Robot. (2014). All-Terrain Autonomous Navigation Robot with GPS-IMU. Markham, Ontario, Canada. Retrieved from Dr Robot: http://jaguar.drrobot.com/images/Jaguar_4x4_wheel_manual.pdf

Edy. (n.d.). Retrieved from Pacejka '94 parameters explained – a comprehensive guide: http://www.edy.es/dev/docs/pacejka-94-parameters-explained-a-comprehensive-guide/

Hieema, M. (2013). *Motion Planner for Skid-Steer Unmanned Ground Vehicle.* Tallinn: Tallinn University of Technology.

Iagnemma, K., & Dubowsky, S. (2004). In *Mobile Robots in Rough Terrain* (pp. 7-12). Cambridge: Massachusetts Institute of Technology.

Koren, Y., Borenstein, J., & Arbor, A. (1991). *Potential Field Methods and Their Inherent Limitatons for Mobile Robot Navigation.* Sacramento: Institute of Electrical and Electronics Engineers.

Kuipers, B. (2005). *The University of Texas at Austin.* Retrieved from Autonomous Robots Resources: http://www.cs.utexas.edu/~pstone/Courses/395Tfall05/resources/

Kummer, H. (1996). Unified Theory of Rubber and Tire Friction. *Engineering Research Bulleting B-94.* The Pennsylvania State University.

Lee, T.-L., & Miller, D. P. (1998). High-Speed Traversal of Rough Terrain Using a Rocker-Bogie Mobility System. Norman, Oklahoma, USA.

Lundgren, M. (2003). Path Tracking for a Miniature Robot. Umeå, Sweden. Retrieved from http://www8.cs.umu.se/kurser/TDBD17/VT06/utdelat/Assignment%20Papers/Path%20Tracking%20for%20a%20Miniature%20Robot.pdf

Mafrica, S. (2012). *Advanded Modeling of a Skid-Steering Mobile Robot for Remote Telepresence.* Genova: University of Genova.

Mandow, A., Marinez, J. L., Morales, J., Blanco, J. L., Garcia, A., & Jimenez, J. G. (2007). Experimental kinematics for wheeled skid-steer mobile robots. *Intelligent Robots and Systems* (p. 7). Malaga: IEEE Xplore.

Mondal, R. (2014, February 27). *Ron Robotics*. Retrieved from 4 Wheeled Robot Design Basics and Challenge: http://www.rakeshmondal.info/4-Wheel-Drive-Robot-Design

Pompa, O. E., & Suarez, I. d. (2016). *Control of in-wheel motors for an outdoor skid-steering robot.* Milan: Politecnico Di Milano.

Raymond, M. B. (2006). *Brach Engineering.* Retrieved from Adhesion, Hysteresis and the Peak Longitudinal Tire Force: https://www.brachengineering.com/content/publications/Wheel-Slip-Model-2006-Brach-Engineering.pdf

Roy, A. (2016). *Design of an Unmanned Ground Vehicle Capable of Autonomous Navigation.* Columbus: The Ohio State University.

Samuel, M., Hussein, M., & Mohamad, M. B. (2016, February). A Review of some Pure-Pursuit based Path Tracking Techniques for Control of Autonomous Vehicle. *International Journal of Computer Applications*, pp. 35-38. Retrieved from http://www.ijcaonline.org/research/volume135/number1/samuel-2016-ijca-908314.pdf

Schechter, E. (2014, November 24). *Popular Mechanics*. Retrieved from Firefighting Drones Tested at U.S. Air Force Base: http://www.popularmechanics.com/flight/drones/a11655/firefighting-drones-tested-at-us-air-force-base-17461645/

Solc, F., & Sembera, J. (2008). Kinetic Model of a Skid Steered Robot. *Signal Processing, Robotics and Automation* (p. 5). Brno: University of Cambridge. Retrieved from http://www.wseas.us/e-library/conferences/2008/uk/ISPRA/ispra-08.pdf

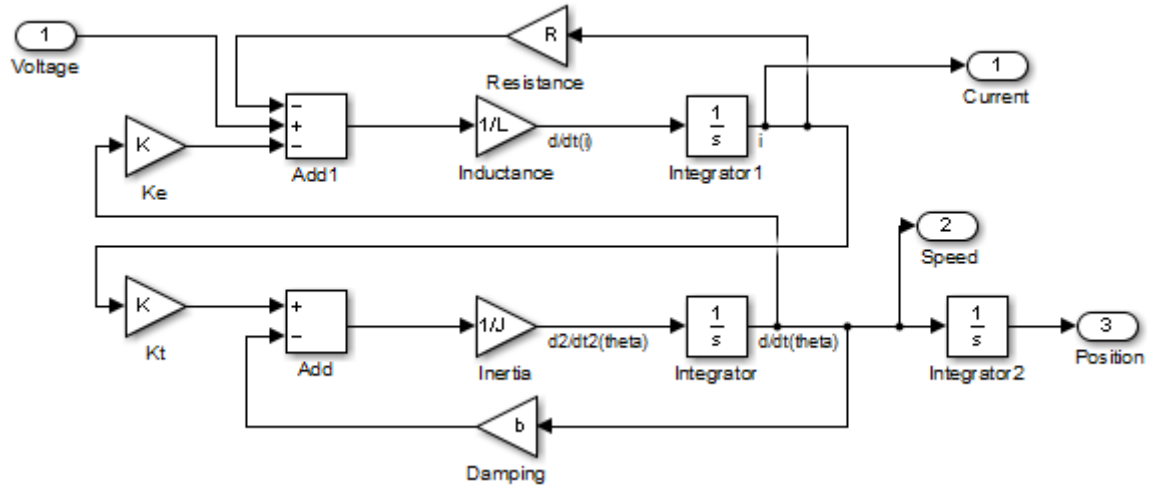
Wit, J. S. (2000). *Vector Pursuit Path Tracking for Autonomous Ground Vehicles.* Florida: Center for Intelligent Machines and Robotics.

Wit, J., Crane, C. D., & Armstrong, D. (n.d.). *Autonomous Ground Vehicle Path Tracking.* Gainesville, Florida: Center for Intelligent Machines and Robotics. Retrieved from http://www.ijcaonline.org/research/volume135/number1/samuel-2016-ijca-908314.pdf

World Nuclear Association. (2016, July). Retrieved from Fukushima Accident: http://www.world-nuclear.org/information-library/safety-and-security/safety-of-plants/fukushima-accident.aspx

Appendix I: SIMULINK models & MATLAB scripts

DC Motor Subsystem Lower Level Block Diagram

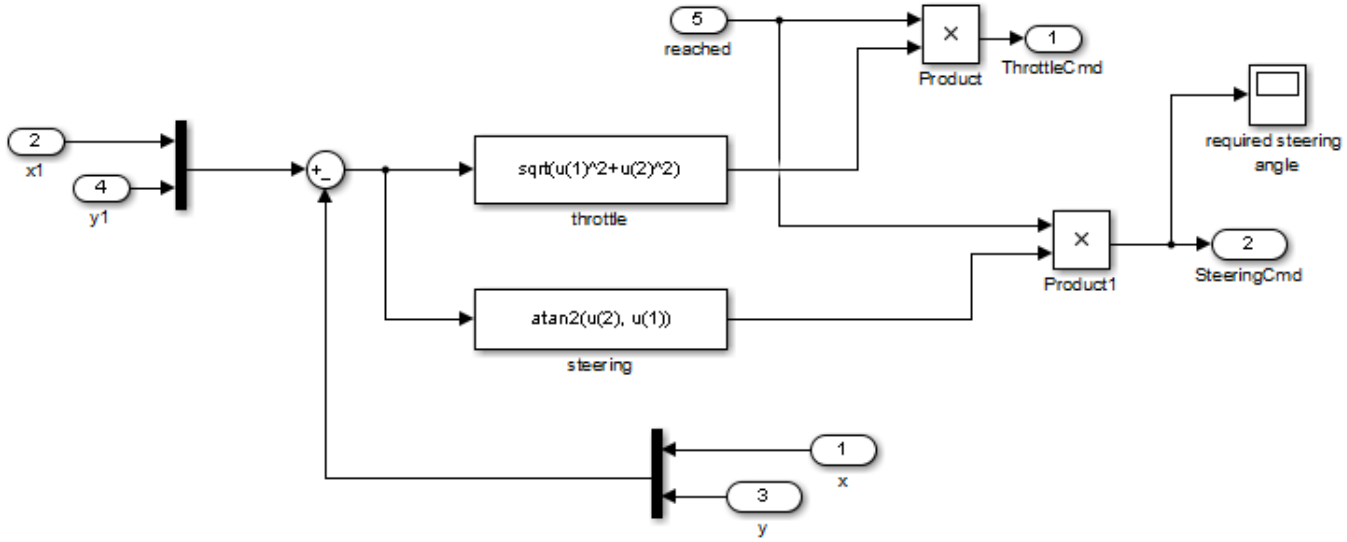


Control system output

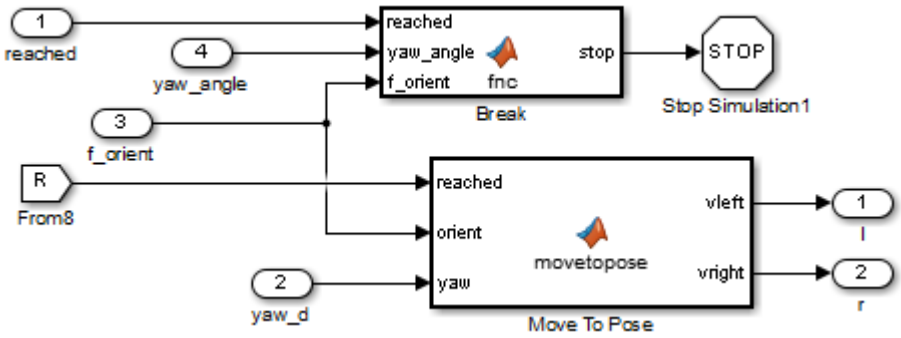
```
C = pid(Kp,Ki,Kd) %Controller settings
J = 0.01; %moment of inertia
B = 0.1; %viscous friction constant
K = 0.01; %electromotive force constant=torque constant
R = 1; %electric resistance
L = 0.5; %electric inductance
s = tf('s'); %transfer function
DC_Motor = K/((J*s+B)*(L*s+R)+K^2) %DC_motor equation

Gc = feedback(C*DC_Motor,1)%closed loop transfer function
step(Gc) %see tuning results
```

Cruise Control Lower Level Block Diagram



Move To Pose Control



Brake

```
function stop = fnc(reached, yaw_angle, f_orient)
stop=nan;
if yaw_angle>(f_orient - 0.01) && yaw_angle<(f_orient + 0.01)
    stop = 1*reached; % stops the simulation after the rover is moved into
desired final position
else
    stop = 0; %prevents a termination of the simulation if the navigation
is not finished yet
end
end
```

Move To Pose

```
function [vleft, vright] = movetopose(reached, orient, yaw)
vleft = 0.1*(reached-1)*(orient-yaw) %speed on left wheels
vright = -0.1*(reached-1)*(orient-yaw) %speed on right wheels
```

Setting file:

```
a=0.615; %wheel base in [m]
b1=0.573; %wheel track in [m]
W=20; %weight of the rover
p=0.5; %relative position of centre of gravity in percentage
mu=0.61; % coefficient of friction
Ji=1.2; %moment of inertia
c=5000; %tire stiffness
J=0.01; %dc motor: moment of inertia of the rotor
K=0.01; %dc motor: Ke=Kt, Ke=electromotive force constant, Kt= torque
constant
L=0.5; %dc motor: electric inductance
g=9.81; %gravity acceleration constant
R=1; %dc motor: electric resistant
r=0.13; %radius of wheel
Kv=1; %cruise control gain
b=0.1; % dc motor: viscous friction constant
load ('input map name'); %loads map to workspace
map = flip('input map name'); %map representation is corrected to match the
Dstar algorithm
map = robotics.BinaryOccupancyGrid(map, 1); %map created
inflate(map,2); %map is inflated by K cells
Updated_map = occupancyMatrix(map); %map is converted to a logical array
dmap=double(Updated_map); %map is converted to a numerical matrix
ds=Dstar(dmap);
goal=['x1','y1'];
start=['x','y'];
ds.plan(goal);
ds.path(start);
plannedpath = ds.path(start);
Path_x_req = [plannedpath(:,1)];
Path_y_req = [plannedpath(:,2)];
z=2; % gain for wheel speed
ipose=['x','y','rad']; % initial pose [x, y, rad]
```

Plot path tracking result

```
hold on
figure(1)
plot(simout2.data(:,1),simout2.data(:,2),'-y.')
hold on;grid on;axis
```

Trajectory Controller

```
function [x_dest,y_dest,reached] =
Waypoints(x,y,Path_x_req,Path_y_req,err_zone_rad,S)
%#codegen
reached = 1; %if the rover has not reached the goal the velocities will be
generated as normal
persistent final_destination; %defines the target that is local to the
function in which they are declared.
%The values are retained in memory between calls to function.
if(isempty(final_destination))
 final_destination = zeros(1,206);
 final_destination(1) = 1;
end
%find the index of highest 1 in current target matrix
j=1;
while((j<S+1)&&(final_destination(j)==1))
 j=j+1;
end
j=j-1; % This is that index
dist_target = sqrt((Path_x_req(j)-x)^2 + (Path_y_req(j)-y)^2); %calculates
the distance from target
if(dist_target <err_zone_rad) %rover considers next target if it has
reached within a given radius of current target
 if(j<S)
 j=j+1;
 final_destination(j)=1;
 end
 if(j==S) %S is the size of the matrix, basically the number of coordinates
used to define the desired path.
     %change manually in simulink. Constant block with name size created.
 reached=0; %upon reaching the goal, or within the given radius of goal the
rover stops.
 %Reached = 0 eliminates the exisiting error in the velocity block inside
 %the main controller
 final_destination(S)=1;
 end
end
x_dest = Path_x_req(j); %x coordinates of desired path
y_dest = Path_y_req(j); %y coordinates of desired path
```

PVCR

```
function [v_CLm, v_CRm] = PVCR(v_CL, v_CR, d, limit)
mode = 1 % default = 1, switching it to zero turns off the system
% v_CL = controlled left side velocity
% v_CR = controlled right side velocity
% v_CLm = modified left side velocity
% v_CRm = modified right side velocity
Vtotal = (abs(v_CL + v_CR))/2 %takes the absolute value of left side speed
and
%the right side speed coming out from the heading controller
if abs(d) < limit*pi
    a_limit = 0.02 %this is the maximum acceleration limit allowed, set it
accordingly to the path or as desired.
else
    a_limit = 10
end

if a_limit >= mode*Vtotal % if the acceleration limit is larger or equal
than Vtotal, then the original velocity
    %coming out of the heading controller will remain unmodified and fed
    %into the DC motors
    v_CLm = v_CL
    v_CRm = v_CR
else %otherwise if total velocity exceeds the acceleration limit then the
left side and right side
    %speeds will be modified (reduced proportionally)
    v_CLm = sqrt(a_limit/Vtotal)*v_CL
    v_CRm = sqrt(a_limit/Vtotal)*v_CR
end
```

Wheel Speeds

```
function [LongVel_Wheel_left, LongVel_Wheel_right, LatVel_Wheel_front,
LatVel_Wheel_rear] = Initial_frame(xdot, ydot, yaw_rate, yaw_angle)
b = 0.573 % Wheel track in metres
p = 0.5; % Relative position of centre of gravity
a = 0.615; % Wheel track in metres
% xdot = vehicle speed in m/s
% ydot = lateral vehicle speed m/s
% yaw_rate = yaw rate of vehicle in rad/s
% yaw_angle = yaw angle of vehicle in radians
% LongVel_Wheel_left = Longitudinal wheel velocities for left wheels
% LongVel_Wheel_right = Longitudinal wheel velcoties for right wheels
% LatVel_Wheel_front = Lateral wheel velcoties for front wheels
% LatVel_Wheel_rear =  Lateral wheel veocities for rear wheels
LongVel_Wheel_left = xdot*cos(yaw_angle) + ydot*sin(yaw_angle) -
(b/2)*(yaw_rate);
LongVel_Wheel_right = xdot*cos(yaw_angle) + ydot*sin(yaw_angle) +
(b/2)*(yaw_rate);
LatVel_Wheel_front = ydot*cos(yaw_angle) - xdot*sin(yaw_angle) +
p*a*(yaw_rate);
LatVel_Wheel_rear = ydot*cos(yaw_angle) - xdot*sin(yaw_angle) - (1-
p)*a*(yaw_rate);
```

Tire Forces

```
function [F1, S1, F2, S2, F3, S3, F4, S4] =
Wheel_Dynamics(LongVel_Wheel_left, LongVel_Wheel_right, LatVel_Wheel_front,
LatVel_Wheel_rear, w1, w2, w3, w4)
mu = 0.61; % coefficient of friction
W = 20; % weight of the rover in kg
C = 5000; % stiffness of tyres in N/m
r=0.13; % wheel radius
% LongVel_Wheel_left = Longitudinal wheel speeds for left side
% LongVel_Wheel_right = Longitudinal wheel speeds for right side
% LatVel_Wheel_front = Lateral speeds for front wheels
% LatVel_Wheel_rear = Lateral speeds for rear wheels
% w(i) = angular speed of DC motor, (i) refers to the wheel the motor is
% connected to
% F(i) = Longitudinal wheel force, (i) refers to the wheel number
% S(i) = Lateral wheel force, again (i) refers to the wheel number
% Front wheels are numbered 1 and 2 from left to right, and rear wheels are
% numbered 3 and 4 from right to left
v1 = w1/r;
v2 = w2/r;
v3 = w3/r;
v4 = w4/r;
Long_slip_vel_1 = v1 - LongVel_Wheel_left;
Long_slip_vel_2 = v2 - LongVel_Wheel_right;
Long_slip_vel_3 = v3 - LongVel_Wheel_right;
Long_slip_vel_4 = v4 - LongVel_Wheel_left;
 Total_slip_vel_1 = sqrt((Long_slip_vel_1)^2 + (LatVel_Wheel_front)^2);
Total_slip_vel_2 = sqrt((Long_slip_vel_2)^2 + (LatVel_Wheel_front)^2);
Total_slip_vel_3 = sqrt((Long_slip_vel_3)^2 + (LatVel_Wheel_rear)^2);
Total_slip_vel_4 = sqrt((Long_slip_vel_4)^2 + (LatVel_Wheel_rear)^2);

if Total_slip_vel_1 > (mu*W)/C;
    P1 = mu*W;
else Total_slip_vel_1 <= (mu*W)/C;
    P1 = C*Total_slip_vel_1;
end
if Total_slip_vel_2 > (mu*W)/C;
    P2 = mu*W;
else Total_slip_vel_2 <= (mu*W)/C;
    P2 = C*Total_slip_vel_2;
end
if Total_slip_vel_3 > (mu*W)/C;
    P3 = mu*W;
else Total_slip_vel_3 <= (mu*W)/C;
    P3 = C*Total_slip_vel_3;
end
if Total_slip_vel_4 > (mu*W)/C;
    P4 = mu*W;
else Total_slip_vel_4 <= (mu*W)/C;
    P4 = C*Total_slip_vel_4;
end

F1 = P1*(Long_slip_vel_1/Total_slip_vel_1);
F2 = P2*(Long_slip_vel_2/Total_slip_vel_2);
F3 = P3*(Long_slip_vel_3/Total_slip_vel_3);
F4 = P4*(Long_slip_vel_4/Total_slip_vel_4);
S1 = -P1*(LatVel_Wheel_front/Total_slip_vel_1);
S2 = -P2*(LatVel_Wheel_front/Total_slip_vel_2);
S3 = -P3*(LatVel_Wheel_rear/Total_slip_vel_3);
S4 = -P4*(LatVel_Wheel_rear/Total_slip_vel_4);
```

Angle wrap

```
function angle = wrap(angle2, minAngle, span)
%wrap Input angle is wrapped to [minAngle,minAngle+span].
% All angles should be given in either degrees or radians.
% For example, minAngle = -pi/2 and span = pi would convert the input angle
% to the interval -pi/2 to +pi/2.
%% Calculation
angle = angle2 - span.*floor((angle2 - minAngle)./span);
end
```

Heading Controller

```
function [Lw, Rw, alpha] = Heading_Controller(d, ThrottlePosition,
SteeringAngle, theta, limit)
% Lw = Left wheel speeds
% Rw = Right wheel speeds
% alpha = modified steering angle
% d = difference between steering angle and current yaw angle of the rover
% ThrottlePosition = controlled velocity from the throttle
% SteeringAngle = steering angle
% theta = yaw angle of the rover
% limit = the limit of direction angle allowed before it the error is
% corrected to align the rover heading with the original target
limit_radians = (limit*pi)/180
if abs(d) < limit_radians % if the difference between the yaw and desired
steering angle is less than the limit
    % the following command controls the wheel speeds
    Lw = ThrottlePosition - 0.05*d
    Rw = ThrottlePosition + 0.05*d
    alpha = SteeringAngle
else abs(d) > limit_radians % if the difference between the yaw and desired
steering angle is greated than the limit
    % the following command control performs the wheel speeds causing the
    % rover to skid
    if d > 0 % if difference between the yaw and desired steering angle is
greater than zero, then right
        % turn is activated by reversing the left wheel speeds in the same
        % but negative value of the right wheel speeds
    Lw = -0.3
    Rw = 0.3
    alpha = theta %the modified steering angle is now set equal to the yaw
angle to maximise the rotation of the rover
    else d < 0 % when the difference between the yaw angle and desired
steering angle is less than zero, right turn is
        %triggered
        Lw= 0.3
        Rw= -0.3
        alpha = theta
    end
end
```

Appendix II: Calculations

Inertia calculation using equation (23) from the thesis:

Model by Solc and Sembera: $\mathrm{I} = \frac{59}{12}(0.5^2 + 0.4^2) = 2.016 \approx 2$

The inertia value given in the paper is 2 kgm^2 as well. Therefore, this is the method used to calculate the inertia of the robotic platform used in this project.

Tuned rover model: $\mathrm{I} = \frac{20}{12}(0.615^2 + 0.573^2) = 1.18 \approx 1.2$

Appendix III: Gantt Chart