2013

Investigation on the mobile robot navigation in an unknown environment

Ahmed S. Khusheef
Edith Cowan University

Edith Cowan University

Copyright Warning

Investigation on the Mobile Robot Navigation in an Unknown Environment

Author: Ahmed Shany Khusheef

Student No: 10139104

This thesis is presented for the degree of

Master of Engineering Science

Edith Cowan University

Faculty of Computing, Health and Science, School of Engineering

USE OF THESIS

The Use of Thesis statement is not included in this version of the thesis.

Abstract

Mobile robots could be used to search, find, and relocate objects in many types of manufacturing operations and environments. In this scenario, the target objects might reside with equal probability at any location in the environment and, therefore, the robot must navigate and search the whole area autonomously, and be equipped with specific sensors to detect objects. Novel challenges exist in developing a control system, which helps a mobile robot achieve such tasks, including constructing enhanced systems for navigation, and vision-based object recognition. The latter is important for undertaking the exploration task that requires an optimal object recognition technique.

In this thesis, these challenges, for an indoor environment, were divided into three sub-problems. In the first, the navigation task involved discovering an appropriate exploration path for the entire environment, with minimal sensing requirements. The Bug algorithm strategies were adapted for modelling the environment and implementing the exploration path. The second was a visual-search process, which consisted of employing appropriate image-processing techniques, and choosing a suitable viewpoint field for the camera. This study placed more emphasis on colour segmentation, template matching and Speeded-Up Robust Features (SURF) for object detection. The third problem was the relocating process, which involved using a robot's gripper to grasp the detected, desired object and then move it to the assigned, final location. This also included approaching both the target and the delivery site, using a visual tracking technique.

All codes were developed using C++ and C programming, and some libraries that included OpenCV and OpenSURF were utilized for image processing. Each control system function was tested both separately, and then in combination as a whole control program. The system performance was evaluated using two types of mobile robots: legged and wheeled. In this study, it was necessary to develop a wheeled search robot with a high performance processor. The experimental results demonstrated that the methodology used for the search robots was highly efficient provided the processor was adequate. It was concluded that it is possible to implement a navigation system within a minimum number of sensors if they are located and used effectively on the robot's

body. The main challenge within a visual-search process is that the environmental conditions are difficult to control, because the search robot executes its tasks in dynamic environments. The additional challenges of scaling these small robots up to useful industrial capabilities were also explored.

DECLARATION

I certify that this thesis does not, to the best of my knowledge and belief:

(i) incorporate without acknowledgement any material previously submitted for a degree or diploma in any institution of higher education;

(ii) contain any material previously published or written by another person except where due reference is made in the text; or

(iii) contain any defamatory material.

I also grant permission for the Library at Edith Cowan University to make duplicate copies of my thesis as required.

Acknowledgements

I would like to acknowledge many people for their role in the completion of this thesis. I want first to express my gratitude to my supervisors, Dr Ganesh Kothapalli and Dr Majid Tolouei-Rad, for their guidance and support during my studies. I am grateful for the time and attention they have so generously given. I also want to thank Dr Greg Maguire for his constructive proofreading, and for helping me to produce a thesis I can be proud of.

I would also like to thank all of the Engineering School employees, my other colleagues and friends for the long and wonderful conversations about our studies. I am grateful for listening to my study problems even though, for most of them, my research topic is not even close to theirs. I enjoyed the various discussions we had and I have learnt a lot from them.

Financial support for my graduate studies has been provided by Ministry of Higher Education and Scientific Research, Iraq. Without this help and support this study journey would not have been possible.

Above all I wish to express my gratitude to my family for the unbelievable amount of support and love shown to me throughout my graduate school journey. Especially, I would like to thank my mother for always supporting my choices in life; my wife who has done whatever she could to make me happy; and my brother, Mohammed, for always supporting.

Finally I would like to dedicate this work to friends who I will never see again.

Table of contents

Chapter 1 Introduction

Mobile robots have been used in various applications including: manufacturing, mining, military operations and search and rescue missions. As such, the robot interacts with many tools and other types of equipment and therefore, must model its environment, control its motion and identify objects by using the navigation system and manage assigned tasks with its control system [1]. A robot's navigation system controls three functions in real-time: path planning [2], self-localisation [3] and motion control [4, 5]. The first is the process of finding an optimal path for movement from the start point to the destination while avoiding obstacles. The second is the robot's ability to determine its position and orientation within its environment. The third is the essential task of enabling the robot to operate in its environment [5]. The assigned tasks enable mobile robots to perform specific useful duties within its environment, such as grasping and relocating objects.

There are many types of manufacturing operations and environments for which mobile robots can be used to search, find and relocate objects [1]. In this role, the robot explores its environment to learn the location of a specific object and then performs a useful task, such as moving the object to another place. Such robots will need enhanced systems for navigation and vision-based object recognition. The navigation system is important for generating a path that covers the entire environment and for locating the position of the robot within that environment. It must also identify all potential obstacles in order to select a suitable path towards the desired destination in real-time [6]. Vision-based object recognition is important for undertaking the exploration task; it involves using a vision sensor and employing an optimal object recognition technique.

1.1 Problem Statement

Recently, autonomous robots have been considered for service applications. Service robots can include intelligent wheelchairs or vacuum cleaners and medicine or food delivery robots. However, for search services, robots must recognise specific objects, which they may then be required to approach, grasp and relocate. Novel challenges exist in developing a control system that helps a mobile robot to navigate and search its

environment. These include constructing an optimal navigation system that enables the mobile robot to search the entire area, because the target might have an equal probability of being at any location. As the robot performs a visual search, the choice of an exploration strategy and vision-based object recognition technique is difficult. The search strategy that directs the robot to move to a search site and to relocate as required, involves selecting the vision sensor's viewpoint. One aspect is the object detection process, which is challenging because the robot needs to navigate and place the object in the field of view. The robot also requires a vision system that employs some image analysis techniques, which are sensitive to environmental conditions, such as lighting, texture and background colour. In a classical path planning process, the robot is aware of the start and target locations. However, in a search robot application, the target position is unknown and therefore, the exploration path should cover the entire area and maximise the probability of detecting the target object. Lastly, the robot needs to manipulate the detected object by implementing specific task codes.

1.2 Research Questions

- The problem statement identified the challenges in constructing a navigation system for an indoor search robot. Part of this research involves designing systems that are capable of overcoming the challenges. The general question is: How should a self-navigating mobile robot control system be designed?

- Another aspect of the research is: How should the different types of sensors be integrated within the control system for the search robot?

- The accuracy of a measurement system will dramatically rise if the robot is equipped with the high number of sensors. However, this increases the robot's price and leads to a more complex control system in its implementation [7]. Therefore, the number of sensors must be reduced without affecting the efficiency of the robot motion. The question is: How should a minimum number of sensors be attached on the robot's body for autonomous navigation?

- Although there are many options, another important question is: How should vision technology be integrated with robot technology for autonomous navigation? (This investigation is expected to lead to an optimal mobile robot navigation system.)

- Which algorithms should a mobile robot use to search and locate objects in the visual field?

- How can a robot be enabled to perform its tasks in different terrains? (The research has the potential to identify limitations on the navigation system and path planning methods, due to terrain.)

- Which custom-built instruments are needed for the robot's navigation system, which is based on the vision system and range sensors, to optimally function in the intended simulation of an industrial environment?

- The issue of scalability with regard to the size of the robot must be addressed, as in general, this issue has not received sufficient attention by researchers. Thus, this poses the question: Is the legged robot used in this research scalable for use in industrial tasks?

- Whilst no large-scale robots will be designed in this project, the question to be addressed is: What are the theoretical challenges in scaling the model robots used to a size useful to industry?

1.3 Aims and Significance

The need for a completely autonomous mobile robot has stimulated considerable research on a navigation system technology. The aims of this research are:

- To develop a mobile robotics system that is suitable for a search robot that works autonomously in unknown indoor static environments.

- To develop an efficient way to identify the location of orientation of robot for an effective control mechanism.

- To find a suitable method for locating a minimum number of sensors on the robot's body that is ideal for autonomous navigation.
- To find a suitable image processing technique that is optimal for object detection within robot exploration applications.
- To find the optimal exploration path that covers the entire environment for a search robot.
- To find an ideal strategy for searching the designed environment.
- To construct a motion control system that employs a camera and range and force sensors.

1.4 Methodology

The development task for the robot control system, to enable the robot to search for, find and relocate the target (object), is divided into three parts. First, the exploration (navigation) task includes finding a suitable exploration path that covers the entire environment with minimal sensing requirements and then constructing this path for the robots. Second, the visual-search task involves finding appropriate image processing algorithms that are suitable for object detection and then implementing and assessing them with the robots. This also includes rotating and then choosing a suitable viewpoint field for the camera. Third, the relocating task consists of using a robot's gripper to grasp the detected desired object and move it to the assigned final location. The relocation process also involves approaching both the target and the delivery site, using a visual tracking technique.

As mentioned above, one of the main objectives of this work is to address the theoretical challenges posed in scaling model robots to an industrially useful size. It would also be worth studying two types of mobile robots having two philosophies of locomotion configurations to identify limitations on the navigation system. Therefore, two types of mobile robot that have different software and electronic modules are used to test the functionality of the control system. First, an existing hexapod mobile robot is employed. The sensor platform for the robot is designed and constructed to enable the

robot to navigate within its environment. Second, the wheeled robot that is also designed and constructed as part of the project is used to validate the methodologies used. All codes implemented are written in C++ and C programming languages. The OpenCV and OpenSURF libraries are employed for image processing. Each control system function is tested separately and then in combination as a whole control program.

1.5 Structure of the Thesis

The next chapter comprises the background and a literature review. Chapter Three describes the hexapod mobile robot used in this work. Chapter Four presents the wheeled robot construction steps in detail. In Chapter Five, the implementation of the sensory platforms for both robots is explained. It also describes the process of following the exploration path in detail. Chapter Six demonstrates the object detection algorithms and presents the results. The viewpoint field of the camera is described in Chapter Seven, which also shows the combination of all codes and results. Chapter Eight summarises and discusses the presented work. Here, the ideas and possibilities for future research are also presented.

Chapter 2 Background and Literature Review

2.1 Background

The first industrial robot was developed and used in industry by General Motors in 1961. Since then industrial robots have been widely used in manufacturing settings for performing various tasks; especially repetitive, heavy and hazardous processes. Typically, industrial robots are fixed and designed to work within a limited operating range. More recently, mobility has been added to industrial robots, which means the robot can perform the same tasks in different locations. In this scenario, industrial robots have to work autonomously and thus, they must be equipped with the required tools to allow them to explore their environment in order to carry out appropriate tasks. In 1969, a mobile robot (SHAKEY) was developed by Stanford University as the first robot that could control its mobility; in this case, to navigate through office buildings [8]. In subsequent decades, the design of mobile robots and their navigation systems underwent rapid development as more researchers joined this field [9]. Not surprisingly, 'The World Robotics Report produced by the United Nations Economic Commission for Europe predicts massive growth in the robot industry over the next decade' [10].

2.1.1 Mobile Robot Mechanisms

Mobile robots can be classified into three categories depending on their ground locomotion configuration: wheeled, legged and articulated [11]. Each type includes specific characteristics that make them appropriate for particular classes of applications. Typically, wheeled robots use rotational devices in their motion, such as wheels and tracks. They usually have simple mechanisms, a low total weight and are fast and efficient when they move on structured, regular surfaces. Therefore, they are utilised in almost all industrial applications. However, they are inefficient on very soft or rough surfaces, such as outdoor, unpaved terrains. For instance, the wheeled robot consumes high energy when it wants to move on an uneven surface or over a small obstacle [11]. Accordingly, the other two types of robots are needed because more than half of the Earth's landmass is not accessible to existing wheeled and tracked vehicles [11, 12].

Characteristically, their biological counterparts have inspired the designs of legged and articulated robots. Legged robots [12] provide superior mobility in soft and unstructured terrains because they use discrete footholds. This consists only of point contacts with the ground for support and traction, whereas wheeled robots require a continuously supportive surface. However, they have some limitations, such as low speeds, complex control, high weight and large energy consumption. Articulated robots consist of several segments that are gathered and connected in such a way as to imitate a snake [13] or a centipede [14]. The main benefit of these types of construction is their ability to move along and across irregular terrains and narrow passages.

2.1.2 Mobility and Autonomy

Mobility is the ability of robots to move freely from one location to another in an environment to perform their tasks. If the movement is controlled remotely by an operator, the robot is called non-autonomous [3]. Conversely, the autonomous robot assesses its environment by using various sensors. The sensors' measurements are employed to control the robot's motion without any operator intervention other than for the provision of the assigned tasks.

2.1.3 Robot Navigation

Robot navigation is the ability of the autonomous mobile robot to plan its motion in real-time and to navigate safely from one place to another. The robust navigation process requires three aspects, namely: path planning, self-localisation and motion control.

- Path planning is the process of finding an optimal path from a start point to the target location without any collisions [15].
- Localisation means that the robot estimates its position relative to specific objects within the environment [3].
- Motion control is the robot's ability to transfer sensory information into accurate physical movement in a realistic world [16].

The process of robot navigation is a complex, technological problem as it determines a robot's autonomy and reliability in performing assigned tasks; it has been widely

researched since the 1970s [17]. Whilst many solutions and techniques have been proposed, the navigation problem remains challenging. This is not because of limited navigation algorithms but because of the requirement for robust and reliable methods to acquire and extract environmental information, which is then automatically related to the navigation map [18]. Negenborn [3] described a further three additional problems in robust robot navigation: limits in computational power (CPUs); difficulties in detecting and recognising objects; and complexities in obstacle avoidance.

2.1.4 Examples of Application Areas

Typically, mobile robots are developed to replace human beings in hazardous work or relatively inaccessible situations, such as: exploration of nuclear power plants [19], undersea areas [20] and space missions [21]. Another potential application is search and rescue for lost or injured people where the robot must explore the entire searched area. Such robots are typically controlled remotely by the rescue team [22]. Recently, autonomous robots have been considered for service applications. Service robots can include intelligent wheelchairs or vacuum cleaners and medicine or food delivery robots. However, for search services, robots must recognise specific objects, which they may then be required to approach, grasp and relocate. The target objects might occur with equal probability at any location in the environment; therefore, the robot must navigate and search the whole area autonomously and be equipped with specific sensors to detect the objects. The next section is a literature review on the required attributes of search robots.

2.2 Literature Review

The most important three aspects required of a mobile search robot are: navigation (exploration path), target finding and control of a vision sensor. The former is carefully planned to cover the robot's entire environment while taking account of the visibility of the target and optimising both navigation time and collision avoidance [23]. The navigation system must help the robot approach and observe the target efficiently through optimal object recognition techniques; typically using vision sensors supported

by image processing techniques. The control of the vision sensor includes selection of the camera's viewpoint.

2.2.1 Navigation: the Exploration Path

A search robot navigates in an environment that typically has a starting point, a target object and a number of obstacles of random shapes and sizes. As such, the starting point is known whereas the target position is unknown. The robot moves from the starting point with the objective of finding the target. The robot must find an obstacle-free, continuous path that covers the entire environment. It should also localise itself within the environment and be aware when the search process is accomplished.

2.2.1.1 Navigation Strategy

Navigation strategies differ depending on whether the environment is static (static obstacles) or dynamic (static and dynamic obstacles) [24]. Both categories can be subdivided into unknown and known environments. In the latter, information is provided on the location of obstacles before motion commences. Across the various environments, there are many navigation algorithms that address the robot navigation problem [17]. All navigation planning algorithms assume that the mobile robot has detailed knowledge of the start and target locations and thus, of the direction between them, so that it can find an optimal path between these two locations and avoid obstacles. Some algorithms require extra environmental information or even a comprehensive map. According to Zhu, *et al.* [25], navigation algorithms are classified into global and local planning.

Global navigation planning

The global navigation algorithms plan the robot's path from the start to the goal by searching a graph that represents a map of the global environment. The environmental graph is constructed either off-line or on-line. In the former, the comprehensive map is initially loaded into the robot and then the navigation algorithm determines the optimal path before the robot commences its motion. For instance, Jan, *et al.* [15] presented some optimal path planning algorithms suitable for searching an environmental

workspace within an image. The view of the environment is divided into discrete cells, one of which is the robot. This method can be criticised for making use of a camera at a fixed position. Similarly, Huiying, *et al*. [26] combined the Voronoi diagram with the Dijkstra algorithm to easily find an optimal path for a robot. This off-line method assumes that the robot moves in a static environment and it has a precise motion system to satisfy the navigation conditions. These statements are unrealistic for the actual robot and therefore, this method is used rarely in robot navigation.

Conversely, in the on-line technique, although the environmental map is loaded into the robot, the navigation algorithm continues updating it by using the robot's sensors. This method enables the robot to navigate in dynamic environments and to correct continually its location within the map. For example, a navigation algorithm that integrates the A* search algorithm, the potential field method and the Monte Carlo localisation (MCL) method was explained in [27]. A visibility graph was generated using a camera and image processing. The A* search algorithm was then used to perform global path planning, while the potential field method was used to avoid the obstacles. The MCL algorithm continuously updates the robot's steps in the environment. Nguyen Hoang, *et al.* [28] introduced a multi-ant colony algorithm that successfully found the optimal path and avoided round obstacles in a simulation. Both of these strategies [27, 28] supported concurrent examination of all environmental information. However, recalculating the path in response to a change in the environment incurs an extremely high computational cost. Typically, the global planning methods have three intrinsic drawbacks: they are expensive to compute, complex to construct and it is difficult to obtain an accurate graph model.

Research on modelling environments and achieving exploration paths for mobile search robots has generally relied on global navigation planning. For instance, Fukazawa, *et al.* [4] proposed a points-distribution, path-generation algorithm in which the robot is given a set of points that completely cover the environment. The robot in that study sought the shortest path that encompassed all the points and it kept looking for an object while it moved along the path and then once found relocated it. The authors assumed that the robot had a complete map of the environment. They also argued that three types of path planning algorithm could cover the entire environment in exploration applications: the random walk, the spiral path and the zigzag path. The authors considered that the

random walk could not guarantee accomplishment of the exploration task. The other two techniques generate the exploration path by joining line segments arranged in the environment. Clearly, the computational cost for creating a path increases with the total number of line segments.

Another study proposed an efficient approach for modelling the search path by minimising the expected time required to find the target [29]. The assumptions made in that work were: that the mobile was equipped with efficient sensors, that the environment containing the object was completely known and that the motion strategy enabled the robot to find the target quickly. The known environment in [30] was divided into a set of regions for the robot that was used to search for multiple targets. The robot's task was to discover the sequence of motions that reduced expected time to find the targets. However, the authors in [29, 30] did not describe how the robot recognised and discovered the objects. Furthermore, these studies were simulations and did not involve a robot.

Some researchers have tried to avoid constructing a comprehensive environmental map. Tovar's [31] robot used critical events in on-line sensor measurements, such as crossing lines, to build a minimal representation that provided a sensor feedback motion strategy. The authors introduced a visibility tree, which represents simply-connected planner environments, to dynamically encode enough information for generating optimal paths. Another study [32] presented a guide tracking method in which the mobile robot is provided with a trail from a starting point to the target location. The benefit of a trail is that the mobile robot reaches the target location with little requirement for autonomous navigation skills. However, the trail needs to be shaped prior to the robot navigation process.

Local navigation planning

Local navigation algorithms directly use the sensors' information in the commands that control the robot's motion in every control cycle, without constructing a global map [25]. Therefore, these algorithms are employed to guide the robot in one straight path from the start point to the target location in unknown or dynamic environments. While the robot navigates, it avoids obstacles that are in its path and keeps updating the

significant information, such as the distance between its current location and the target position. Typically, the local navigation algorithms are easy to construct and optimal for real-time applications.

A potential field algorithm [33] is widely used within the local navigating technique. It is constructed by creating the artificial potential field around the robot. The target position's potential attracts the robot while the obstacles' potential repulses it. As the robot moves toward the target, it calculates the potential field and then determines the induced force by this field to control the robot's motion. Typically, the robot moves from a higher to a lower potential field. The optimal potential field is constructed so that the robot is not trapped into a local minimum field before reaching the target but it is impossible to create such a field [34]. Therefore, this method is combined with other navigation algorithms to increase its efficiency, as in [27].

The Bug algorithms [2], which are well-known navigation methods, are relatively efficient as they solve the navigation problem by saving only some points of the path curve and do not build full environment maps. As such, they are identical to the local planning techniques because they only need local environmental information but the robot needs to learn little of the global information. If the robot discovers that no such path exists, that is, a local minimum, the algorithms will terminate its motion and report that the target is unreachable. The authors [2], who compared eleven members of this family, claimed that these techniques presume the robot to have perfect localisation ability, perfect sensors and no size (point object). Consequently, the algorithms are not used directly for realistic robot navigation. The Bug movement strategies are appropriate for a robot that is designed to navigate in an unknown environment that is constantly changing [2, 25].

2.2.1.2 Robot Localisation

Robot localisation is the robot's ability to estimate its location relative to specific aspects within its environment, using whatever sensors are available. This process can be either relative localisation or absolute localisation [3, 35].

Relative localisation

In relative localisation, the robot calculates its current position relative to the previous locations, trajectories and velocities over a given period. As such, the robot requires knowledge of its initial location before it can continue determining its current location based on the direction, speed and time of its navigation [3]. The odometry method is widely used to measure the relative position because of its low cost and easy implementation. This method is implemented by using wheel encoders that count the revolutions of each wheel and an orientation sensor, such as electromagnetic compass that calculates the robot's direction (see [35]). Because the robot measures its distance based on the start location, any error in the measurements resulting from the drift or slippage of the wheels will compound over time.

Absolute localisation

In the absolute localisation method, the robot estimates its current position by determining the distance from predefined locations without regard to the previous location estimates [35]. Therefore, any error in the localisation measurement does not increase. This method usually employs landmarks to estimate the robot's location. Landmarks are classified into active and passive landmarks. The former can be satellites or other radio transmitting objects and they actively send out information about the location of the robot. This has the advantage that the robot does not require prior information about the environment. However, the active landmarks' signals might be disturbed before being received by the robot and this will cause errors in the measurement [3]. The Global Positioning System (GPS) is frequently used to measure the absolute position of robot that use active landmarks (see [36, 37]). The passive landmarks do not send signals as active landmarks do but they must be actively seen and recognised by the robot in order for it to determine its location [18]. Landmark recognition depends on the type sensors used.

2.2.2 Computer Vision for Mobile Robot

The availability of low cost, low power cameras and high speed processors, are the main reasons for the rapid development of image sensor applications [38]. Computer vision

relies on visual sensors that can extract relatively large amount of environmental information from an image [39]. Consequently, there has been intensive research on computer vision for mobile robot navigation since the early 1980s, as indicated in a survey of developments in this field [9]. The extracted information is provided to a robot's controller, which then dictates the robot's motion. In the case of the search robot, the main objective of image processing is to detect the target object.

The literature review is divided into object recognition and vision-based mobile robot navigation. The former is the process of detecting, recognising and extracting object information, whereas the latter concerns the use of this information for robot navigation.

2.2.2.1 Object Recognition

In the very active field of research of computer vision [40], the techniques being used to detect and recognise an object in an image, include: image segmentation, template matching, Scale Invariant Feature Transform (SIFT) and Speeded Up Robust Features (SURF).

Image segmentation scheme using colour image model

Image segmentation is one of the basic techniques in computer vision. It is an analytical process, which recognises image content based on variations in colour and texture. RGB colour space, in which each colour involves three weights: red, green and blue, has been commonly used in the segmentation process to detect the target object [41, 42]. Other colour descriptors, such as the HSI colour space [43, 44] and dominant colour descriptor (DCD) [45] can also be used. Lin, *et al.* [46] developed a real-time algorithm that allows a mobile robot to detect and track a moving object by utilising adaptive colour matching, as well as a Kalman filter. The RGB colour space is used for object recognition, whilst the Kalman filter is used to estimate the object's position and velocity. Browning and Veloce [47] proposed a new four-step image segmentation algorithm to detect objects in indoor and outdoor environments. First, a soft segmentation is applied to label the image pixels by colour class. Next, a hard threshold is applied to distribute the image pixels to areas that belong to a colour class of interest

or not. Then, the areas that are similarly labelled are revealed and connected. Finally, the relevant object is detected and recognised in the image.

Template matching

Template matching is a well-known technique to find a small part R, of an original image I, that matches a template image T [48]. The dimensions of the template must be smaller than the dimensions of I. The matching is done by sliding and comparing a given template with windows of the same size in the image I, to identify the window R that is most similar to the template. The location of R(*x, y*) in I, which is defined as a pixel index of the top-left corner of R in I, points to the location of the closest match as measured by a correlation number (Figure 2.1). The accuracy of the template matching process depends on the algorithm used for measuring the similarity between the template and the original image [49]. Matching tolerance against various image distortions that might occur during the process of acquiring the images, such as rotation, scaling and changed environmental lighting, is the major challenge with this method [50]. The accurate matching process is also achieved by selection of the optimal templates, which must present "a highly detailed and unique region" [49].

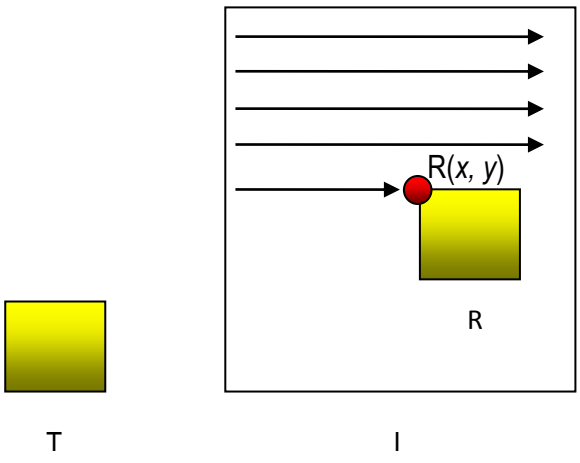


Figure 2.1. Template matching technique

Template matching has been intensively researched with results reflecting the algorithms used. For instance, Omachi, *et al.* [51] proposed a template matching algorithm that can efficiently identify the portion of the input image that is similar to the template; it was deemed efficient because processing time was very short and computational costs were reduced. Do and Jain [52] presented a template matching

algorithm that recognises objects in two stages: pre-attentive and attentive. The former is a fast process used to find regions of interest that are more predictable for detecting the target object in them. In contrast, the latter is the process of detecting and recognising the target object within the selected regions of interest found in the first stage.

Scale Invariant Feature Transform (SIFT)

Scale Invariant Feature Transform (SIFT), which is a well-known technique in computer vision, was initially presented by Lowe [53] in 1999 and has been widely used to detect, recognise and describe local features in an image. SIFT can extract an object based on its particular (key) points of interest in an image with scaling, translation and rotation. A SIFT algorithm consists of four major steps: scale-space extrema detection, key point localisation, orientation assignment and key point description. The first step employs a difference-of-Gaussian (DoG) function, that is $D(x, y, \sigma)$, to specify the potential interest points that are invariant to scale and orientation. This is done by applying Equations 2.1 to 2.3; the results are shown in Figure 2.2A. Then, each pixel is compared with its neighbours to obtain maxima and minima of the DoG (Figure 2.2B). The pixels that are larger than all or smaller than all of their neighbours are chosen as potential interest points.

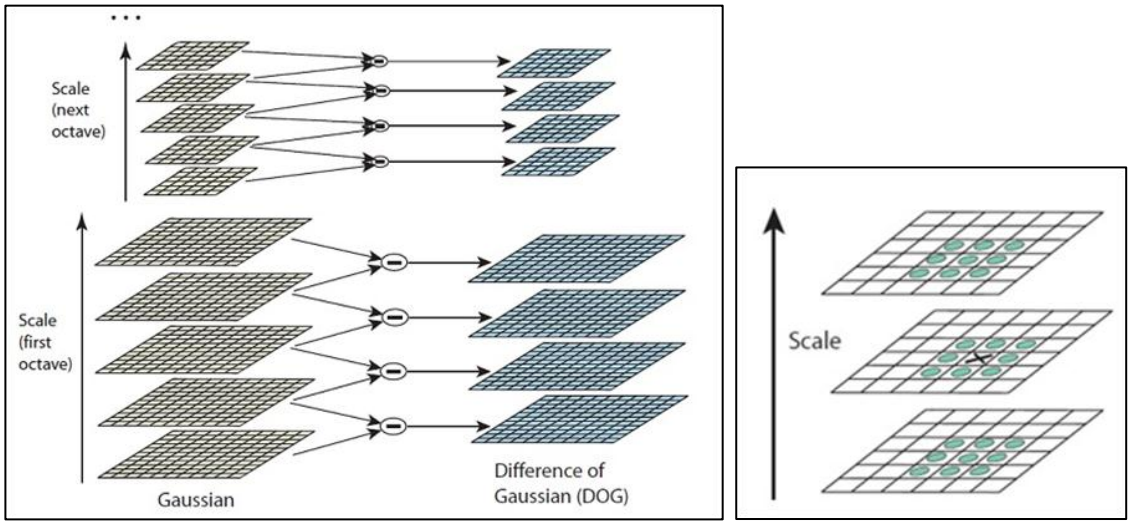
$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.1}$$

where

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2 + y^2)/\, 2\sigma^2} \tag{2.2}$$

$$D(x, y, \sigma) = [G(x, y, k\sigma) - G(x, y, \sigma)] * I(x, y) = \; L(x, y, k\sigma) - L(x, y, \sigma) \tag{2.3}$$

in which I is an image; (x, y) are the location coordinates; σ is the scale parameter (the amount of blur); the * denotes the convolution operation in x and y; k is a multiplication factor that separates two nearby scaled images; G is the Gaussian Blur operator; and L is a blurred image.



(A) (B)

Figure 2.2. from [54] where Figure 2.2A represents the building of the Gaussian and DoG pyramid. Figure 2.2B represents the comparison of each pixel (i.e., the pixel marked with X) to its 26 neighbours (the pixels are marked with circles; and they are in 3 × 3 regions at the current and adjacent scales) to find the maxima and minima of the DoG images.

Key point localisation executes a detailed fit to the nearby data for location, scale and ratio of principal curvatures, in order to reject low contrast points and eliminate the edge response. This was achieved by using a Tyler expansion of the scale-space function $D(x, y, \sigma)$ in Equation 2.4. Then, unstable extrema with low contrast are rejected using the function in Equation 2.6. Finally, a 2 × 2 Hessian matrix *H* (Equation 2.8) was used to eliminate the edge response.

$$D(X) = D + \frac{\partial D^T}{\partial X} X + \frac{1}{2} X^T \frac{\partial^2 D}{\partial X^2} X \tag{2.4}$$

where $X = (x, y, \sigma)^T$ (2.5)

$$D(X_{\max}) = D + \frac{1}{2} \frac{\partial D^T}{\partial X} X_{\max} \tag{2.6}$$

where $X_{\max} = -\frac{\partial^2 D^{-1}}{\partial X^2} \frac{\partial D}{\partial X} = 0$ (2.7)

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{2.8}$$

Orientation assignment collected gradient directions (Equation 2.9) and magnitudes (Equation 2.10) of sample points in the region that is around each key point and then the most prominent orientation in that region was assigned to the key point location.

$$\theta(x,y) = tan^{-1}\left(\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)}\right) \quad (2.9)$$

$$m(x,y) = \sqrt{(L(x+1,y)-L(x-1,y))^2 + (L(x,y+1)-L(x,y-1))^2} \quad (2.10)$$

In the final step, the descriptor of the local image region was computed to allow for significant levels of local shape distortion and change in illumination. SIFT was introduced into mobile robotics navigation systems in 2002 (see [55]). SIFT provides accurate object recognition with a low probability of mismatch but it is slow and when illumination changes, it is less effective [56].

Speeded Up Robust Features (SURF)

A SURF algorithm, using an integral image for image convolution and a Fast-Hessian detector, was proposed by Bay, *et al.* [57]. First, the integral image representation of an image was created from the input image by using Equation 2.11. Then, the integral image was used within a Hessian matrix (Equation 2.12) to find an accurate vector of interest points. The interest points were localised by using a Tyler expansion of the scale-space function $H(x,y,\sigma)$ in Equation 2.13. Next, the interest points and integral image were employed to extract a vector of the SURF descriptor components of the interest points.

$$I_{\sum(x,\ y)} = \sum_{i=0}^{i\leq x}\sum_{j=0}^{j\leq y} I(x,y) \quad (2.11)$$

$$H = \begin{bmatrix} L_{xx}(x,y,\sigma) & L_{xy}(x,y,\sigma) \\ L_{xy}(x,y,\sigma) & L_{yy}(x,y,\sigma) \end{bmatrix} \quad (2.12)$$

$$H(X) = H + \frac{\partial H^T}{\partial X}X + \frac{1}{2}X^T\frac{\partial^2 H}{\partial X^2}X \quad (2.13)$$

in which I is an image; (x, y) are the location coordinates; σ is the scale parameter; $I_{\sum}$ is the integral image; $L_{xx}(x,y,\sigma)$ is the convolution of the second order Gaussian derivative $\frac{\partial^2 g(\sigma)}{\partial X^2}$ (Laplacian of Gaussian) and similarly for L_{xy} and L_{yy}.

SURF and SIFT techniques use slightly different methods of detecting an object's features in an image. The detectors in both calculate the interest points in the input image but they work differently. They use the interest points to create a descriptor vector that can be compared or matched to descriptors that were extracted from other images. SURF proved to be faster than SIFT and a more robust image detector and descriptor [56].

Each of above-mentioned methods has its limitations in detecting an object's features in an image [9]. Furthermore, there are three parameters that influence the object detection process, namely: environmental conditions, target characteristics and sensor efficiency [58]. Lighting, texture and background colour are the main environmental conditions. Sufficient texture and contrast features are the main target characteristics. Some researchers have combined and tested different image processing techniques to achieve better results [39]. For example, Ekvall, *et al.* [59] detected an object by applying a new method that combined SIFT and a colour histogram called the Receptive Field Co-occurrence Histogram (RFCH) [60]. First, an image of the environment was captured without an object being present and then the operator placed the object in front of the camera. The object is then separated from the background by using image differentiation. Their experimental results showed that this method is robust to changes in scale, orientation and view position.
In robot applications, object location and orientation relative to the robot have to be calculated and used to effect the robot's motion [61]. The authors compared geometrical moments and the features from Eigen-space transformation for determining object characteristics in the image. The former was less susceptible to noise.

2.2.2.2 Vision-based Mobile Robot Navigation

Developments in mobile robot navigation based on a vision system can be divided into indoor and outdoor navigation [9]. The former can then be divided into map-based, map-building-based and mapless forms of navigation. The first relies on a sequence of landmarks, which the robot can detect for navigation, whereas the second involves sensors to construct the robot's environment, so that it forms an internal map for navigation. Finally, mapless navigation is based on observing and extracting

information from the elements within the robot's environment, such as walls and objects, before it is used for navigation.

Conversely, outdoor mobile robot navigation can be divided into structured and unstructured environments. These will cover obstacle avoidance, landmark detection, map building and position estimation. A structured environment requires landmarks to represent the robot's path, whereas in an unstructured environment there are no regular properties, so a vision system must extract possible path information. For example, to find a path in outdoor robot navigation, Blas, *et al.* [62] proposed an on-line image segmentation algorithm whose framework combined colour and texture segmentation to identify regions that share the same characteristics as the path. Lulio, *et al.* [63] applied a JSEG segmentation algorithm for an agricultural robot, so that it could classify an image into three areas: planting area, navigable area and sky. The image segmentation method was performed in three stages: colour space quantification, hit rate region and similar colour region merging.

Visual tracking

Visual tracking is a crucial research area [64] because it is involved in many robot applications, such as navigation and visual surveillance [39]. It consists of capturing an image by a camera, detecting a goal object in the image by image processing and guiding the robot automatically to track the detected object [48]. For indoor robot navigation, tracking is widely used for service robots [44]. For example, the robot used by Abdellatif [44] tracked by following a coloured target. Colour segmentation was applied to recognise the object and then the target's location was determined. In addition, a camera with three range sensors was used to detect obstacles and target distances. The camera and range sensors outputs were used as inputs for a controller, which enabled the mobile robot to follow the object while avoiding obstacles. Abdellatif's work was limited to using a single colour for target detection. Furthermore, there was no option available to the robot if the object was not detected in the current view. Medioni, *et al.* [65] also presented a robot navigation system that enabled a service robot to detect and track a human face or head, based on skin-coloured pixels, image intensity and circle detection in the image.

Landmarks

Landmark recognition has been widely researched because landmarks enable a robot to perform tasks in a human environment [40, 65] by helping the robot to navigate and localise its position. Landmarks can be natural and artificial; the former employs natural symbols, such as trees, while the latter has specially designed signs, which are located frequently in indoor environments to allow easy recognition. When more than two landmarks are used to localise the robot, it might need to use either the triangulation or trilateration methods [3]. The former uses distances and angles, whereas the latter only employs distances to calculate the robot's position and orientation.

Some researchers have designed and implemented landmark recognition systems. For instance, a landmark detection and recognition system proposed by [40] involved detection of landmarks in a captured image, segmenting of the captured image into smaller images and recognition and classification of the landmarks by using colour histograms and SIFT. Another study featured a visual landmark recognition system that combined an image processing board and genetic algorithms for both indoor and outdoor navigation [66]. The system can detect and evaluate landmarks that are pre-defined in the system's library within the real-time image. Some researchers have used environmental features as landmarks. For example, Zhichao and Birchfield [67] explained a new algorithm that detects door features, such as colour, texture and intensity edges from an image. The extracted door information was used as a landmark for indoor mobile robot navigation. Murali and Birchfield's [68] robot always performed straight line navigation in the centre of a corridor by keeping a ceiling light in the middle of the image but this greatly restricted its motion.

2.2.3 Vision-Sensor Control

The control of the vision-sensor includes two stages: 'where to look next' and 'where to move next' [69]. In the former, while the camera is fixed in the current position, all its configurations, such as zooming, are examined one by one. This strategy will be inefficient if the number of the camera's setting is large or image processing consumes considerable time. Therefore, the authors in [69] introduced a 'best-first' strategy in which all the camera's configurations are examined in the start location before the

navigation process is executed. The authors claimed that if this strategy is applied first and the target is detected, then time and effort involved in the searching process will be saved. The robot in that work was equipped with a stereo camera that did not have a zoom capability and the authors set the largest and closest distances between the robot and the target. If the target is not detected within the current viewpoint, the second stage (‘where to move next’) is performed and the next optimal viewpoint is determined. The next position should be attained with a high probability of detecting the object.

2.3 Conclusions

Researchers have developed many techniques to analyse images and detect objects but there are also limitations in these techniques. Subsequently, researchers have combined some of these techniques to achieve better results. This thesis will place less emphasis on SIFT than on colour segmentation, template matching and SURF because the long processing time incurred with SIFT is a major limitation with on-line image processing. The robot in this work uses a vision system to detect a target (object) and then approach, grasp and relocate it to a final location that is specified by an artificial landmark. The robot will execute following and tracking processes while it approaches and relocates the target.

In terms of the exploration path, the navigation is planned either globally or locally based on the algorithm used. Most algorithms assume that the robot has sufficient knowledge about the start and goal locations; its task is to find the optimal path to connect these two locations. Most researchers who have worked with mobile search robots assume that the searched area is completely known. The robot task was to find the target; unfortunately, there was insufficient information about the tools the robot used to detect the target.

In this study, the target location is totally unknown and therefore, the robot should search the whole area. Thus, the exploration path must be planned to cover the entire environment. It is assumed that the searched area has boundaries that are completely known, whereas its internal configurations are unknown. The robot starts its motion from the start location and then follows the walls or obstacles. While the robot navigates it continues to search for the target; if it is found, the robot approaches, grasps and

relocates it to the start location. If it is not found, the robot keeps following the walls until the start location is reached again, where the robot terminates its motion. The Bug algorithms are adapted to achieve the proposed motion planning.

Chapter 3 Hexapod Mobile Robot

3.1 Introduction

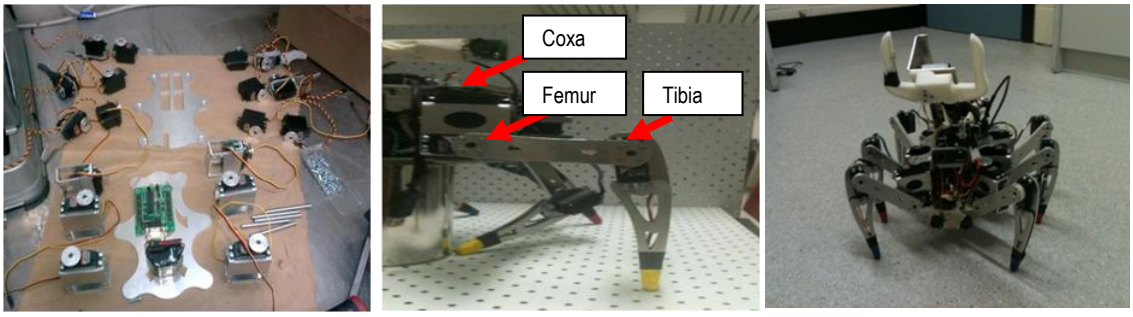
One of the main objectives of this work is to address the theoretical challenges posed in scaling model robots to an industrially useful size. This research has also the potential to identify limitations on the navigation system and path planning methods, due to terrain. As mentioned in Chapter 1, the functionality of the proposed control system will be tested using two types of mobile robots. The first is a legged mobile robot that is trained to search for, find and relocate a target object. The sensor platform for the robot is designed and constructed to enable the robot to navigate within its environment. The second robot is a wheeled one that will be designed, constructed and used to validate the methodology used.

Wheeled robots are re-used in most industrial applications, however, some objects may be dropped on the ground and obstruct the robot's motion. Even if these obstacles are small and the robot can navigate over them, the robot will consume high energy. Conversely, if the robot follows the obstacles' boundaries, this makes the navigation path and travel time longer. Wheeled robots are also inefficient on very soft or rough surfaces, such as outdoor, unpaved terrains.

Legged robots provide superior mobility on soft and unstructured terrains because they use discrete footholds. This consists only of point contacts with the ground for support and traction, whereas wheeled robots require a continuously supportive surface. They can also move over and overcome small obstacles more easily than wheeled robots. There are various types of legged robots classified by their number of legs; humanoid robots (two legs), tetrapod robots (four legs) and hexapod robots (six legs). This chapter will explain the configuration of the six-legged (hexapod) mobile robot, used in this work.

3.2 Mechanical Design

The robot's body comprises lower and upper legs, servo brackets and body bottom and top plates, all made from 3 mm thick aluminium sheet. The bottom and top plates of the body are separated by five separators, each 5 cm long. All these parts are shown in Figure 3.1A. The mobile robot has six legs and each of them has three rotary joints, namely: coxa, femur and tibia (Figure 3.1B), which provide three degrees of freedom. The joints are actuated by servo motors (see Appendix), which are able to provide up to 2.5 Nm of torque. The robot has a gripper driven by two servo motors for moving it up/down (by the 14 cm long arm) and closing/opening the 10 cm long jaws in order to grasp objects (Figure 3.1C).



(A) (B) (C)

Figure 3.1. The hexapod mobile robot structure

3.3 The Electronic System

Figure 3.2 illustrates the main parts of the electronic circuit used in the hexapod mobile robot.

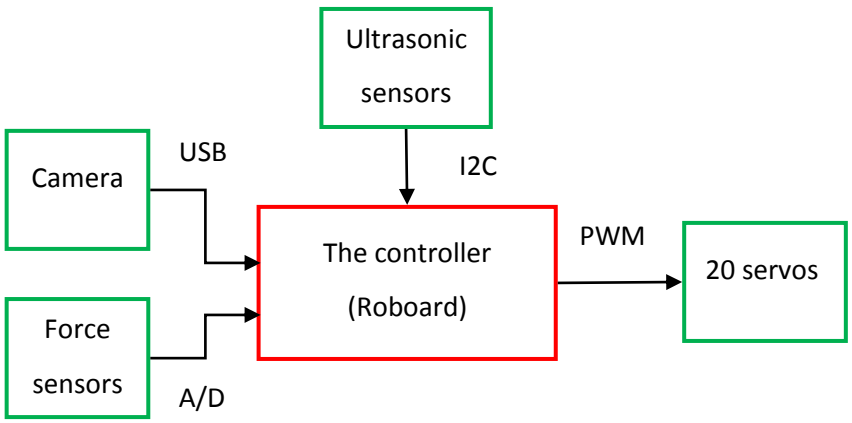


Figure 3.2. The hexapod electronic circuit

3.3.1 The Main Controller

The hexapod has 20 servos (three for each leg and two for the gripper); therefore, it needs microcontroller controls for simultaneous operation. The controller used is a Roboard controller RB-100 computer based [70], which has a Vortex86DX, a 32 bit x 86 CPU running at 1 GHz, and 256 MB of on-board memory, which consumes 400 mA at 6–24 V. This controller has I/O interfaces to the servo, DC motors, sensors and other devices and uses Open source C++ library code for Roboard's unique I/O functions. A Linux operating system is installed in the main controller. Figure 3.3 shows the Roboard (with dimensions 96 × 56 mm), controller's pins and features. The servo motors are controlled through pulse width modulation (PWM).

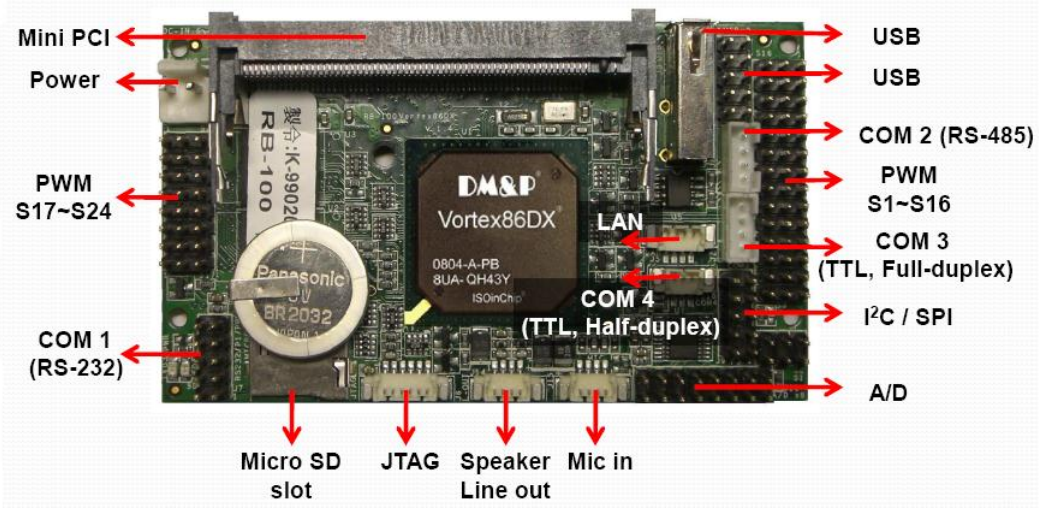


Figure 3.3. The main controller (Roboard)

3.3.2 Sensors

The main controller has 8 ports of analogue to digital convertors (A/D) and 24 digital ports (PWM). Accordingly, up to 8 analogue sensors and 24 digital sensors can be used simultaneously. It also has I^2C and SPI. The platform can support many types of sensors, which makes the robot scalable and suitable for many applications. The robot is equipped with the following sensors:

- Seven analogue tactile sensors; one sensor is in each leg and one in the gripper.
- Ultrasonic range sensors (Devantech SRF02) have been connected to the controller via the I^2C interface and placed at the front and sides of the robot.

- A camera has been positioned at the front of the robot.

The configuration of the sensors is explained in Chapter 5.

3.4 Power

The robot's power is provided by 6 × 1.2 volt cells connected in series. The main controller uses all the cells (7.2 volts), whilst the servos are joined to just 5 × 1.2 volt cells (Figure 3.4).

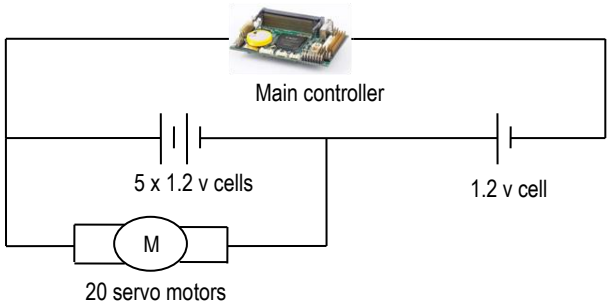


Figure 3.4. Power supply connection

3.5 Walking Control System

The forward and inverse kinematic equations are determined to establish the maximum and minimum alternating angular displacements through which the leg joints can move, as explained in (3.6). The hexapod home configuration (the reference joint angles), is also calculated at the beginning from the inverse kinematics. These angles are then used to implement the walking control system.

In this study, the robot requires a motion control system that provides two aspects: to move the robot forward and to rotate it about the central axis. The hexapod robot is programmed using the alternating wave gait ("4 + 2" gait) [71] for steering and the tripod gait ("3 + 3" gait) [71-73] for walking forward. In the wave gait, the robot walks forward by lifting and moving only two legs at a time. Figure 3.5 shows the four cycles of one robot step. The green parts indicate a state of motion and the brown parts show a state of rest; that is, the feet are on the ground to support the robot. The front two legs are raised and moved first (Figure 3.5B), followed by the middle pair (Figure 3.5C) and

then the back pair (Figure 3.5D). Once all legs have moved forward, the robot then moves its body forward (Figure 3.5E) to complete one step.

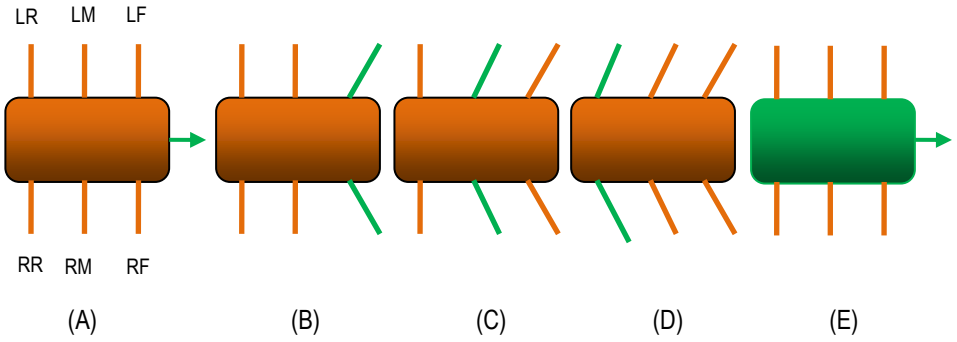


Figure 3.5. Wave gait of the hexapod

The robot can also rotate or change its direction by moving two legs at once. For instance, if the robot wants to rotate clockwise, it moves its front-left leg forward and rear-right leg backward, followed by the middle legs, then the rear-left leg and front-right leg (the left legs are all moved forward, whilst the right legs are all moved backward). The robot rotates its body once all the legs have completed their respective motions (see Figure 3.6). At any point of the motion in the wave gait, there are four legs or more in contact with the ground.

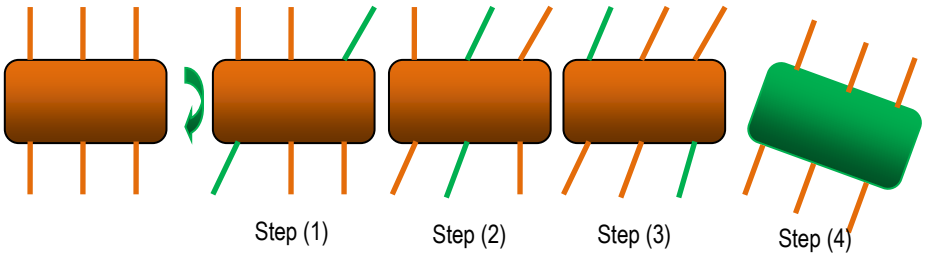


Figure 3.6. Steering using wave gait

In the tripod gait method, the robot walks forward by moving three legs at once, instead of the two legs as in the previous method. As such, the legs are divided into two groups of three; each group includes the front and back legs of one side and the middle leg of the opposite side. Each robot step has three cycles. First, the robot lifts and moves any set of legs (Figure 3.7B), followed by the other set (Figure 3.7C) and then by the robot body itself (Figure 3.7D). In this case, the robot can also rotate by moving three legs at

once. For instance, if the robot wants to rotate clockwise, it performs the same previous steps but the right legs are moved backward, whilst the left legs are moved forward. Then, the robot's body is rotated clockwise (Figure 3.8). The motion of the robot using the wave gait is slower than that using the tripod gait but it is more stable. The tripod gait method requires more leg coordination than the wave gait.

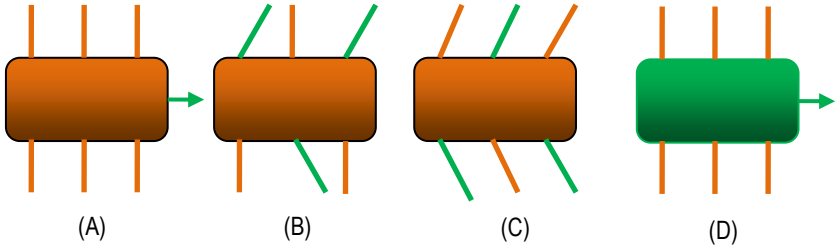


Figure 3.7. Tripod gait of the hexapod

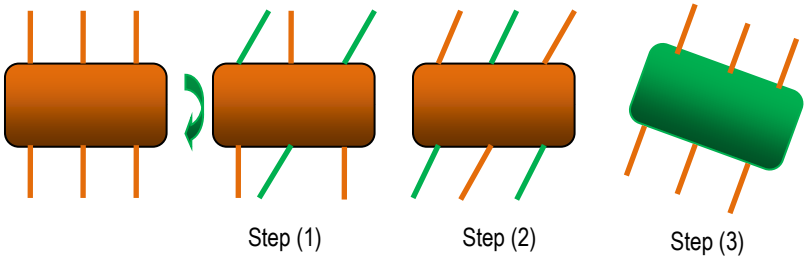


Figure 3.8. Steering using tripod gait

In each step, the robot starts by raising its legs from the ground by controlling the θ_2 and θ_3 angles and then they are moved forward by controlling the θ_1 angle (θ_1, θ_2 and θ_3 are specified in Figure 3.9). In this study, the maximum distance of one walking step, specified by θ_1, could be 7 cm. However, the robot is programmed to move with 5 cm as its maximum displacement to reduce the probability of legs colliding with each other. As a simple example, in order for the robot to move forward 50 cm at maximum speed, it will need 10 steps. The robot uses sensory feedback to control its walking steps and to correct motion error, as explained in Chapter 5.

3.6 Kinematic Modelling

Kinematic modelling describes the motion of the robot's leg joints without consideration of the forces or torques that cause the motion. The problem of the forward kinematics is to find the relationships between the joint variables of the individual leg, and the position and orientation of the foot of the given leg on the ground. Conversely, inverse kinematics is used to calculate the values of the joint variables, which represent the angles between the links of the individual leg [34].

The forward kinematic is specified by using the Denavit-Hartenberg (DH), which is a well-known convention for selecting joints' frames in robotic applications [34]. In this convention, each homogeneous transformation (A_i) that represents the position and orientation between the joints' (i and $i-1$) frames is given by the formula

$$A_i = \begin{bmatrix} C_{\theta_i} & -S_{\theta_i}C_{\alpha_i} & S_{\theta_i}S_{\alpha_i} & a_{\mathrm{i}}C_{\theta_i} \\ S_{\theta_i} & C_{\theta_i}C_{\alpha_i} & -C_{\theta_i}S_{\alpha_i} & a_{\mathrm{i}}S_{\theta_i} \\ 0 & S_{\alpha_i} & C_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

where the quantities a_i, α_i, d_i and θ_i are parameters associated with link i and joint i; and they are ith link length, link twist, link offset and joint angle, respectively. Note: a_i is the shortest distance between z_i and z_{i-1} measured along x_i; α_i is the angle between z_i and z_{i-1} measured about x_i; d_i is the distance along z_{i-1} from o_{i-1} to the intersection with x_i; and θ_i is the angle between x_{i-1} and x_i determined about z_{i-1}. In this thesis, the kinematics of a single three-joint leg located on the right side of the hexapod body will be derived (see Appendix for left legs). Figure 3.9 shows a graphical representation of a right leg that is either the right front (RF), right middle (RM) or right rear (RR) (Figure 3.5A). Note: the $z_{(i-1)}$ represents the rotation axis of the ith joint, while α_i specifies the change in the direction of the z_i axis relative to the direction of the $z_{(i-1)}$ axis and is determined about x_i.

First, the base frame $o_0\ x_0\ y_o\ z_0$ is established. The origin o_0 is placed at joint 1; o_0 can be located along z_0. The direction of the x_0 axis is first chosen arbitrarily and then the direction of the y_o axis that must achieve the right-hand rule is chosen. Next, the $o_1\ x_1 y_{\ 1}\ z_1$ frame is established at joint 2. The z_0 and z_1 axes are not coplanar; as such, the shortest line segment (a_1) that is perpendicular to both axes defines the x_1. The

length of the line segment (a_1) is determined from the coxa link. When θ_1 is equal to zero, the x_1 axis is parallel to the x_0 axis, as shown in Figure 3.9. However, the direction of the x_1 axis will change because θ_1 is variable. As z_1 and z_2 are parallel and in the same direction, α_2 and $d_2$will be zero in this case. The line segment between z_1 and z_2 is a_1 and it is determined from the femur link. Finally, the $o_3\, x_3\, y_3\, z_3$ frame is chosen at the end (foot) of the tibia link, as shown in Figure 3.9. In the case of revolute joints, all joint variables are angles, so that all d_i (link extensions) are zeros. The DH parameters for the right legs of the robot are shown in Table 3.1.

Table 3.1: *DH parameters of the robot leg on the right side*

Link	a_i	α_i	d_i	θ_i
1	a_1	$-\frac{\pi}{2}$	0	θ_1
2	a_2	0	0	θ_2
3	a_3	0	0	θ_3

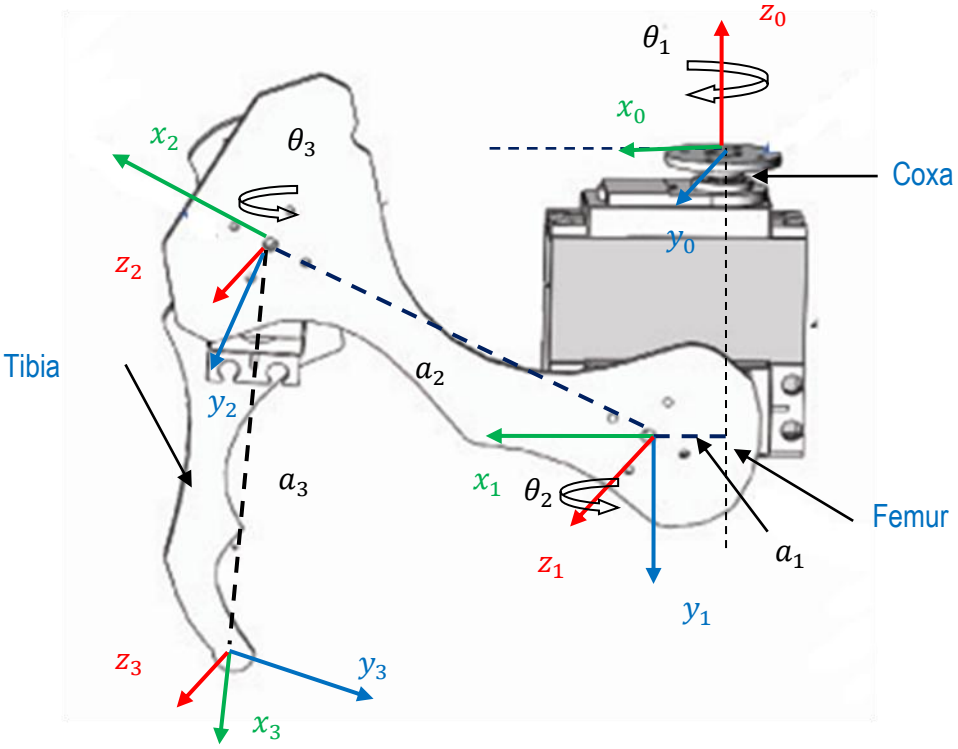


Figure 3.9. Graphical representation of the three-joint robot leg on the right side

The homogeneous transformation matrices (A_i) between the joints' frames are determined by substituting the DH parameters, which are in Table 3.1, in DH matrix, which is Equation 3.1 [34]. Performing the required calculations yields

$$A_1 = \begin{bmatrix} C_{\theta_1} & 0 & -S_{\theta_1} & a_1 C_{\theta_1} \\ S_{\theta_1} & 0 & C_{\theta_1} & a_1 S_{\theta_1} \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

$$A_2 = \begin{bmatrix} C_{\theta_2} & -S_{\theta_2} & 0 & a_2 C_{\theta_2} \\ S_{\theta_2} & C_{\theta_2} & 0 & a_2 S_{\theta_2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} C_{\theta_3} & -S_{\theta_3} & 0 & a_3 C_{\theta_3} \\ S_{\theta_3} & C_{\theta_3} & 0 & a_3 S_{\theta_3} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

in which S and C represent the sine and cosine functions, respectively. Then, the final transformation matrix (T_3^0) that represents the forward kinematics of the right leg is determined from Equation 3.3, which is represented as a dot product of three (A_i) matrices. Multiplying these together yields

$$T_{3(right)}^0 = \begin{bmatrix} R_3^0 & O_3^0 \\ 0 & 0 \end{bmatrix} = A_1 A_2 A_3 \tag{3.3}$$

$$T_2^0 = A_1 A_2 = \begin{bmatrix} C_{\theta_1} C_{\theta_2} & -S_{\theta_2} C_{\theta_1} & -S_{\theta_1} & a_2 C_{\theta_1} C_{\theta_2} + a_1 C_{\theta_1} \\ S_{\theta_1} C_{\theta_2} & -S_{\theta_1} S_{\theta_2} & C_{\theta_1} & a_2 S_{\theta_1} C_{\theta_2} + a_1 S_{\theta_1} \\ -S_{\theta_2} & -C_{\theta_2} & 1 & -a_2 S_{\theta_2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{3(right)}^0 = \begin{bmatrix} C_{\theta_1} C_{(\theta_2+\theta_3)} & -C_{\theta_1} S_{(\theta_2+\theta_3)} & -S_{\theta_1} & C_{\theta_1}(a_1 + a_2 C_{\theta_2} + a_3 C_{(\theta_2+\theta_3)}) \\ S_{\theta_1} C_{(\theta_2+\theta_3)} & -S_{\theta_1} S_{(\theta_2+\theta_3)} & C_{\theta_1} & S_{\theta_1}(a_1 + a_2 C_{\theta_2} + a_3 C_{(\theta_2+\theta_3)}) \\ -S_{(\theta_2+\theta_3)} & -C_{(\theta_2+\theta_3)} & 0 & -a_2 S_{\theta_2} - a_3 S_{(\theta_2+\theta_3)} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The final matrix T_3^0 represents the position and orientation of the leg's foot relative to the base frame. The first three entries (O_3^0) of the last column of T_3^0 are the x, y and z components of the O_3 (foot location) relative to the base frame; that is,

$$x = C_{\theta_1}(a_1 + a_2 C_{\theta_2} + a_3 C_{(\theta_2+\theta_3)}) \tag{3.4}$$

$$y = S_{\theta_1}(a_1 + a_2 C_{\theta_2} + a_3 C_{(\theta_2 + \theta_3)}) \quad (3.5)$$

$$z = -a_2 S_{\theta_2} - a_3 S_{(\theta_2 + \theta_3)} \quad (3.6)$$

These coordinates represent the right leg's (RF, RM or RR) foot in the base frame. The 3×3 rotation matrix R_3^0 presents the orientation of the frame $o_3\, x_3\, y_3\, z_3$ relative to the $o_0\, x_0\, y_o\, z_0$.

The last two links (femur and tibia) are moving in one plane ($x_1 - y_1$ plane) and all of the rotational axes (z_1, z_2 and z_3) are perpendicular to this frame. When the robot moves, the robot raises its legs from the ground by changing the angular values of θ_2 and θ_3. Then, these legs are moved forward or backward by changing the angle θ_1 for each leg individually; as such, the $x_1 - y_1$ alternates between two values of θ_1. If the robot wants to move forward on a flat floor with constant speed, each of the θ_1, θ_2 and θ_3 moves alternately between two constant angular values.

The previous approach described how to calculate the feet's position and orientation in terms of the joint variables. In the next approach, the joint variables will be determined in terms of the feet's position and orientation. This will be done by using the inverse kinematics with geometric approach to find the θ_1, θ_2 and θ_3 values. First θ_1 is determined by the formula

$$\theta_1 = Atan2(x, y) \quad (3.7)$$

To simplify the problem, θ_2 and θ_3 are determined when $\theta_1 = 0$. In this case, the $x_1 - y_1$ plane falls on the $x_0 - z_0$ plane, as shown in Figure 3.10, which represents the geometry of the robot's leg. Therefore, y in Equation 3.5 will be zero and by following the procedure presented in [73], θ_2 and θ_3 are

$$\theta_2 = Atan\,2(z, x - a_1) + Asin\,2(b, c) \quad (3.8)$$

$$\theta_3 = \theta_2 + Acos2(a_3, x - a_2 cos\theta_2 - a_1) \quad (3.9)$$

where

$-90° \geq \theta_2 \leq 90°, and\ 0 \leq \theta_3 \leq 90°$, and

$$b = 2\sqrt{z^2 {a_2}^2 + (x - a_1)^2 {a_2}^2}$$

$$c = {a_3}^2 - z^2 - (x - a_1)^2 - {a_2}^2$$

Note that θ_2 is assigned in Figure 3.10 with a negative direction and this does not affect the calculations because θ_2 can take $\mp 90°$.

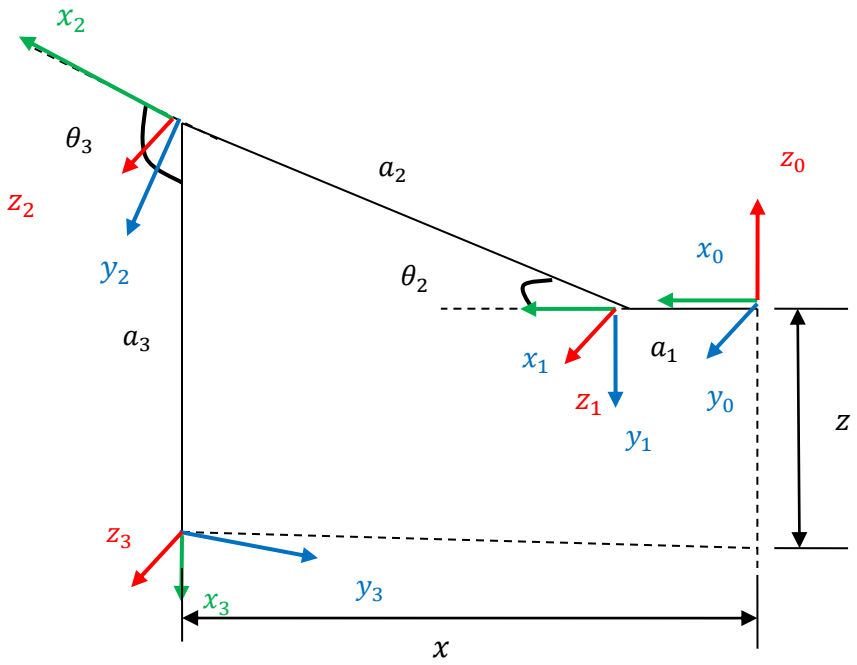


Figure 3.10. Geometry of the robot's leg

3.7 Velocity Kinematics

Once the forward kinematic equations are found, the velocity relationships that relate the linear and angular velocities of the leg's foot to the joint velocities can be determined by using the Jacobian (*J*). The Jacobian is the most important matrix in the analysis and in controlling the robot's motion because it is crucial to: the execution a smooth trajectory, determining the dynamic equations of motion and finding the relationships among the joints' forces and torques [34]. The velocity relationships can be determined using the formula

$$\xi = J\dot{q} \tag{3.10}$$

in which ξ is a vector of the linear and angular velocities of the leg's foot $\xi^T = [v \quad w]$, $\dot{q}$ is a vector of the joints' angular velocities $\dot{q}^T = [\dot{\theta}_1 \quad \dot{\theta}_2 \quad \dot{\theta}_3]$, and J is the Jacobian which is obtained from

$$\mathrm{J} = \begin{bmatrix} z_{i-1}\, X\, (O_3 - O_{i-1}) \\ z_{i-1} \end{bmatrix} \tag{3.11}$$

where $i = 1, 2, 3$

The various quantities above are easily seen in the forward kinematics approach to be

$$O_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad O_1 = \begin{bmatrix} a_1 C_{\theta_1} \\ a_1 S_{\theta_1} \\ 0 \end{bmatrix}, \quad O_2 = \begin{bmatrix} C_{\theta_1}(a_1 + a_2 C_{\theta_2}) \\ S_{\theta_1}(a_1 + a_2 C_{\theta_2}) \\ -a_2 S_{\theta_2} \end{bmatrix}, \tag{3.12}$$

$$O_3 = \begin{bmatrix} C_{\theta_1}(a_1 + a_2 C_{\theta_2} + a_3 C_{(\theta_2 + \theta_3)}) \\ S_{\theta_1}(a_1 + a_2 C_{\theta_2} + a_3 C_{(\theta_2 + \theta_3)}) \\ -a_2 S_{\theta_2} - a_3 S_{(\theta_2 + \theta_3)} \end{bmatrix}$$

The Z_{i-1} of the DH frames are given by

$$Z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \qquad Z_1 = Z_2 = \begin{bmatrix} -S_{\theta_1} \\ C_{\theta_1} \\ 0 \end{bmatrix} \tag{3.13}$$

Substituting the quantities of Equations (3.12) and (3.13) into Equation (3.11), and performing the required calculations yields

$$J_{(right)} = \begin{bmatrix} -S_{\theta_1}(a_1 + a_2 C_{\theta_2} + a_3 C_{(\theta_2 + \theta_3)}) & -a_2 C_{\theta_1} S_{\theta_2} - a_3 C_{\theta_1} S_{(\theta_2 + \theta_3)} & -a_3 C_{\theta_1} S_{(\theta_2 + \theta_3)} \\ C_{\theta_1}(a_1 + a_2 C_{\theta_2} + a_3 C_{(\theta_2 + \theta_3)}) & -a_2 S_{\theta_1} S_{\theta_2} - a_3 S_{\theta_1} S_{(\theta_2 + \theta_3)} & -a_3 S_{\theta_1} S_{(\theta_2 + \theta_3)} \\ 0 & -a_2 C_{\theta_2} - a_3 C_{(\theta_2 + \theta_3)} & -a_3 C_{(\theta_2 + \theta_3)} \\ 0 & -S_{\theta_1} & -S_{\theta_1} \\ 0 & C_{\theta_1} & C_{\theta_1} \\ 1 & 0 & 0 \end{bmatrix} \tag{3.14}$$

The first three rows of Equation (3.14) represent the linear velocity of the o_3 (foot) relative to the base frame (the robot body). The last three rows are the angular velocity of $o_3\ x_3\ y_3\ z_3$ frame (leg's foot frame).

3.8 Torque distribution

The total weight of the robot's constituent parts creates forces and moments at the feet of the robot's legs, which produces torques at the legs' joints. It is important to compute the joint torques in order to select suitable motors to support the robot motion.

Generally, the relationships between the joint static torques and the forces and moments that are produced at the robot's feet can be determined by using the formula

$$\tau = J^T(q)F \tag{3.15}$$

in which $J^T(q)$ is the J transposition of the leg, F is the resulting vector of the forces and moments at the end of lower leg (foot) and τ is the corresponding vector of the joint torques.

Substituting Equation (3.14) into Equation (3.15) and performing the required calculations yields the joint static torques of the legs on the right side of the robot's body, which are given as

$$\begin{bmatrix}\tau_1\\ \tau_2\\ \tau_3\end{bmatrix} = \begin{bmatrix} -S_{\theta_1}(a_1 + a_2C_{\theta_2} + a_3C_{(\theta_2+\theta_3)}) & C_{\theta_1}(a_1 + a_2C_{\theta_2} + a_3C_{(\theta_2+\theta_3)}) & 0 & 0 & 0 & 1\\ -a_2C_{\theta_1}S_{\theta_2} - a_3C_{\theta_1}S_{(\theta_2+\theta_3)} & -a_2S_{\theta_1}S_{\theta_2} - a_3S_{\theta_1}S_{(\theta_2+\theta_3)} & -a_2C_{\theta_2} - a_3C_{(\theta_2+\theta_3)} & -S_{\theta_1} & C_{\theta_1} & 0\\ -a_3C_{\theta_1}S_{(\theta_2+\theta_3)} & -a_3S_{\theta_1}S_{(\theta_2+\theta_3)} & -a_3C_{(\theta_2+\theta_3)} & -S_{\theta_1} & C_{\theta_1} & 0 \end{bmatrix}\begin{bmatrix}F_x\\ F_y\\ F_z\\ M_x\\ M_y\\ M_z\end{bmatrix} \tag{3.16}$$

The problem of torque distribution in a dynamic state is more complicated because the angular and translational accelerations and velocities of the joints will affect the calculations, for more information see [74].

3.9 Movement (Gait) Control

Figure 3.11 illustrates the movement control system of the hexapod. The robot starts to read and process the sensor data using the main controller (Roboard) to obtain the environmental information. This information is then sent to the motion planning and navigation algorithm. Motion planning divides the total displacement that the robot wants to achieve into smaller sections. As mentioned in section (3.5), the maximum distance of one walking step is 7 cm and thus, each section can be equal to or less than this displacement (in this study, the maximum step size is limited to 5 cm to reduce the probability of legs colliding with each other). The navigation algorithm uses the environmental information within the robot's motion and determines the robot's state and location in the environment. Accordingly, the robot calculates the desired total displacement or steering angle needed. The inverse kinematic is used to calculate the desired joints' angles to manage the movement of the robot. The microcontroller

translates the desired joints' angles into the PWM signals that control the operation of the servo motors.

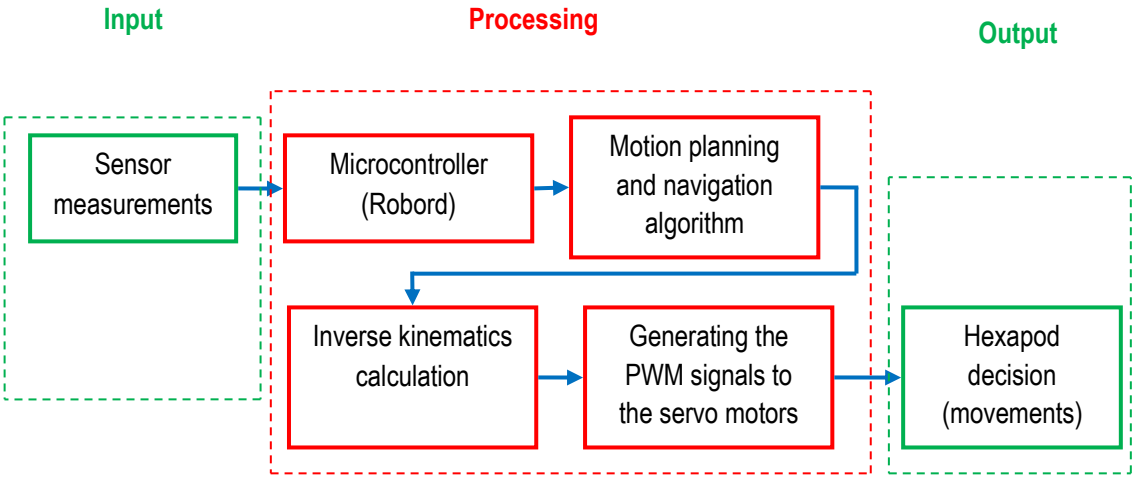


Figure 3.11. The movement control system in the hexapod

3.10 Summary

In this chapter, the hexapod mobile robot is described, analysed and discussed. First, the mechanical and electronic systems are described. Then, the walking control system is explained; the tripod gait is used for walking forward and the wave gait is employed to rotate the robot about its central axis. Next, the forward and reverse kinematics modelling is derived, followed by calculations of the Jacobian. The calculation of the torque distribution on the joints of the legs is then described. Finally, the movement control system is explained, in which, the robot explores the environment by using its sensors. The sensors' signals are processed by the robot's controller, which accordingly generates the PWM signals to the servo motors. The robot's movement using the tripod gait is faster but its coordination is more complicated. The movement by wave gait is slower but more stable.

Chapter 4 The Wheeled Robot

4.1 Introduction

As mentioned in Chapter 3, it would be worth studying two types of mobile robots having two philosophies of locomotion configurations to identify limitations on the navigation system due to terrain. The methodology will also be tested within different software and electronic modules. Therefore, the decision was taken to design and implement a new wheeled robot to test the proposed methodology. The robot must have an on-board processor that has more computational capacity than the one in the hexapod, because of after implementation of all the algorithms within the hexapod's controller, two problems have come to light. The first is that the board does not support the image processing techniques that need high computation capacity; therefore the robot stopped moving when any failure happened in processing the image. The second is that the board was extremely slow with the computation process, and this makes the robot slower than the expected speed. In this chapter, the design and development of an autonomous, wheeled mobile robot's platform will be explained. This robot will be used to search for, find and relocate a small, cylindrical object in an indoor environment. The main structure of the robot consists of mechanical and electronic systems.

4.2 Mechanical Design

The mechanical structure consists of the robot body (chassis), wheel configuration, motor drives and gripper.

4.2.1 Chassis

The robot's chassis was constructed from aluminium with a thickness of 3 mm. It is 300 mm long and 200 mm wide with a frame that includes two floors, each one consisting of four bars (4 bars: two 20 cm long and two 30cm long) and forming a rectangular shape. These floors are separated by 4 holding bars, each 15 cm long (see Figure 4.1). The bottom floor of the frame is used to attach the DC motors, microcontroller, motor drives, range sensors that are placed at the front, and the battery. The upper floor is used

to attach the motherboard and hold other range sensors that are placed at the robot's sides.

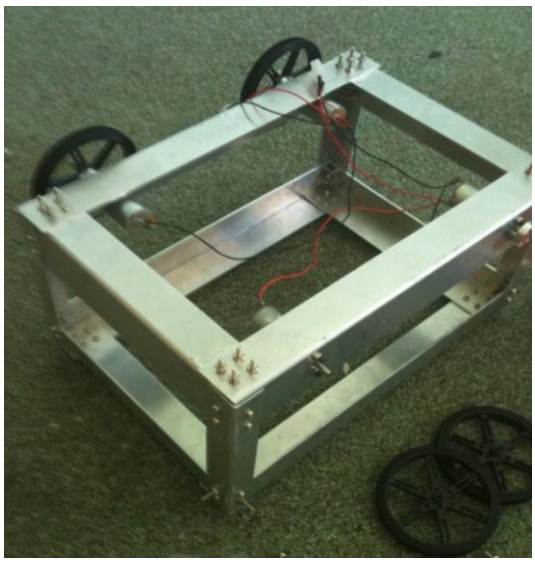


*Figure 4.1***.** The robot chassis

4.2.2 Wheel Configuration

The decision was made to use a four-wheeled drive principle with a zero-turn radius mechanism. A robot using this configuration reorients itself by rotating the pair of wheels mounted on one side in a particular direction, while rotating the pair on the other side in the opposite direction. If the robot is required to drive in a straight line, it will rotate all the wheels at the same speed and in the same direction. This mechanism has some benefits with respect to other wheel configurations (see Appendix). The main benefits are:

- Robot stability: the four-wheeled configuration ensures stability;
- Robot movement: the robot can move to a desired site and turn in place to attain a particular orientation;
- Robot power: four wheels contribute to produce the robot's motion and steering; this makes the robot more powerful;
- Robot design: this kind of robot is easy to design and implement; and

- Robot steering radius: the robot is capable of executing a zero-point turn radius.

Although using four wheels with four motors has many advantages, it also has some disadvantages, the main ones being:

- Robot controllability: the straight line motion control has proved to be difficult, since all the motors have to rotate at the same speed;
- Robot precision: the alignment of the four wheels must be precisely set to guarantee straight line motion. Thus, all wheels have to contact the ground at the same time to ensure straight line motion; and
- Robot drive: this method of robot motion needs strong motors on both sides to perform the zero-point turn radius.

4.2.3 Motor Drives

The previous section reports that four motors were first needed on the mechanical platform. The next step was choosing the motors. Three types of motor can be considered for driving the mobile robot: DC motors, stepper motors and servo motors (see Appendix). The decision was taken to use DC motors for the driving system. Initially, some specifications of DC motors were examined and determined. These motors must be able to drive the robot and produce motion. To determine the power and torque needed by each motor, the maximum mass of the robot (m) was assumed to equal 12 kg and the robot will be used on a flat floor. Therefore, the force (F) needed to move the robot is

$$F = F_f = \mu * m * g \tag{4.1}$$

in which F_f and μ are the friction force and the estimated friction coefficient between the robot's wheels and the ground, respectively, while 'g' is gravitational acceleration. Since the four-wheeled configuration is used, the robot's weight is applied on four wheels. In this case, the force supplied by each motor is

$$F_{motor} = (\mu * m * g)/4 \tag{4.2}$$

If the maximum friction coefficient μ, when the robot starts moving, and g are assumed to equal 0.8 and 10 $\mathrm{m/s^2}$ respectively, this will produce a force

$$F_{motor} = \frac{0.8*12*10}{4} = 24\ N \tag{4.3}$$

The maximum linear velocity of the robot is influenced by both the speed of the motors (measured in revolutions per minute (*RPM*)) and the diameter of the wheels (*d*). This is determined by the formula

$$v_{max} = \pi * d * RPM \tag{4.4}$$

Since the mobile robot will be used to search the indoor environment, which might have maximum dimensions of 10 x 10m, then the robot's maximum speed (v_{max}) that is appropriate for the navigation and search tasks is assumed to be 10 m/minute. If the wheel diameter (*d*) is chosen to be 8 cm, then the motor's speed is

$$10 = \pi * 0.08 * RPM \tag{4.5}$$

$$RPM \approx 40$$

In this case, the motor's maximum output power will be

$$P_{max} = F_{motor} * v_{max} \tag{4.6}$$

$$P_{max} = 24\ N * \left(\frac{10\ m}{minute} * \frac{1}{60}\frac{minute}{s}\right) = 4\ watt$$

The torque needed (*T*) is then calculated from

$$T = P_{max}\ /(2\pi * RPM * \tfrac{1}{60}) \tag{4.7}$$

$$T = 4\ /(2 * \pi * 40 * \frac{1}{60}) \approx 0.96\ N \cdot m$$

The various calculations being completed, the choice of suitable motors can now be started. Four geared high-torque (1.17 $\mathrm{N \cdot m}$) DC motors that operate at 12 V were chosen (see Appendix). These motors have a maximum current draw of 1.5 Amps and rotate at 36 RPM. This is less than 40 RPM and reduces the robot's maximum speed to 9 m/minute, but it does not affect the calculations since the motor has a high torque that is greater than the maximum torque needed. The motors are attached to the robot's chassis and their shafts are directly coupled to wheels with 80 mm diameter. Each pair

of motors, mounted on one side, turns in the same direction and at an identical speed. This is achieved by connecting each pair of wheels with the same line, as shown in Figure 4.2.

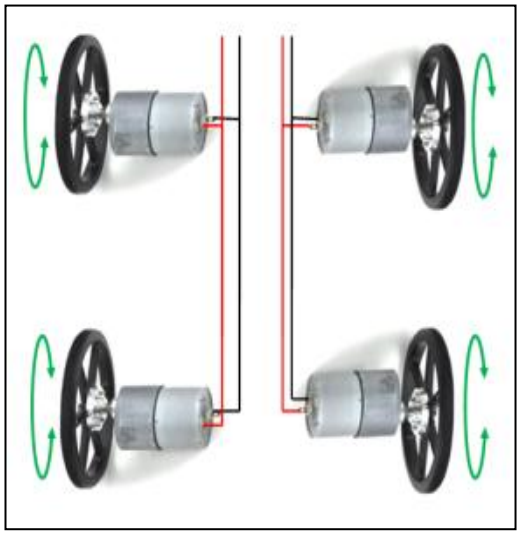


Figure 4.2. The wheeled-drive configuration

4.2.3.1 Powering Motors

The speed and direction of the DC motors are controlled by the microcontroller board; however, the motors need more energy than can be supported by this board. Consequently, a separate motor drive amplifier that can support the required power and can also be controlled by the microcontroller must be used. The motors' control board [75], having 4 channels, each of which can control and provide 4 Amps (peak load) per motor, is chosen. This board provides pulse width modulation (PWM) pins that can individually control the speed and direction of four motors. Note that the PWM signals are generated using the microcontroller software and then sent to the pins of the motors' control board. As mentioned above, one of the requirements is that the robot is capable of executing a zero-turn radius. Accordingly, each pair of motors is connected to one channel and controlled together at the same time. For instance, if the robot wants to move in a straight line, it will run both pairs of wheels in the same direction and at identical speeds. Conversely, if it wants to change orientation, it will rotate both pairs of motors in opposite directions. The benefit of using this method is that the robot can change its direction by spinning about its central axis.

4.2.4 The Gripper

Typically, industrial robots (manipulators) need devices, called end effectors, at the end of their arms in order to interact with the environment and to perform their tasks. These devices might be either tools, used to perform tasks like welding and drilling, or grippers, used to reposition objects from one location to another. In the case of mobile robotics, the robots might need these kinds of devices to do the assigned tasks. In this study, the robot is used to relocate the targets; therefore, it needs a gripper to execute this process.

Two factors must be considered when the gripper is designed: the grasping force that is applied to the object, and the jaw torque [76]. The former is influenced by two aspects: the style of the gripper's jaws and the object's weight. Jaws are generally designed in two styles: friction and encompassing [76, 77]. In the friction-jaw style, the holding of the object is totally done by the grasping force. However, the encompassing-jaw gripper cradles the object because its jaws are designed in the same shape as the object. Consequently, it needs less holding force. The object's weight that the gripper experiences from both gravity and acceleration is a critical factor in determining the required gripping force. Lastly, the jaw's total torque is produced by the grasping force, together with the acceleration and weight of the object.

In this work, a small gripper was designed and implemented to grasp the target object, assumed to have a maximum mass (m) equal to 100 grams. The gripper is a friction-jaw style; however, it can be considered an encompassing-jaw gripper if it is used to grasp a rectangular shaped object. The gravitational acceleration (g) is assumed to equal $10\ \mathrm{m/s^2}$ and the total jaw length is 9 cm; the distance from the gripping force to the rotational centre is 5 cm. The weight that comes from acceleration is assumed to be equal to the gravitational weight. Then, the formulas that are in [76, 77] were used as follows

$$\textit{Grip Force Required (F)} = \textit{Object's mass} * (g + \textit{Part Gs}) * \textit{Jaw style factor} \quad (4.8)$$

where the *Part Gs* represents the robot's acceleration and it is assumed to equal the gravitational acceleration (g), and the *Jaw style factor* equals one in case of the encompassing-jaw gripper. As such, the force required will be

$$F = m * g * 2 * 1 = 0.100 * 10 * 2 * 1 = 2N$$

This force generates the jaw torque that can be determined by

$$\textit{Jaw Torque (J)} = \textit{Jaw Length} * \textit{Grip Force} \quad (4.9)$$

$$J = 5 * 2 = 10N \cdot cm$$

The gripper will also experience the torque from the object that is essentially given by

$$\textit{Object Torque (P)} = \textit{Jaw Length} * \textit{Object's mass} * \textit{Part Gs} \quad (4.10)$$

$$P = 5 * 0.1 * 10 = 5N \cdot cm$$

As mentioned above, the jaw's total torque is made by the grip force, together with the acceleration and weight of the object. As such, the total torque is

$$\textit{Total Torque (T)} = \textit{Jaw Torque (J)} + \textit{Object Torque (P)} \quad (4.11)$$

$$T = 10 + 5 = 15N \cdot cm$$

Therefore, the specifications of the gripper have to be 2 N of the gripping force and 15 N · cm of the torque. The jaws of the gripper were designed to be driven by two servo motors. The gripper is attached to the 10 cm long arm and can be moved up and down by another servo motor which is attached to the robot's body (see Figure 4.3). The servo motors are controlled using the PWM signals that are generated by the microcontroller software.

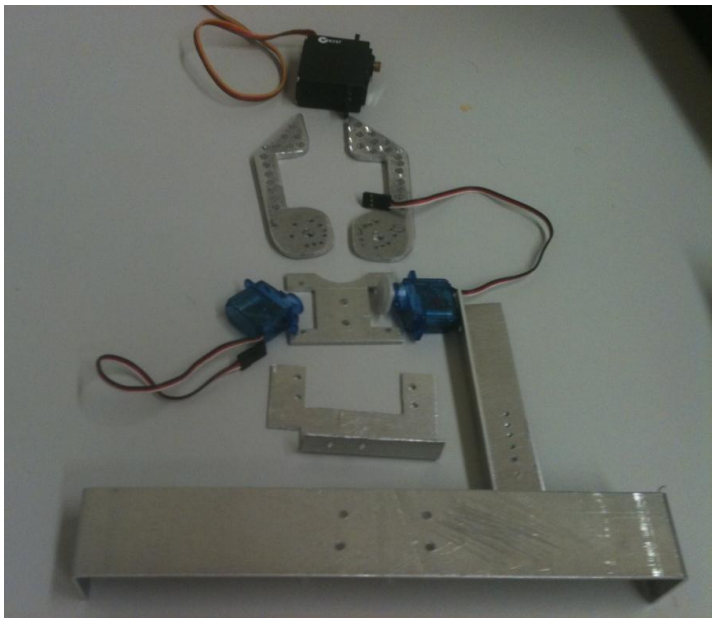


Figure 4.3. The gripper parts

4.3 Electronic Circuit Design

Before describing the electronic system, it is worth mentioning the main requirement for designing a new robot. An autonomous mobile robot is needed to search for, find and relocate a target (object) in an indoor environment. In this scenario, it needs to employ sensors for navigating and detecting the target, and for controlling the robot's motion. An electronic module system will play the main role in the performance of these tasks. Typically, detecting the target requires use of a vision system that needs high-level software (high computational processing). Conversely, the signals of other sensors are processed by low-level software (low-level processing).

The high-level software needs to be run by a processor with a high computational capacity. The processor should also operate in standby mode since it is mounted on the robot. The decision then taken is to use a PC motherboard. However, this board does not have the sensor drivers and digital/analogue I/O circuitry necessary for handling the sensors and driving the motors. To solve this problem, there are two options. The first employs adapters that can handle the sensors and the motors to the PC motherboard. This will create two more issues. First, the sensors need to be run by software programs that can drive and process the sensory data; however, the motherboard does not have these. Second, even if the required software is installed, the CPU needs extra time and effort to process the sensors' signals. The previous two problems would be solved by using the second method, which employs a microcontroller to handle and manage the low-level software. As such, the microcontroller could serve as the interface unit between the main electronic components in the robot: PC motherboard, sensors and motor driver. The decision was finally made to combine these two modules, the PC motherboard and the microcontroller, to implement the main electronic system.

4.3.1 The PC Motherboard

An IBM Lenovo motherboard [78], which has Intel Pentium 4 CPU 3.00 GHz and 1.00GB of RAM, is used. It has six USB ports and both parallel and serial ports. Some additions are made to this board to operate on the robot.

4.3.1.1 Software

The software programs play the main role in implementing the robot's electronic system. The required software includes a choice of an operating system, the programming languages, and whatever tools and libraries are necessary.

Operating system

Choosing a suitable operating system (OS) is important because it takes care of several aspects in the computer system such as handling, organizing and control of the hardware and its programming languages [79]. Various OS exist such as Windows, Mac OS X, and Linux. In robotics, Linux is extensively used because it is an open source, has good hardware and software support, is well documented, and has high performance and stability. Consequently, Linux OS (Ubuntu 10.04 LTS) [80] was chosen and then installed in the USB flash memory (8 GB). This memory is used as a hard disc for the motherboard because it is small, light and has low power consumption. Note that it is important to create partitions in the USB flash memory to save any changes or install programs, because without this partition any changes will be omitted.

Program languages

The main software tools and libraries that have been downloaded and installed are:

- A programming IDE (Integrated Development Environment) "codeblocks" to edit, compile and run the image processing algorithms and to control and program the motherboard ports;
- Intel Open Computer Vision (OpenCV [81]) and OpenSURF [82] libraries in order to do image processing; and
- The microcontroller (Arduino) open-source software "sketches" to edit, compile and download the program, which controls the robot's motion based on the sensory information and also controls the gripper's action, into the microcontroller.

4.3.2 The Microcontroller

The Arduino Board [83] was chosen because: it provides all the needs for sensors and motors; it has the open source software for Windows and Linux; and it is easy to

configure I/O for other devices. The board has a USB port that is used for both downloading the control program for running on the board, and powering the board. The board's features include an ATmega328 microprocessor, 32k flash memory and 16 MHz clock speed. It has three groups of female-pin headers that allow connection and control of other devices. The groups are: power, analogue input, and digital input/output. The analogue input group has six pins which can also serve as general purpose digital I/O. The digital input/output group has 14 pins including six headers that generate pulse width modulation (PWM) signals; these are useful for controlling the speed of the motors. Through I/O pins, the board also supports the basic communications standards such as I2C and TTL serials. Overall, the platform can support many types of sensors which make the robot more scalable and suitable for more applications.

4.3.3 Communication Process

The PC motherboard and the microcontroller need to communicate and control each other. Normally, in the communication process the PC behaves as a master (host) and each of other peripherals are slaves (devices). The PC motherboard has the ports that can be used mainly to control and communicate with external devices (see Appendix). The PC parallel port (sometimes called a printer port), which has 25 pins, is employed for the communication process. The port's pins are classified into four categories: data registers (pins 2-9); control registers (pins 1, 14, 16 and 17); status registers (pins 10-13 and 15); and grounds (pins 18-25). The data registers can be used to store a byte of data, which is sent as output to the data register pins. The control registers are mainly used to send control data to the printer port, while the status registers can read the states of the status pins. As such, the motherboard can be programmed to send 8 bits of information to the data pins, and to receive 5 bits of data from the status pins at once. The parallel port pins can also be used individually to send and receive data. Generally, this port is easier to program and faster than the serial ports [84].

The microcontroller board has a USB connection that works as a serial connection. Typically, this port is used to download the control program and to power the microcontroller. This port can be used as a communication interface between the microcontroller and the motherboard. In this case, a communication program must be

written for each board to control each other. As mentioned above, the serial port is more difficult to program and control than the parallel port. It also requires additional CPU time to process and assess the communication messages between the two boards. Consequently, the decision was taken to use some of the parallel port's pins to send 2 bits of data to control the microcontroller, and to receive 2 bits of information from the microcontroller to control the motherboard (this process is explained in Chapter 7).

4.3.4 Sensors and Movement Control

The robot must be able to move safely and to recognise objects in the environment. Therefore, ultrasonic range sensors and a camera were used within the navigation system to achieve the task. Three types of SONAR devices were used, namely, Devantech SRF02, SRF08 and SRF10 ultrasonic range finder [85]. These sensors communicated with the microcontroller board (Arduino) by analogue pins 4 and 5 in alternate I2C function mode. In this study, the robot was equipped with a Logitech camera model C200 [86] as a vision sensor. It takes images with a resolution of up to 640 by 480 pixels with the maximum rate of image capture of 30 frames per second. The camera was connected to the motherboard by an USB port. The camera was attached on a servo motor to rotate it 180° to its left or right side, and placed on the front of the robot (the sensor configuration is described in Chapter 5).

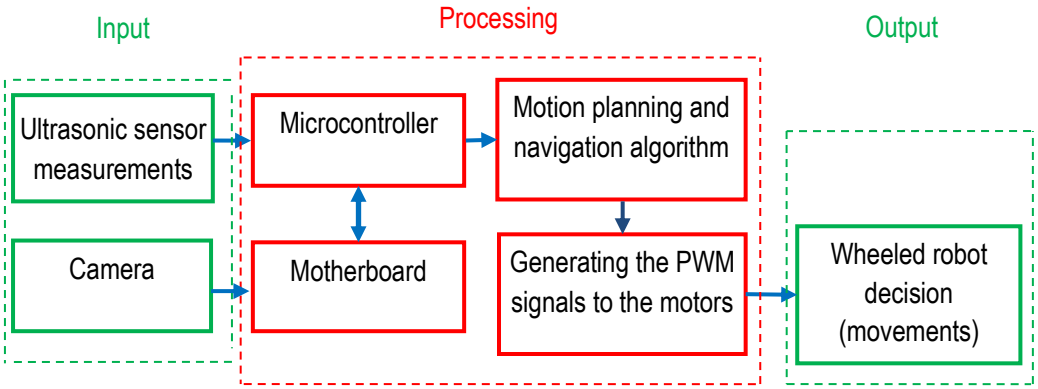


Figure 4.4. The movement control system in the wheeled robot.

Figure 4.4 illustrates the movement control system of the wheeled robot. As mentioned before, the robot was made to combine two modules to implement the main electronic system. The PC motherboard was to execute the image processing calculations while

the microcontroller was to process the sensory signals and performing the movement decisions. As such, the robot could capture images and process them while it navigated. Then, the image processing results were sent to the microcontroller that accordingly decided on the robot's motion. The robot could update the environmental information in each of its movement cycles without stopping its motion. Therefore, the robot's reaction for any change in the environment was much faster than the hexapod.

4.4 Power System

The PC motherboard normally needs an AC-DC power supply providing +3.3, +5 and ±12 V. When this board is used for a robotics project, it must be run by a DC battery instead of the power supply. As such, a small DC-DC power supply, called Pico-PCU, was used with a 12 V DC battery to provide the required voltages. The DC motors and servo motors require a 12 V and 6 V DC power supply. In this case, there were three options for using batteries as explained below:

- two DC batteries (12 V and 6 V) are used; however, this increases the robot's weight;
- a DC to DC converter is used; or
- DC battery cells are used to power the different parts.

In this project, the third option was at first used to feed the motors and the motherboard. However, it was found that the motherboard was reset when the robot started to move. This was because the motors draw high current when they start motion. Therefore, a separate 12 V DC battery was used to feed the motherboard, and the 10 × 1.2 volt DC cells were employed to drive both the robot and its gripper.

4.5 Circuit Schematics

Figure 4.5 shows all the components and data lines in the electronic circuitry of the robot.

- The microcontroller was powered from the motherboard via a USB cable.
- The microcontroller fed the logic circuit of the motor controller via two wires (Vcc and Ground).

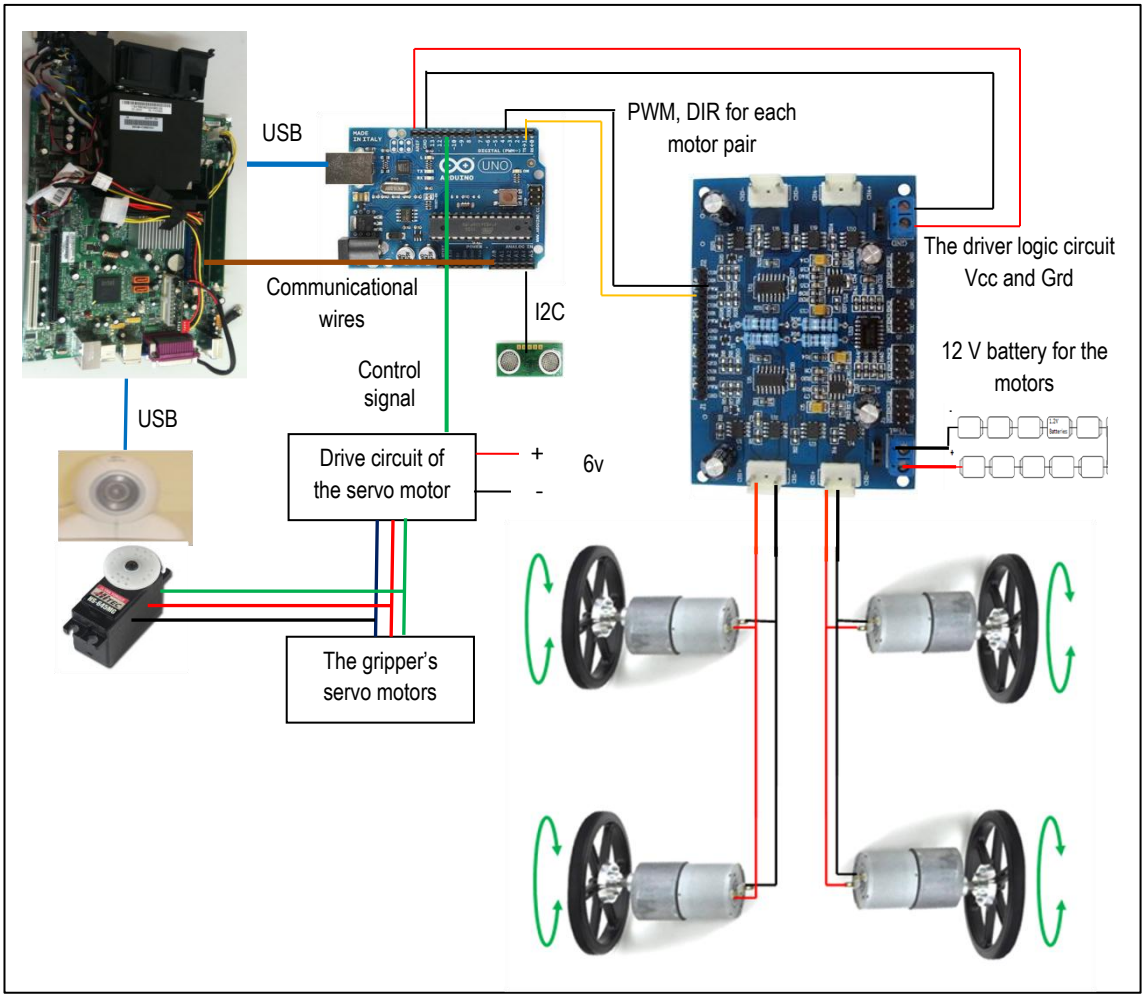


Figure 4.5. The circuit configuration

- The range sensors were attached to the microcontroller via the I2C communication (pins 4 and 5).
- Each pair of motors, attached to one side of the robot, was connected to one channel of the motors' control board. This board was controlled by two of the microcontroller's I/O-digital pins, which represent PWM (to control the speed of the motors) and DIR (to control the direction of the motors). One of the motor pairs was connected to pins 3 and 4 while the other pair was connected to pins 5 and 6.
- The microcontroller controlled the PC motherboard via two digital pins (7 and 8), which were connected to two status pins of the parallel port.

- The PC motherboard controlled the microcontroller by two data pins of the parallel port, which were connected to two analogue pins A0 and A1 of the microcontroller.
- The camera was attached to the motherboard via a USB.
- The robot's power was provided by 10 × 1.2 volt cells connected in series. The DC motor controller used all the cells (12 volts), whilst the servos were joined to just 5 × 1.2 volt cells
- The servo motors (three for the gripper and one to "pan" the camera) were controlled by the digital pins of the microcontroller (pins 9, 10, 11, 12).

4.6 Summary

The decision was made to test the proposed methodology using two types of mobile robots having two philosophies of locomotion configurations in order to identify limitations on the navigation system. In Chapter 3, the hexapod mobile robot is described, analysed and discussed. This chapter proposes a methodology for designing and implementing a wheeled robot. The designed robot has a four-wheeled drive principle with a zero-turn radius mechanism. The robot's electronic system combines two modules, the PC motherboard and the microcontroller. The former is used to execute image processing while the latter is to process other sensors' signals. The communication between the two boards is done by using the motherboard's parallel port. The robot also has a gripper to transport the objects.

Chapter 5 Exploration Path and Motion Control

5.1 Introduction

A mobile robot generally moves in an environment that contains a starting point, the target position and a number of arbitrarily shaped obstacles, each of which has a finite area. The robot's objective is to find a continuous path from the start point to the target position without collision. This process is organised by means of the controller that includes the navigation system. The aim of the navigation system is to enable the robot to plan its motion. The motion planning for the mobile robot involves three considerations: gathering the environmental information and then transferring it to a mapping module in real-time; generating a path that the robot can navigate without collisions; and controlling the robot throughout its motion [16, 87]. In the case of a mobile search robot, the motion planning must enable the robot to navigate in a planned path and to continue searching and discovering its environment. The target position is unknown, which means that the path needs to be carefully planned in order to cover the entire search area.

The navigation system is required to model the environment in order to enable the mobile robot to execute its tasks. Modelling includes the process of mapping the environment based on the information obtained from the mobile robot's sensors, in order to determine the position of various entities, such as landmarks or objects. Without this mapping the mobile robot cannot navigate and find objects in the environment or plan its path to a target location [88]. In most mobile search robot applications, the environment and the target's location are unknown and therefore, the environment modelling has to be done taking into consideration the similarities among different environments. In indoor environments, there are many aspects that can be used as a reference for the robot's motion, such as walls and doors. The walls are particularly important when designing a navigation system that enables the mobile robot to work autonomously in different indoor environments. Pieces of furniture on the floor of the environment may obstruct access to the target object and need to be considered when designing the navigation system.

This chapter describes the exploration path and motion control. It starts by explaining how the Bug algorithms are adapted and implemented to guide the robot when exploring its environment. Then it explains the process of control of the robot motion, which is dependent on its sensory information.

5.2 Motivation for Bug Algorithms

As mentioned in Chapter 2, the navigation algorithms are classified into global and local planning [25]. The global algorithms plan the robot's path from the start to the goal, by searching a graph that represents a map of the global environment. The environmental graph is constructed either off-line or on-line. The former assumes that the environment is known completely, whereas the latter builds the graph based on local sensory information. The global planning methods have three drawbacks: expensive in computation, complexity of construction and difficulty of obtaining an accurate graphical model. Conversely, local planning algorithms use sensor information directly in the commands that control the robot's motion in every control cycle, without constructing a global map [25]; therefore, they are easy to construct and optimal for real-time applications.

The Bug algorithms [2] are intended to steer the robot from the starting point to the target position without an environmental map. As such, they are identical to local planning techniques, as they need only local environmental information. However, the robot needs to learn a little of the global information, such as the number of path points between the start and the final locations. If the robot discovers that no such path exists, the algorithms will terminate the robot's motion and report that the target is unreachable.

5.2.1 Bug Family Close Review

The Bug family includes many algorithms that involve small variations on those initially developed by [89, 90]. The family generally assumes that the robot is a point object (analogous to a bug), having perfect localisation ability and perfect sensors [2]. The Bug family also assumes that the robot knows the distance and direction between the starting point and the target location, in addition to a minimal number of its path's

points. All the Bug algorithms employ at least two procedures to operate: navigating to the target; and following the obstacles' boundaries. Typically, these algorithms can be adapted for any robot that has range or tactile sensors and a localisation technique such as landmark recognition [91]. The simplest Bug algorithm (Bug 1) could be described in Figure 5.1.

Figure 5.2 represents two sample environments. The green tile indicates the start S, while the red identifies the target T. The dashed black line represents the desired path between the start and the goal, which is defined to the robot, while the green arrows denote the actual path of the robot's motion. The aims of all Bug family algorithms are explained as follows. The robot starts moving directly towards the target when it can; if it hits an obstacle, it goes forward along the intervening obstacle boundaries until that obstacle can be overcome and the robot continues directly towards the goal, as shown in Figure 5.2A. However, if no path exists to the target, the algorithms are able to identify this state and they terminate, reporting that the goal is unreachable, as shown in Figure 5.2B.

1. Drive towards the target T. Go to (2).
2. Is the target reached?
    a) Yes. Stop.
    b) No. Go to (3).
3. Is the obstacle encountered?
    a) Yes. Define the hit point (Hi). Go to (4).
    b) No. Go to (1).
4. Go around the obstacle. Continue updating and saving (Dm) the closest distance to the target, until:
    a) The target is reached. Stop.
    b) (Hi) is encountered again. Check if the target is surrounded by this obstacle (the line segment [Dm, T] is obstructed by the obstacle):
        i. Yes. The target is unreachable. Stop.
        ii. No. Return to (Dm). Go to (1).

Figure 5.1. The simplest Bug algorithm (Bug 1)

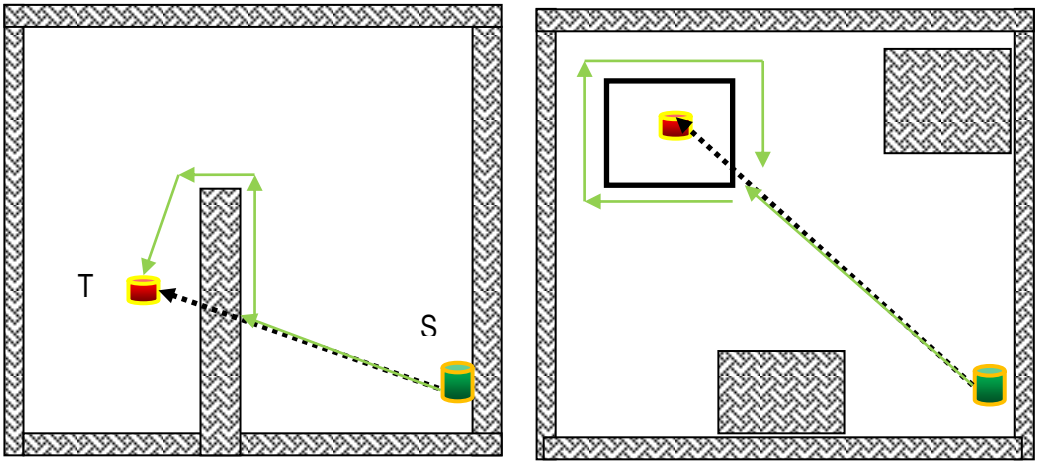


*Figure 5.2.*The Bug Algorithm states

5.3 The Bug-Like Algorithm

Ng and Bräunl [2] claimed that Bug algorithms are required to follow a wall to control the robot motion. This means that when the robot moves towards the target and there is an obstacle in the path, the robot will navigate around the obstacle by keeping it completely to one of its sides. Zhu, *et al.* [25] criticised previous works, which are mainly theoretical and concerned with optimising the path length and designing new situations for leaving obstacles. The authors claimed that earlier works ignored the practical implementation issues. The Bug algorithms also assumed that the robot moves close to an obstacle but no attention is paid to collision avoidance. The robot must actually maintain a safe distance from the wall or the obstacle that the robot is following. This is done by using a method based on ultrasonic range sensors, which are employed to generate the control commands to steer the robot. This will be explained in the following sections.

In this study, the spirit of Bug algorithms is employed for the Bug-like technique that will be used to implement the exploration path for the search robot. As mentioned before these algorithms use two procedures to operate: navigating to the target; and following the obstacles' boundaries. The robot with Bug algorithms also needs to know the exact distance and direction of the path between the starting point and target position, in addition to some points on that path. However, the robot in this work

explores an environment that is unknown but is defined for the robot as an indoor world that is constrained by external walls. In this scenario, the target objects might reside with equal probability at any location in the environment and, therefore, the robot must navigate and search the whole area autonomously.

Figure 5.3 shows the two sample environments previously depicted in Figure 5.2. Each environment contains the starting point, the target and a number of obstacles. The target position is unknown. The aim of the Bug-like algorithm is for the robot to start its motion by following the wall (for instance, the robot positions itself with the wall to its left). If it encounters an obstacle or another wall, it turns clockwise until it can drive forwards again. If the robot discovers that there is no wall to its left that can be followed, it moves in a circular pattern counter-clockwise until a wall is detected to the left or in the front of the robot. The navigation process then continues until the robot discovers the target, as shown in Figure 5.3A. The robot then terminates this algorithm and starts a new task that requires approaching, grasping and relocating the detected target object. However, if the target object is hiding and the robot cannot detect it, that is, no path exists to the target. In this case, the robot will continue to search for the target object until the starting point is encountered again, which means the goal is unreachable and the robot terminates the navigation (Figure 5.3B). The Bug-like algorithm is explained in Figure 5.4. Note that in this section, the exploration path is explained without consideration of the probable locations of the target in the environment (this case is described in Chapter 7).

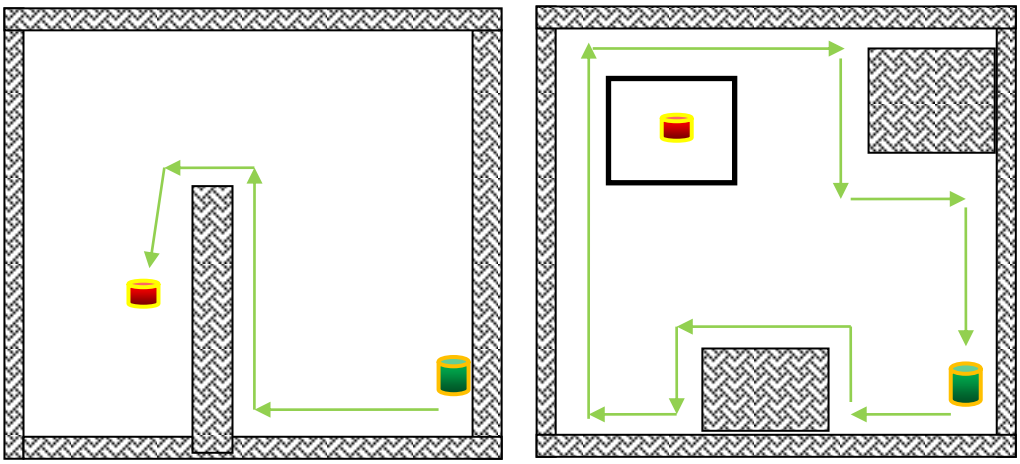


Figure 5.3. The Bug-like algorithm states

1. Is the robot's distance to the wall within the safe ranges?
    a) Yes. Go to (2).
    b) No. Correct the distance by moving further from or closer to the wall.
2. Drive forward from the start S by following the wall (in this case put the wall in the left side), and search for the target T. Test that:
    2.1 Is the target reached (detected)?
        a) Yes. Go to (3).
        b) No. Go to (2.2).
    2.2 Is the obstacle encountered?
        a) Yes. Check if the obstacle is the starting point S:
            i) Yes. The target is not found. Go to (3).
            ii) No. Rotate the robot clockwise until it can drive forwards again. Go to (1).
        b) No. Go to (2.3).
    2.3 Is there any wall on the right:
        a) Yes. Go to (1).
        b) No. Drive the robot in a circular pattern (counter-clockwise) until the wall is detected to the right or in the front of the robot. Go to (1).
3. Exit

Figure 5.4. The Bug-like algorithm

5.3.1 Localisation

The robot must know where it is starting and finishing its navigation and consequently, it has to recognise its location relative to the environment. In this study, the problem is that the robot starts moving from a starting point that can be recognised in the environment. The robot keeps itself a specific distance from an adjacent wall to either right or left. While the robot follows the wall, it keeps searching for the target. If the target is found, the robot will leave the wall and approach the target, grasp it and relocate it.

In the relocating process, whilst the robot is currently in the location where it grasped the object, it will search for the delivery location, which was its starting point. If that is found, the robot will approach it by tracking until the location is reached and the object will be placed there. However, if the delivery location cannot be seen from the current

location, the robot will move to the wall again and then follow it until the delivery location (starting point) is reached. If the target object is not found in the search process, the robot will continue following the wall until it encounters the starting point, where the robot terminates its motion.

This process assumes that the environment is surrounded by walls and its entrances are continuously closed. This does not work in practice because the environment has (entrances) doors that might be open while the robot navigates. In this case, when the robot follows the entrance's boundaries, it will leave the area that it must search. This problem is overcome by covering the environmental entrances with artificial landmarks so that the robot will consider them as walls, and it will navigate beside them (as will be explained in Chapter 7).

5.4 Sensor Configuration

Successful autonomous operations of mobile robots fundamentally require robust motion control systems. The control system selects its actions based on the present conditions of a robot's world. The surrounding environment is typically monitored through various sensors, including those for vision, range and force (applied on objects by the robot). The sensors convert the environmental information obtained into digital or electrical signals, which are used by the control system to control the physical actions of the robot. These mobility actions are normally motions executed with a mobile robot's real hardware, including wheels, legs and grippers. Figure 5.5 shows the control process for a robot with legs or wheels.

The sensor configuration consists of the types of sensors used and the distribution of these sensors on the robot's body. The former is mainly specified depending on the robot application and the amount of information required. Most types of sensors used for the acquisition of environmental information are vision sensors, ultrasonic range sensors, tactile sensors, and laser sensors. Vision sensors are usually used for applications needing a large amount of environmental information such as exploration and search and rescue applications. However, these sensors require processors with high computational capacity to process and extract information from the images. Range sensors are appealing in terms of cost and power consumption. Furthermore, they only

need low computational processing capacity to analyse their signals. Tactile sensors, which are transducers sensitive to touch, force, or pressure, are widely used to improve the stability (balance) of legged robots and to control gripping force. Laser sensors are widely used in navigation processes for mapping the environment and localisation of the robot because their readings are accurate. The number of sensors used and their distribution on the robot's body depends mainly on the robot's size and application. Typically, the accuracy of the robot's measurement system will dramatically rise if the robot is equipped with the high number of sensors. However, this increases the robot's price and leads to a more complex control system in its implementation [7]. Therefore, the number of sensors should be reduced without affecting the effectiveness of the robot motion.

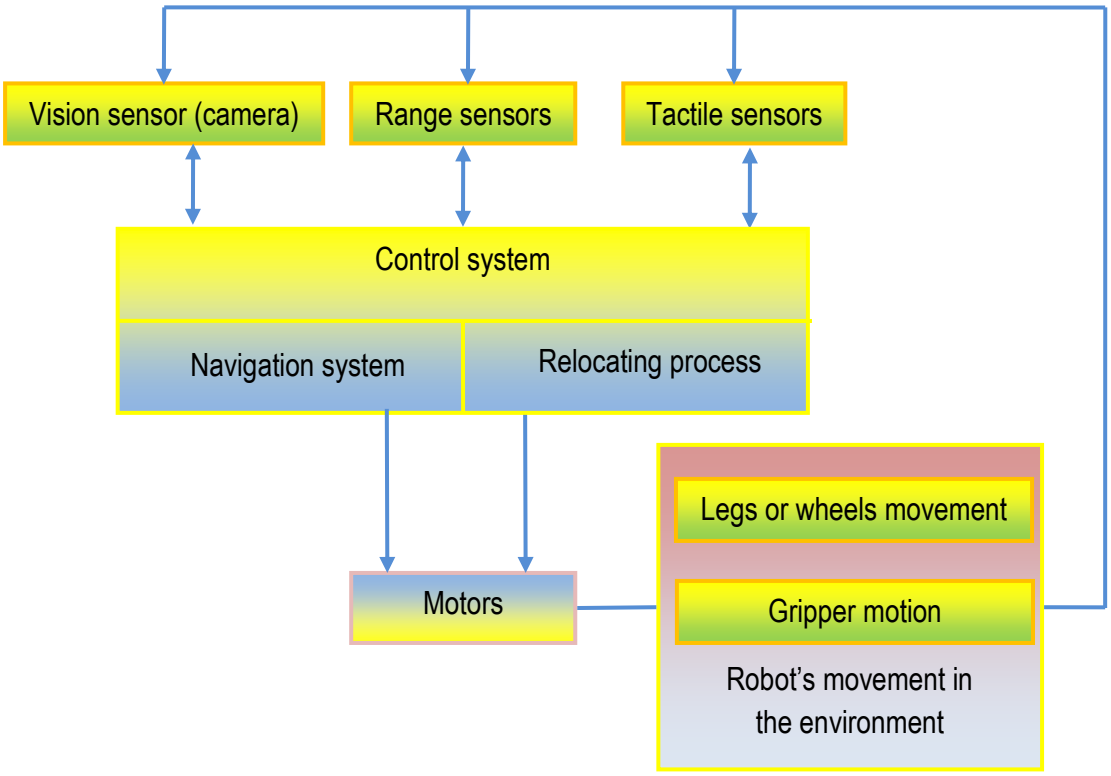


Figure 5.5. Control process

5.4.1 Sensor Types

The description of three types of sensors implemented in robot sensory platforms will follow.

5.4.1.1 Vision Sensors

In this study, a Logitech webcam model C200 [86] was used as the vision sensor. It takes images with a resolution of up to 640 by 480 pixels with the maximum rate of image capture of 30 frames per second. The sensor is connected to the main board by a USB port. The camera is fixed at the front of the hexapod mobile robot but is attached to a servo motor at the front of the wheeled robot, allowing rotation of 180°. Image processing approaches will be explained in Chapter 6.

5.4.1.2 Ultrasonic Range Sensors

An ultrasonic range sensor typically works by emitting a beam of sonic pulses and receives the returned echo that reflects from an object. The sensor computes the time interval between emitting the signal and receiving the echo to determine the distance to the object. The emitted signal has a cone shape that covers a fixed volume. The cone volume or beam width typically depends on the type of transducer. Beam width is the most important characteristic for choosing these sensors within robotics applications because it decides a bore-sight, which is the angle of ranging that the transducer covers in a direction straight-ahead [92]. For example, a sensor that emits a beam within a large bore-sight would discover objects that are not directly in the robot's path.

Three types of sonar devices are used: Devantech SRF02, SRF08 and SRF10 ultrasonic range finders [85]. The SRF02 ultrasonic range finders are low cost, small in size and have low power consumption. The minimum and maximum range distance measured by the sensor could be 15 and 200 cm, respectively. Its beam width is less than 55°. Conversely, the SRF08 and SRF10 sensors measure approximately the same distance interval of between 3 cm and 11 m. However, their beam widths are 55° and 72°, respectively. All these sensors can be operated within the I^2C mode, which allows communication of data between I^2C devices over two lines, the serial data (SDA) and serial clock (SCL) lines. I^2C devices are divided into two categories: masters

(microcontroller) and slaves (sensors). Master devices control the clock and generate START and STOP signals to operate slaves. Conversely, each slave must have a unique address and waits for the master commands to perform its work. A basic read or write sequence of master to slave is:

- Send the START bit
- Send the slave address
- Send the Read (1)/Write (0) bit
- Wait for/Send an acknowledge bit
- Send/Receive the data byte (8) bits
- Expect/Send acknowledge bit
- Send the STOP bit

The default shipped address of these sensors is 0xE0 but it can be changed to any of 16 addresses: E0, E2, E4, E6, E8, EA, EC, EE, F0, F2, F4, F6, F8, FA, FC or FE. Consequently, up to 16 SRF02s can be connected together on a single I2C bus with different addresses (Figure 5.6). To change the default shipped address, only one sensor has to be connected to the I2C bus and then three sequence commands are written in the correct order followed by the new address in code [93] (Figure 5.7).

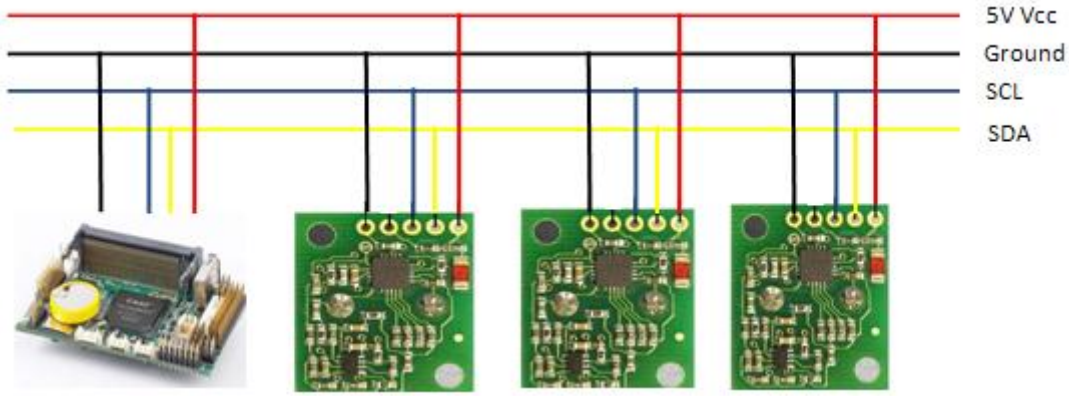


Figure 5.6. The sensors with I^2C

```
//----The code used to change the sensor address
int main()
{
    if (i2c_Initialize(I2CIRQ_DISABLE) == false) {
        printf("Failed to initialize I2C\n\n%s", roboio_GetErrMsg());
    };
    printf("Set I2C speed to 400Kbps");
    i2c0_SetSpeed(I2CMODE_AUTO, 400000);
    i2c0master_StartN(0xE0>>1, I2C_WRITE, 2);  //The defult address
    i2c0master_WriteN(0);
    i2c0master_WriteN(0xA0);                    //The first sequence
    wait_ms(70);                                //to change the address
    i2c0master_StartN(0xE0>>1, I2C_WRITE, 2);
    i2c0master_WriteN(0);
    i2c0master_WriteN(0xAA);                   //The second sequence
    wait_ms(70);
    i2c0master_StartN(0xE0>>1, I2C_WRITE, 2);
    i2c0master_WriteN(0);
    i2c0master_WriteN(0xA5);                  //The third sequence
    wait_ms(70);
    i2c0master_StartN(0xE0>>1, I2C_WRITE, 2);
    i2c0master_WriteN(0);
    i2c0master_WriteN(SRF02_Address);       //The new address
    i2c_Close();
}
```

Figure 5.7. The code to change the sensor's address

5.4.1.3 Tactile Sensors

Tactile sensors can provide information at the time of contact between the robot and other objects in an environment. These sensors are mostly used to calculate the amount of force applied by the robot's end effectors. They are also widely utilised for adjusting the legged robot's motion. Seven of the Interlink Electronics 0.2" Circular FSR sensors [94], which are very thin, robust, polymer thick film (PTF) devices that could measure forces between 1 and 100 N, are attached within the hexapod mobile robot: one sensor in each leg and one in the gripper. The legged sensors enable the robot to walk over rough terrain, which is the type of environment that the robot is designed for, while the gripper sensor allows the robot to control the force applied to the target when it is grasped and relocated.

The leg sensors control the rotation of the motors that provide the motion of the assigned leg. This is done by providing feedback to the controller (Figure 5.8). If the amount of force applied by the assigned leg reaches a pre-defined maximum value,

which is defined according to the robot's weight, the controller will stop the motors within the current angles. This presents three advantages for the robot. The first is that it allows adjustment of the robot legs to provide stability within a horizontal plane. The second is that it prevents any damage to the motors and the third is that it enables the robot to walk on uneven terrain. The gripper sensor controls the grip force that is applied to the object. If this force exceeds the pre-defined value, which depends on the grasped object's weight, the controller will stop the gripper servo motors' motion.

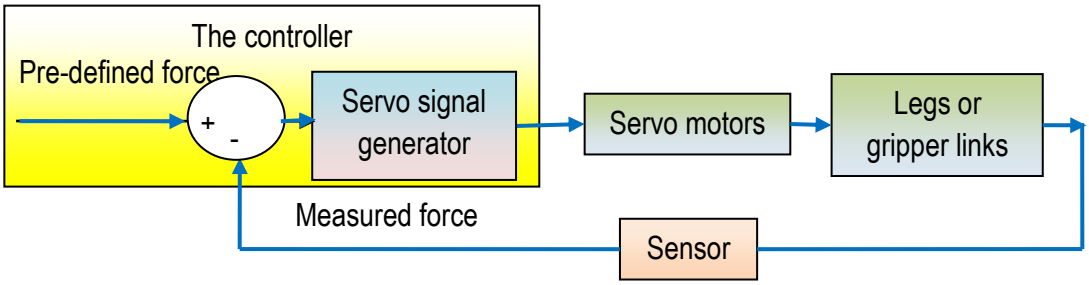


Figure 5.8. The legs and gripper controller

5.4.2 Distribution of Sensors and Motion Problem

One of the main objectives of this work is to find a suitable method for locating a small number of sensors on the robot's body that is ideal for autonomous navigation. In Chapters 3 and 4, an explanation was given of the two types of mobile robot platform, used to validate the approaches of this thesis. In the proposed navigation system, the robot navigates to the nearest wall (or similar construction) and then moves along that structure. If the robot using the built-in ultrasonic range sensor system detects an obstacle, it will navigate around that obstacle and then continue moving along the wall. While the robot is self-navigating in its environment, it continues to look for the target using the vision system. In this case, the main requirement is that the robot must keep driving at a constant distance from the wall, whilst also avoiding any obstacles in its way (Figure 5.9). Consequently, there are a number of requirements that the ultrasonic range sensor system has to guarantee:

- Maintaining a desired distance from a wall;
- correcting errors that come from the robot's legs or wheels slipping on the floor;
- detecting the boundaries of obstacles to enable navigation around them;

- detecting the nearest wall when the robot starts to navigate;
- discovering the distance between the robot and the walls or obstacles directly in the robot's path and
- controlling the robot when it approaches the target object.

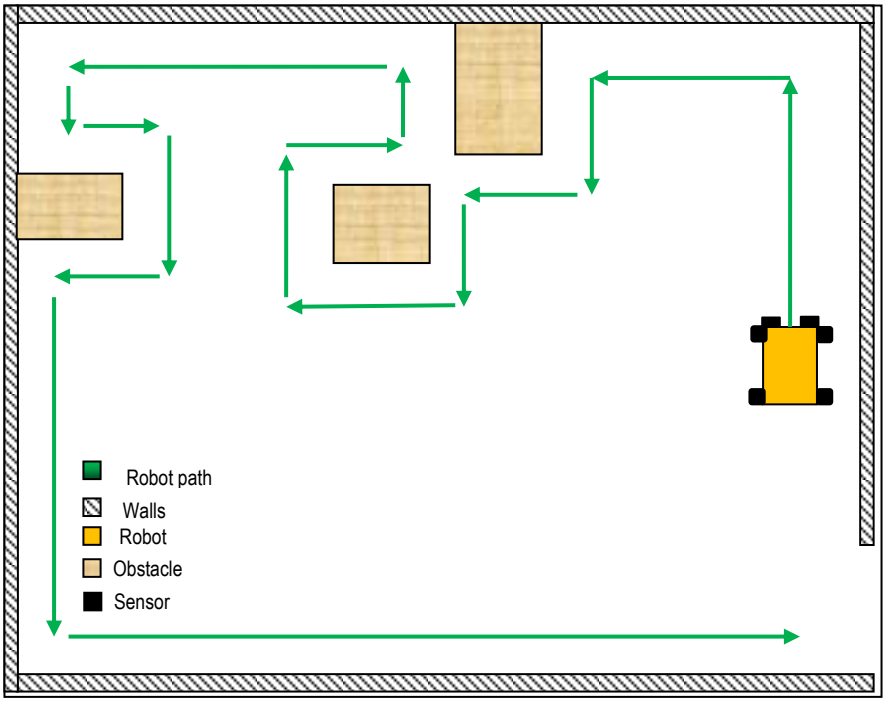


Figure 5.9. The robot navigation strategy

5.4.2.1 The Hexapod Mobile Robot Sensor Configuration

The hexapod mobile robot is 19 cm long, 13 cm wide and 15 cm high (7 cm from the ground). When the robot walks, its legs will be higher than the robot's body covering both sides of the robot. Consequently, if any of the ultrasonic sensors is mounted directly on the robot's body, it will continue detecting both the ground and the legs. The possibility of this problem increases when sensors that emit a beam with a large beam width angle are used. Consequently, the decision taken was to use the SRF02 ultrasonic range sensors because their beam width angle is narrow and suitable for this application. Six of these sensors are mounted: two sensors are on front and on each side of the robot (Figure 5.10). The front sensors measure the distance between the robot and the walls or obstacles in the robot's direct path. They are also used to control the robot when it approaches the target object. The side sensors help to locate the walls or obstacles beside the robot.

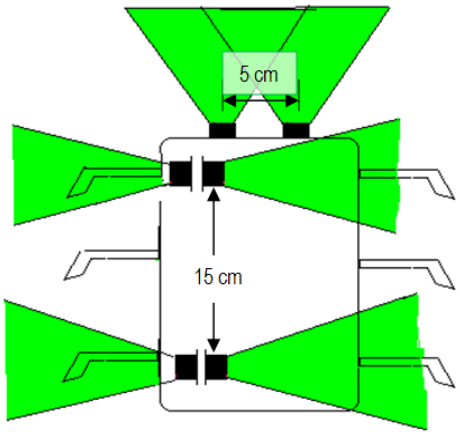


Figure 5.10. The hexapod's sensor platform

5.4.2.2 The Wheeled Mobile Robot Sensor Configuration

The wheeled robot was equipped with the same sensory system as the hexapod (see Figure 5.10). However, SRF10 and SRF08 range sensors were used instead of SRF02 and also three sensors are attached to the front of the robot instead of the two sensors used for the hexapod to increase the robot's sensing ability. Note that SRF10 sensors were mounted to the front because they are much smaller than SRF08. Experiments confirmed that these sensors are more accurate than SRF02; they can also detect a narrow object 6 m distant at an angle of 60 degrees. However, two problems materialised in using the SRF10 sensors. First, these sensors detect obstacles that are not directly in the robot's path. Second, they observe the ground as a wall or an obstacle. Therefore, some experiments, as in [92], were done to reduce the beam width angle. Therefore, a 3 cm long paper tube with a rectangular section with an area the same as the sensor area, was installed. The inside walls of the tube were covered by a thin layer of sponge. Thus, the beam width angle is reduced dramatically because the sponge absorbs and narrows the beam of the sensor. The sensors, after paper tubes were installed, only detected objects within an angle of 30 degrees or less.

5.5 The Motion Problem Analysis

In this section, the conditions under which the ultrasonic sensors mounted on the robot give enough information will be analysed, to determine both the robot motion and environment structure. Assume the robot moves in the environment that is shown in

Figure 5.11 and that it faces some motion problems that are assigned in Figures 5.12 and 5.13. Presume that:

- The robot navigates along the wall that is on its right side and it must keep moving between the two green lines (Figure 5.11). The robot should recognise the obstacles that the robot must move around without any collisions;
- (SRr) and (SRf) are the right-side rear and front distances, measured between the robot and the wall or the obstacle by the right-side sensors;
- (HDS) and (LDS) are the highest and lowest values in the desired distance interval that is assigned by the user and they specify the maximum and minimum distances between the robot and the wall that is at the robot's right-side. They must keep the robot moving between the green lines;
- (DdF) is the desired distance between the robot and the wall straight-ahead and is chosen by the user. The robot must change its direction when it comes closer to the wall than this line and
- (FDl) and (FDr) are the front distances between the robot and the objects or wall straight-ahead and they are measured by the front sensors mounted to the left and right, respectively (Figure 5.11). In the case of the wheeled robot, three sensors are used and therefore, (FDc) is added and it points to the central sensor measurement.

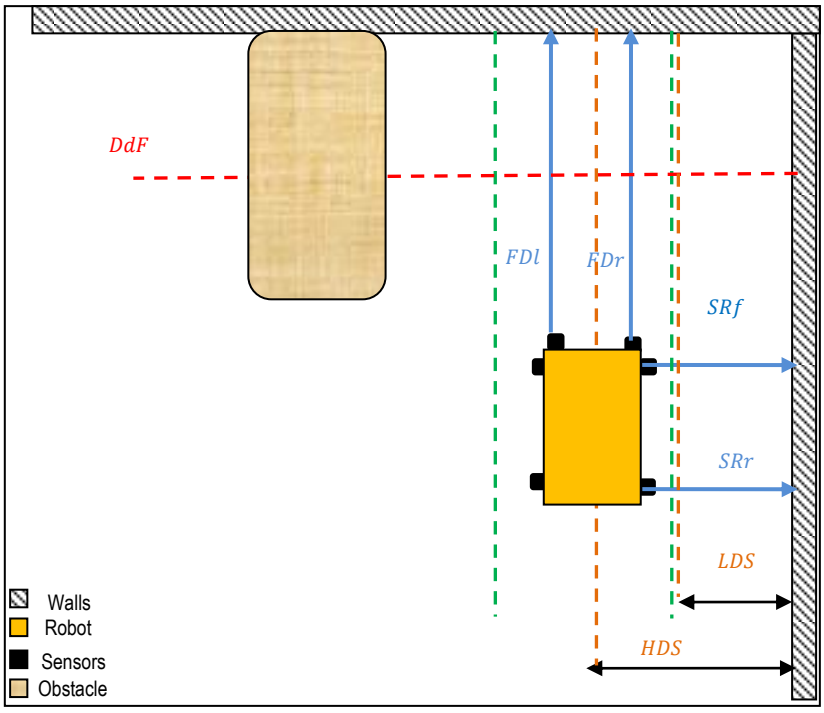


Figure 5.11. The robot motion analysis

Case 1 : In Figure 5.12A, the robot uses its front and right-side range sensors to determine the distance between its body and the walls. It will start moving in a straight forward motion, if: both distances (SRr) and (SRf), measured by the right-side sensors of the robot are equal and they enable the robot to move between the two green lines; AND the front sensors' measurements are greater than the desired front distance (DdF) that is assigned by the red line.

Case 2 : In Figures 5.12B and C, the robot must correct its location beside the wall, if the robot's sensors measure the same conditions as in Case 1. However, the right-side sensors' measurements (SRr) and (SRf) are not equal and thus, the error is given by

$$error = SRf - SRr \tag{5.1}$$

If the error is positive or negative, the robot will rotate right or left, respectively until the minimum difference between the two sensors' measurements, which is specified by the user to enable the robot to move within a minimum error, is reached. Notice that the amount of the rotation angle is specified by three values (small, average and large) based on the quantity of the error.

Case 3 : When any of the front-sensor measurements (FDl), (FDr) or (FDc) indicate that the robot has reached the wall straight-ahead, as shown in Figure 5.12D, the robot turns left until a straight forward space is opened that allows for continued motion of the robot, as shown in Figure 5.12E.

Case 4 : If the robot reaches an obstacle, it will behave in the same way as when it faces a wall. The robot keeps driving along the obstacle until its edge boundaries are detected by the robot's right-side sensors. In this case, the two echoes received by the two side transducers are not equal. For instance, if the measured distance by front-right-side sensor (SRf) is greater than the measurement of the rear-right-side sensor (SRr) and it is also greater than the maximum value in the desired distance interval (HDS), the robot starts to steer in a direction to the right. This continues until the robot detects the boundaries of the obstacle and both the right-side sensors' measurements (SRr) and (SRf) are identical. Then the robot moves forward and

continues to correct its location along the obstacle; this enables the robot to move around the obstacle (Figures 5.12F to H).

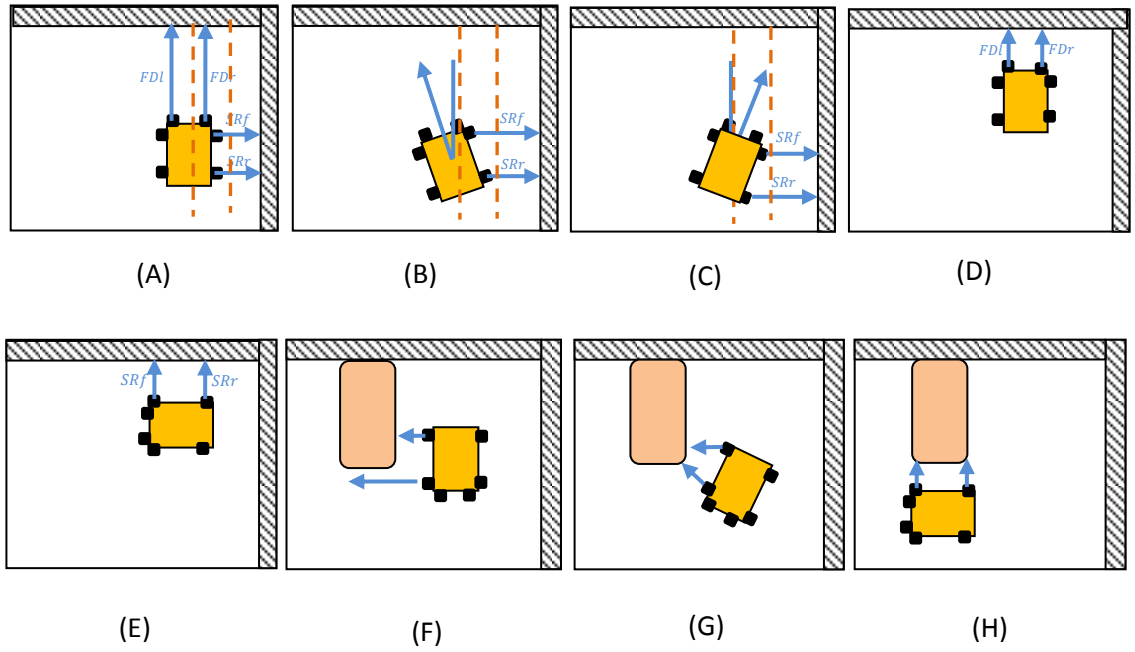


Figure 5.12. The robot's motion (when it moves between the desired values of the distance interval (*HDS*) and (*LDS*))

The above analysis suggests that the robot moves along the wall or a similar structure maintaining the desired distance that is between the maximum and minimum values of the desired distance interval (HDS) and (LDS) (Figure 5.11). The motion faults are now discussed should the robot drive outside of this interval. In this scenario, there are three states that are presented by the ultrasonic sensory measurements. The measurements might show that:

State 1 : There is a wall or an obstacle in the robot's path and that there is no free space to correct the faulty motion. Consequently, the robot discards any faults and the motion in Case 3 is performed.

State 2 : There is a wall or obstacle in the robot's path but the robot has enough free space to perform its motion. In this case, there are two conditions to correct any motion faults. The first occurs when the robot navigates along the wall with a distance that is closer than the minimum value of the desired distance interval (LDS) to the wall (Figure 5.13A). In this scenario, the robot performs the motion in

Case 5. The second situation happens when the robot moves along the wall with distance that is more than the maximum value of the desired distance interval (HDS) (Figure 5.13E); then the motion in Case 6 is executed.

Case 5 : The robot steers a specific counter-clockwise angle (α) that is assigned by the user (Figure 5.13B). It then moves straight forward until the rear-right-side sensor measures a distance (SRr) that is equal to or more than the result of $((LDS + HDS) / 2)$ (Figure 5.13C). Next, the robot turns back at angle (α) in a clockwise direction to locate itself along the wall (Figure 5.13D).

Case 6 : The robot turns a specific clockwise angle (α) (Figure 5.13F) and then moves straight forward until the rear-right-side sensor measures a distance that is equal to or less than the result of $((LDS + HDS) / 2)$ (Figure 5.13G). The robot then rotates back (α) angle in the counter-clockwise direction to locate itself beside the wall (Figure 5.13H).

State 3 : There is no wall or obstacle in the robot's path and therefore, the robot performs similar motion to State 2.

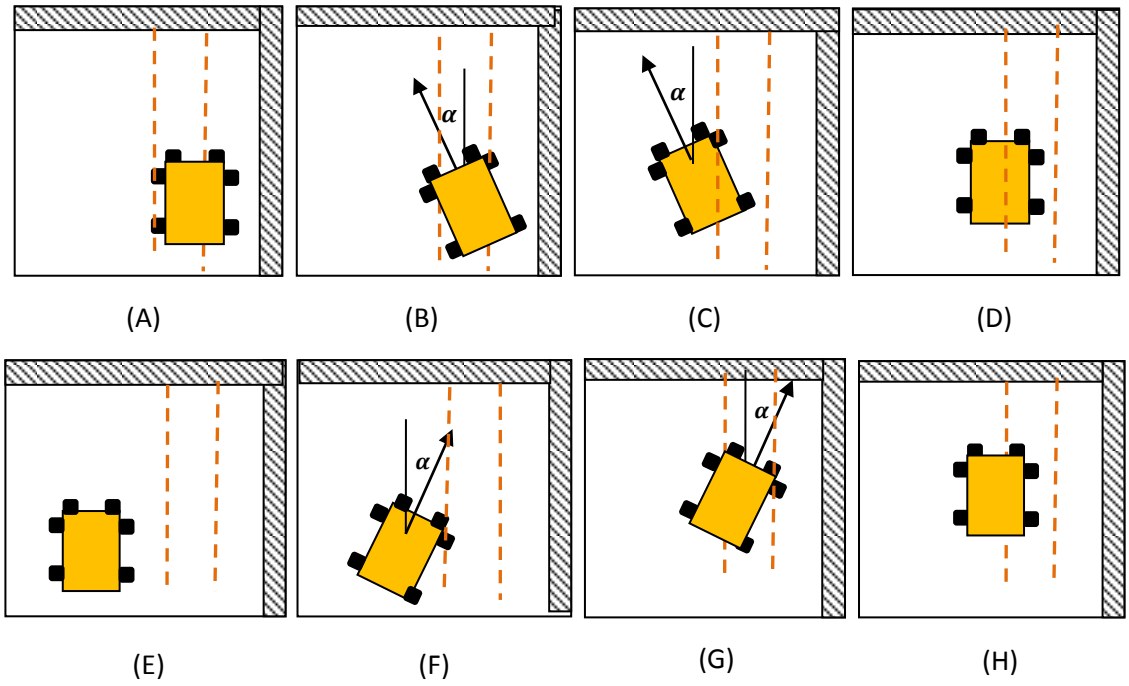


Figure 5.13. The robot's motion states (when it navigates outside of the desired values of the distance interval (*HDS*) and (*LDS*))

5.6 Experimental Results

The algorithm in Figure 5.4 was implemented taking consideration of the motion analysis in Section 5.5. All code was written using C++ and C languages and then tested with both the hexapod and wheeled robots. The experiments were carried out to evaluate the navigation system, which was tested without the robot searching for the target (object). As such, the image processing functions, searching for the target in the image, were not called on by the navigation system. The objective of the robots was to navigate along the walls from the starting point to the target position, avoiding collisions with obstacles in their paths. The obstacles were rectangular shaped objects of various sizes.

First, the experiments were carried out with the hexapod in different environments to evaluate its control system (see [1]). Figure 5.14 represents a graph of an example of the experimental environment (3000 × 4000 mm) that was a part of the laboratory room. The yellow rectangle represents the robot and the red tile denotes the landmark (the nearest wall). The green arrows denote the robot's navigation path and the dashed, rectangular blocks represent the obstacles in the robot's path. The letter scripts in Figure 5.14 point to the locations of the robot in Figure 5.15.

The robot moved from its initial location and approached the nearest assigned wall (Figure 5.15A). After approaching the wall, the robot turned in the desired direction (in this case to the right). When the robot reached the first obstacle, it considered this obstacle as a wall, turned right and navigated along this obstacle, as shown in Figure 5.15B. The robot repeated the process for the second obstacle until it reached the boundaries of the obstacle. In this case, the robot turned in the left direction, in order to keep the obstacle on its left. The processes were repeated until the robot went back to the wall.

The hexapod was initially equipped with just one front, ultrasonic range sensor. However, the robot hit obstacles that were located straight ahead but outside of the sensor's beam width. Consequently, another sensor was added, as shown in Figure 5.10. In this case, the obstacle avoidance process improved significantly. The navigation system was tested without the robot searching for the target object. The system was

executed successfully, proving the effectiveness of the methodology used. The hexapod was successful in analysing the range sensors' information but it moved slowly.

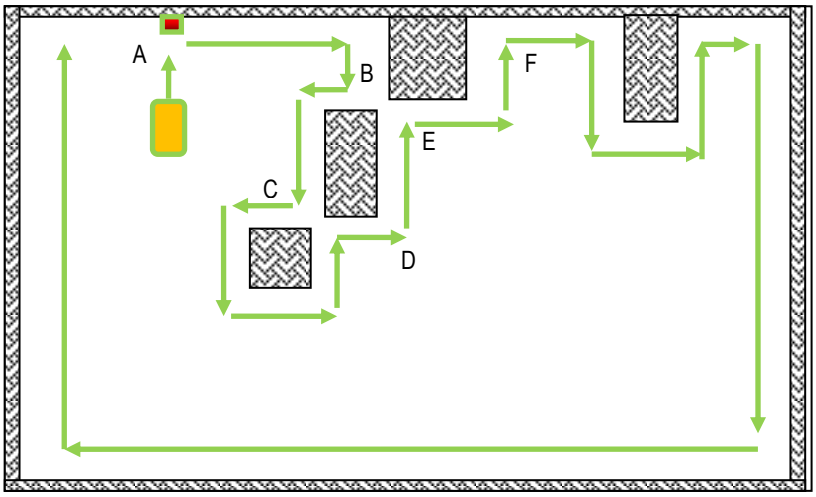


Figure 5.14. The sample environment

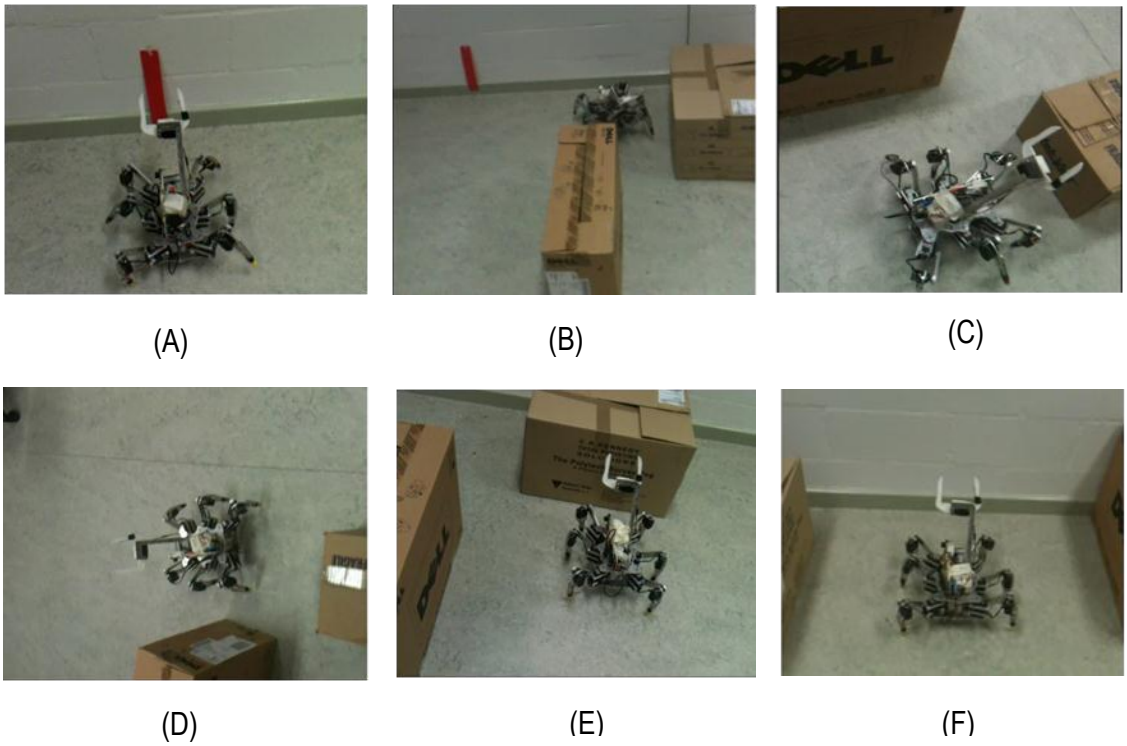


Figure 5.15. The hexapod mobile robot motion

Then the wheeled mobile robot was tested without using the camera and the PC motherboard. All motion codes were implemented in the microcontroller. Figure 5.16 shows a map of the environment in which the robot was used. In this case, the robot started its navigation from point (A) and its objective was to reach point (F) without

colliding with any obstacles that were located in its path. The letter scripts in Figure 5.16 point to the locations of the robot in Figure 7.17. This robot moved much more quickly than the hexapod.

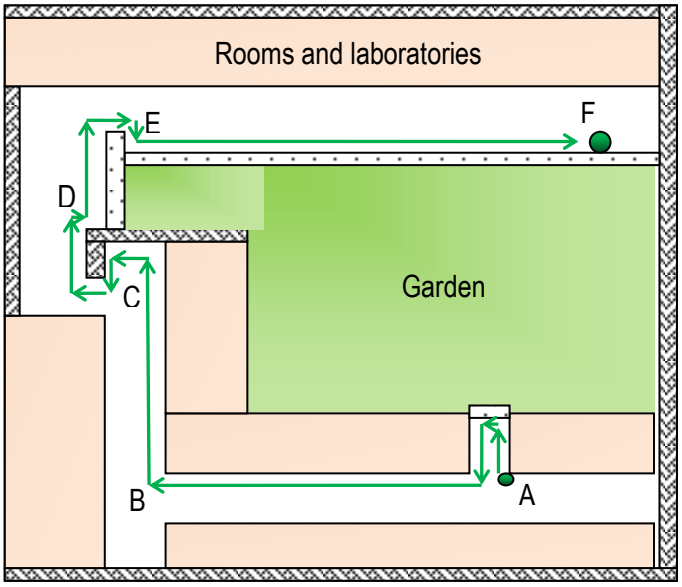


Figure 5.16. The corridor environment where the wheeled robot's motion was tested

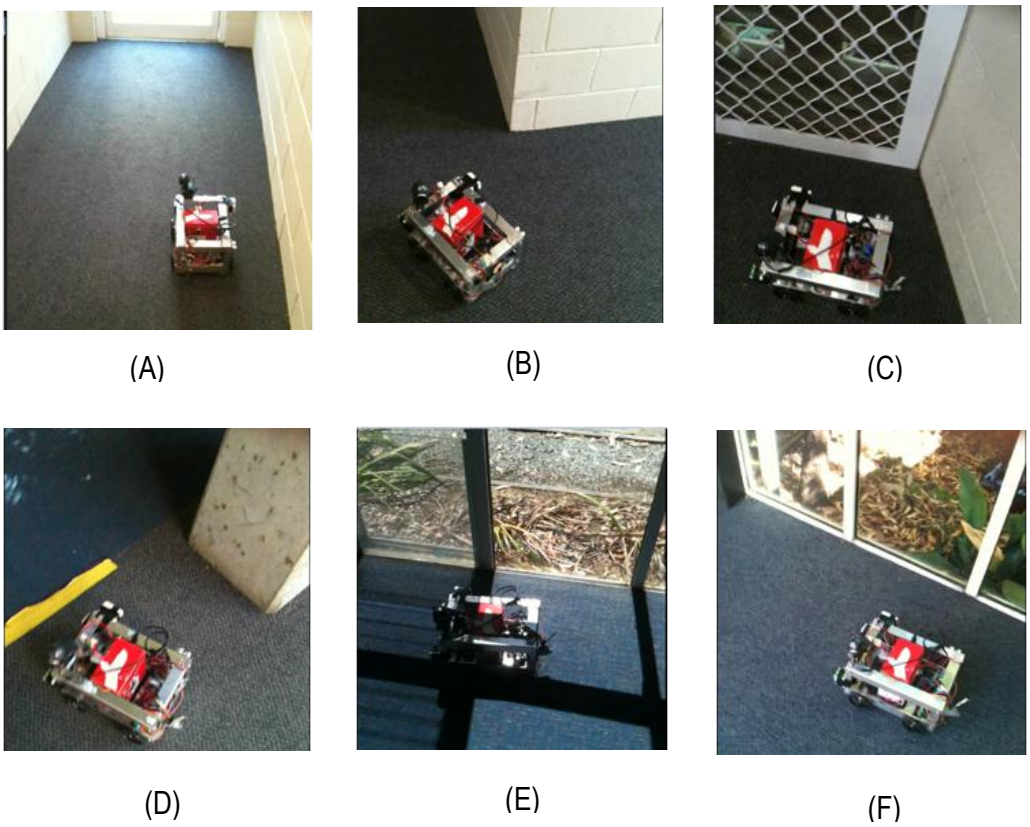


Figure 5.17. The wheeled robot motion

5.7 Conclusions

This chapter introduces the family of Bug algorithms and describes its basic concepts. The main reasons for choosing the Bug algorithms are to implement the exploration path: they are appropriate for a robot designed to navigate in an unknown environment that is constantly changing; they are designed to follow walls or obstacles; and they guide the robot to its goal by saving few of the environmental information. Subsequently, the Bug-like algorithm that uses the spirit of the Bug algorithms is introduced and explained. Next, the sensor configurations that help both robots to navigate along the walls or similar structures and avoid the obstacles are described. The motion problems are then analysed in detail. Finally, experiments were conducted on both robots to demonstrate the effectiveness of the methodology used. It was concluded that it is possible to design and implement a navigation system within a small quantity of sensors if they are attached and employed effectively on the robot's body.

Chapter 6 Object Detection-Based on Vision System

6.1 Introduction

As the mobile robot performs a visual search, the most important aspect it must have, is the ability to detect and recognise the desired objects in an environment. Vision-based object recognition is important for undertaking the exploration task; it involves using a vision sensor and employing an optimal object recognition technique. This chapter discusses the vision-based object detection techniques that are used in this study.

6.2 Object Detection Problem

The mobile robot's object recognition tools must deal with a 3-D environment in real-time. Specifically, the robot requires the ability to detect the target object from different sides, distances and rotation angles. The appearance of an object in an image varies from one viewpoint to another. Variations due to environmental conditions, target characteristics and sensor efficiency also complicate object recognition. Lighting, texture and background colour are the major relevant environmental conditions and the key characteristics of the target include features of texture and contrast. Various methods can be used in order to detect the object in the image but they are severely limited by their need for training data and sophisticated algorithms [95].

6.3 Object Detection

The main objective of image processing is detecting the target object and the delivery location (see Figure 6.1). This can be achieved using several methods, including: colour segmentation, template matching and speeded up robust features (SURF).

6.3.1 Object Detection by Colour Segmentation

Colour segmentation, which is one of the basic techniques in computer vision, is an analytical process that recognises image content based on variations in colour and texture. It has been widely used for object recognition based on its individual profile within these variations. Specifically, RGB (red, green and blue colour space) and HSI

(Hue, Saturation and Intensity colour space) are commonly used for colour segmentation. In this study, RGB and HSI were initially employed to detect the target and the delivery location.


Figure 6.1. Some objects are wanted to be detected: the image on the left side is the target objects while the image on the right side is the delivery location

6.3.1.1 Segmentation by RGB

In RGB colour space, each individual colour involves three weights: red (R), green (G) and blue (B). Eight bits specify each R, G and B in the RGB colour space so that the colour intensity for each weight is between 0 and 255. In other words, all possible colours in the RGB colour space are made from different intensities of red, green and blue colours. For instance, the colour white, which has maximum intensity (255) of red, green and blue, is changed to yellow by decreasing the blue from its maximum to its minimum value. The brightest yellow, which is made from the maximum levels of red and green and the minimum level of blue, is transferred to dark yellow by reducing the red and green values. Figure 6.2 explains how RGB colour space represents colours. Two colour segmentation methods were tested using the RGB colour space.

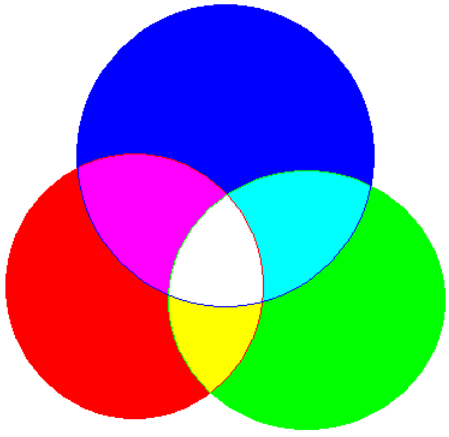


Figure 6.2. RGB colour space

Method (1)

The first method was achieved by using the original 24-bit image directly to segment the desired colour. Four rules were used for the task of object detection based on colour:

1- $\{rl \leq r(u,v) \leq rh\}$ means that the primary colour components (red) should be between the maximum (rh) and the minimum (rl) threshold values
2- $\{gl \leq g(u,v) \leq gh\}$ means that the primary colour components (green) should be between the maximum (gh) and the minimum (gl) threshold values
3- $\{bl \leq b(u,v) \leq bh\}$ means that the primary colour components (blue) should be between the maximum (bh) and the minimum (bl) threshold values
4- Regarding the object colour, choose the absolute of one of these forms: $|r(i,j)-g(i,j)|$, $|r(i,j)-b(i,j)|$ OR $|b(i,j)-g(i,j)| \leq Threshold\ differences$. For instance, if the red object is to be detected, this value will be the greatest absolute difference value between the green and the blue abs $|b(i,j)-g(i,j)|$.

In this method, the segmentation code reads and compares the intensity weight value of each pixel in the input image using the above four rules. Then, the results of the comparison are combined using the AND logical operation. If the pixels satisfy these rules, then the pixels are considered to be the object colour. In this case, these pixels are given the maximum value (255). Otherwise, those pixels that do not satisfy the four rules are given the minimum intensity (0). This process converts the colour image to a binary image (Figure 6.3).

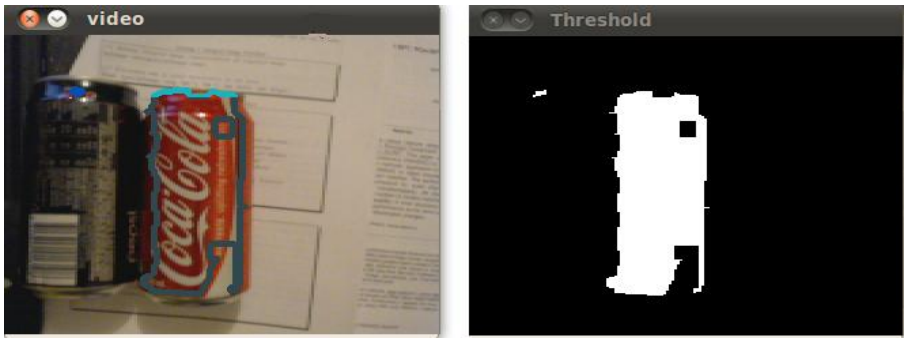


Figure 6.3. Object recognition using its colour (method 1): the image on the left side is the original image while the image on the right side is a binary image

The maximum and minimum threshold values of the object colour weights are determined by using some photo manipulation software, such as MS Paint (Figure 6.4). Then, these values are used with the above four rules to write the segmentation code that determines whether each image pixel either follows the object colour or not.

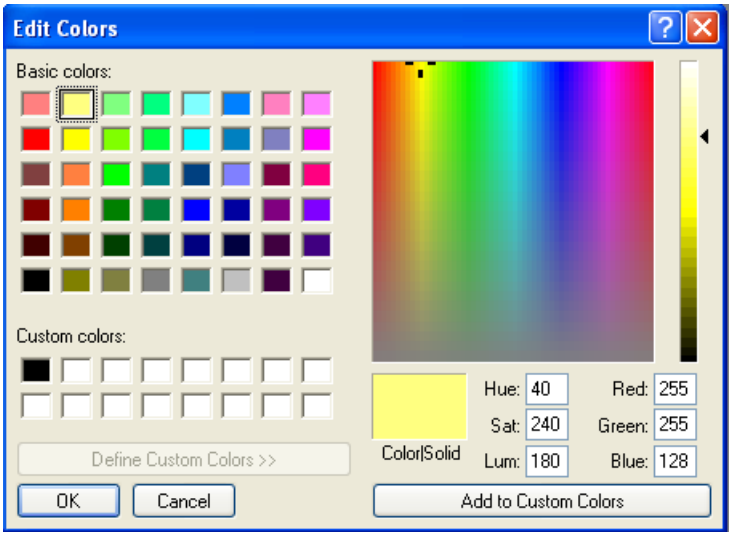


Figure 6.4. MS Paint software

Method (2)

The second method is carried out by splitting the 24-bit colour image into three 8 bit images; one for each colour intensity channel. Next, each is segmented separately and then re-combined. First, the maximum and minimum threshold values for each colour's channel are determined and these values are used to segment each colour channel image. Finally, the three images are combined in a way that releases the object pixel in the image. For example, if the red object is to be detected, the red channel image is segmented using the corresponding threshold values. Then, the other two channel images are segmented by releasing the minimum values of blue and green in the red colour, that is, in the object colour. Next, the green and blue channel images are combined additively. The result is subtracted from the red channel image achieved from the red channel image segmentation. This process produces a binary image in which the white colour defines the object pixels and the black colour the background. In this example, if the blue and green colours are segmented from minimum value (0) to maximum value (255), this will produce detection of only a pure red object that does

not contain any blue or green colour. Figure 6.5 shows the result of recognising a Coca Cola can by this method. The latter method has proven to be more effective than the former (see section 6.5).

Figure 6.5. Recognition a Coca Cola Can (method 2)

6.3.1.2 Segmentation by HSI

In HSI colour space, each colour with all its status from darkest to brightest is assigned by a particular period of Hue values. The amount of the original colour that is mixed with white colour is specified by a Saturation value and the brightness of the colour is assigned by Intensity values. Colour segmentation based on HSI is done to utilise the object's colour content in the input image. Figure 6.6 shows how the HSI colour space symbolises colours. The Hue range is between 0 and 360 degrees, while the Saturation and Intensity components range between 0 and 1 (see [96]). The segmentation process is achieved using the following steps:

1. Determine the object colour interval of the HSI colour space;
2. Convert the image contents from RGB colour space to HSI colour space; and
3. Apply the segmentation method.

The object colour HSI interval is computed by taking an image of the object for which the robot will search. Then, the image is sent to some photo manipulation software, such as MS Paint, in order to assign the object colour's HSI interval. This is done by choosing the object colour in the image and editing the colour to read the HSI values. The process is repeated within different locations on the object's projection in the image

until the HSI interval defines the object's colour. Figure 6.4 depicts the editing colour screen in the MS Paint software showing for each colour that there are HSI and RGB values. The Hue value in MS Paint is between 0 and 239.

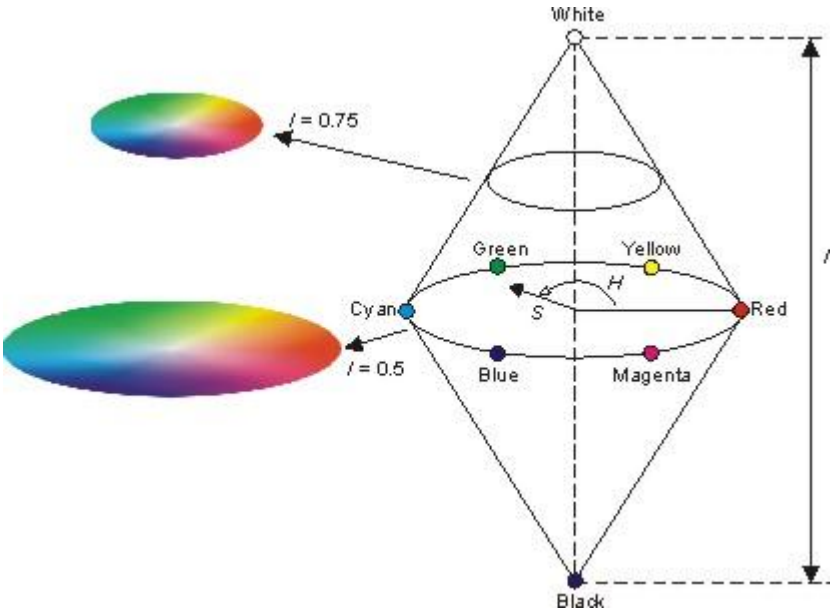


Figure 6.6. HSI colour space [96]

OpenCV is used to convert the RGB image to an HSI image; its Hue value is between 0 and 179 and therefore, the Hue value from MS Paint has to be scaled by multiplying it by 180/240. After converting from the RGB colour space to the HSI colour space, the H, S and I components of each pixel in the image I$(u,\ v)$ are compared with the pre-determined HSI interval, as in the equation (5.1). Next, the results are combined using the AND logical operation to determine whether the pixel follows the object colour or not. If all three values of the H, S, and I in that pixel are within the stated HSI ranges, then the pixel is considered to be following the object colour. In this case, this pixel is given the maximum value (255). Otherwise, it is given the minimum intensity (0). This procedure is repeated for all other pixels to segment the image and convert it to a binary image B(u, v) (see Figure 6.7).

$$B(u,v) = \begin{cases} 1, & (Hl \le H(u,v) \le Hh)\ AND (Sl \le S(u,v) \le Lh)\ AND\ (Il \le I(u,v) \le Ih) \\ 0, & Otherwise \end{cases} \quad (6.1)$$

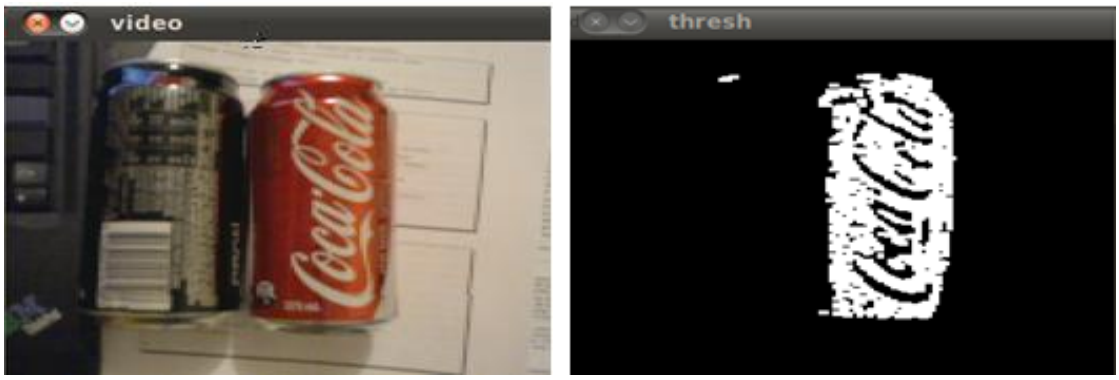


Figure 6.7. Recognition of a Coca Cola can by HSI: the image on the left side is the original image while the image on the right side is a binary image

6.3.1.3 Object features

Once the object is detected, its location relative to the robot must be determined. This is done by calculating the features of the object in the image, which include the object's shape (ratio of height to width), area (number of pixels) and centroid (the centre coordinates). As such, some measurements for the camera calibration should be made to relate the actual information of the distance and orientation between the robot and the object, to the object's features in the image (for more information, see [97, 98]).

In this study, the object's orientation was determined by finding the target's horizontal location in the image. If this location was not in the middle of the image plane, the robot turned to achieve this outcome. Then, the robot determined the distance to the object using the front ultrasonic range sensors. For this, two of the object's features are needed: the area and centroid of the object in the image. After the colour segmentation process and detection of the object, its area and centre coordinates are determined by using the technique proposed by [61, 97]:

$$Mp,q = \iint x^p y^q f(x,y)\, dxdy \qquad (6.2)$$

$$p,q = 0,1,2,\ldots.$$

in which Mp,q represents the moments with the rank (p,q); $f(x,y)$, which in a binary image is 1 when the pixel follows the detected object, otherwise it is 0 which represents a continuous 2D image. $x\ and\ y$ are the coordinates of the pixels in the image. The area of the object in image A is represented by $M_{0,0;}$ and the centre coordinates of the object in this image are represented by $X=M_{1,0}/M_{0,0}$ and $Y=M_{0,1}/M_{0,0.}$

After the area and centre coordinates of an object in the image is calculated, the resulting values are then forwarded as input signals to the robot controller. Note that in both RGB and HSI colour segmentation methods, some image noise might appear in the binary image background because of the segmentation process; therefore, this noise must be removed or reduced. This was done using the morphological opening operation, which is performed by eroding the image and then dilating it (see [99]). This operator was used because it eliminates all small noises and keeps the shape and area of the target object in the image. Figure 6.8 shows the pseudocode of the entire process.

Name: Object detection by colour segmentation
Input: Image = Capture Image ();
Output: The object features (area and Central coordinates), or Null if the object is not found.

1- Area = Null, X-axis = Null, Y-axis = Null;
2- Segment Image(Image);
3- Apply morphological opening operation (to reduce image's noise);
4- **if** (Object found) **then**
5- Determine Area, X-axis and Y-axis;
6- **end if**;
7- **return** Area, X-axis and Y-axis;

Figure 6.8. Object detection algorithm

6.3.2 Object Detection by Template Matching

As mentioned before, when the robot searches for the object, there are three problems regarding the object's invariant features: scaling, rotation and the 3-D models of the object (the camera's viewing angles). These can be solved in a template matching technique by using various template sizes with different possible rotations and object sides [95]. However, it was noticed that this is an extremely slow process in real-time image processing. Accordingly, the second problem is ignored by assuming that the robot approaches the object in a constant vertical direction. Other assumptions made to reduce the number of templates are explained in the template matching procedure as below.

Template matching can be performed by using colour or grey scale images; both the template and the original image must have the same format. In the former, the calculation is made for each channel in the image while the latter is carried out in only one channel. Thus, template matching using the grey scale will be faster than using the colour image. The method of template matching that is explained in [95] was adapted and used to match the 3D object in real-time image I:

1. Using 3D model (object), create 2D object projection templates $T(wi,\ hi,\ \Theta ji,\ di)$, where $(wi,\ hi)$ is the template dimension in location (i), Θji is the object's side views ($j =$ 1 to 4) and di is the object distance from the robot (60 to 150 cm in 30 cm steps)
2. Convert the template T and the captured image I to grey scale, if necessary
3. Find the best match R in I for T using template matching algorithms
4. Find the centre of T in the image and send it to the robot.

In step 1, an infinite number of template images can be created from the 3D object with different distances from the robot. However, if all poses and side projections of the 3D object are taken into consideration, it will be computationally expensive [95]. Therefore, only the four side 2D projections of the object are considered, together with the distance between the robot and the object (60 to 150 cm in 30 cm steps). This will produce 16 template images.

In step 2, both the template and the captured image can be used either as a grey scale or as a colour image. The size of the template is $T(w,\ h)$ and the captured image is $I(W,\ H)$. The size of the resulting image R that holds the correlation number is $R(W-w+1,\ H-h+1)$. In step 3, template matching algorithms move (by sliding) the patch of $T(w,\ h)$ through the $I(W,\ H)$, one pixel at a time (left to right, up and down). At each location, the algorithms compare the data of T to the data of the particular area of I and store the comparative result in R. The algorithms also calculate how "good" or "bad" the match (correlation) is in that location. There are two options to implement the template matching methods. The first is done by using existing libraries in computer vision, such as OpenCV software; the second option would be to write the entire set of code. The former is suitable in this project in order to save time. In OpenCV software, six template matching algorithms have been implemented [100]:

i) method=CV_TM_SQDIFF

It is a square difference matching method that differentially matches the template against the original image; therefore, a perfect match is 0.

$$R(x,y) = \sum_{x',y'}[T(x',y') - I(x+x',y+y')]^2 \quad (6.3)$$

ii) method=CV_TM_CCORR

It is a correlation matching method that multiplicatively matches the template within the original image; therefore, the perfect match is the highest value.

$$R(x,y) = \sum_{x',y'}[T(x',y')\ .\ I(x+x',y+y')]^2 \quad (6.4)$$

iii) method=CV_TM_CCOEFF

It is a correlation coefficient matching method that matches the template against the original image relative to their means.

$$R(x,y) = \sum_{x',y'}[T'(x',y')\ .\ I'(x+x',y+y')]^2 \quad (6.5)$$

There is normalised version for each of the above-mentioned techniques. The normalised methods are used to decrease the effects of the light differences between the original image and the template [100].

iv) method=CV_TM_SQDIFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'}[T(x',y') - I(x+x',y+y')]^2}{\sqrt{\sum_{x',y'} T(x',y')^2 . \sum_{x',y'} I(x+x',y+y')^2}} \quad (6.6)$$

v) method=CV_TM_CCORR_NORMED

$$R(x,y) = \frac{\sum_{x',y'}[T(x',y')\ .\ I(x+x',y+y')]^2}{\sqrt{\sum_{x',y'} T(x',y')^2 . \sum_{x',y'} I(x+x',y+y')^2}} \quad (6.7)$$

vi) method=CV_TM_CCOEFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'}[T'(x',y')\ .\ I'(x+x',y+y')]^2}{\sqrt{\sum_{x',y'} T(x',y')^2 . \sum_{x',y'} I(x+x',y+y')^2}} \quad (6.8)$$

in which $x' = 0 \ldots\ldots\ldots w-1$, $y' = 0 \ldots\ldots\ldots h-1$,

$T'(x',y') = T(x',y') - 1/((w.h)\ .\ \sum_{x'',y''} T(x'',y''))$, and

$I'(x+x',y+y') = I(x+x',y+y') - 1/((w.h)\ .\ \sum_{x'',y''} I(x+x'',y+y''))$.

When using NORMED calculations, the maximum value of a correlation will be 1, while other calculations may generate much larger numbers. Therefore, the methods used with NORMED are more suitable for this project because the correlation threshold that represents the "good" match is more easily specified. CCORR and CCOEFF calculations will return correlations with a value of 1 for a perfect match, while SQDIFF will return 0 for such a match. The best match correlation can be found by the minMaxLoc() Function, which can then be compared with the correlation threshold. The final step (4) is executed if the best match value is equal or greater than the threshold. In this step, the location of the best match is specified using the previous function. Then, the centre of the perfect matching area is determined in the original image. The centre of the area represents the desired object's centre; it is $C(x + w/2, y + h/2)$. Figure 6.9 presents the pseudocode for all steps in this process. Figure 6.10 shows the results of detecting and recognising two objects by using colour and grey scale images; the red rectangles (the black rectangles in Figures 6.10C and D) represent the objects that are found in the real-time images (see Figures 6.10A and B). The next Chapter explains how the object's centre and the match value are used to send signals to the robot's controller for managing its motion.

Name: Object detection by Template matching
Input: Image = Capture Image (), Template (Number) = Template Image (Number)
Output: The object features (correlation and Central coordinates), or Null if the object is not found.

```
1-  Correlation= Null, X-axis = Null, Y-axis = Null;
2-  for Number = 1 to 16 do
3-      Match (Image & Template (Number));
4-      Determine Correlation;
5-  if  (Correlation ≥ threshold) then
6-          Determine X-axis, Y-axis ;
7-  go to return ;
8-  end if;
9-  end for
10- return Correlation, X-axis and Y-axis;
```

Figure 6.9. Algorithm for object detection using template matching

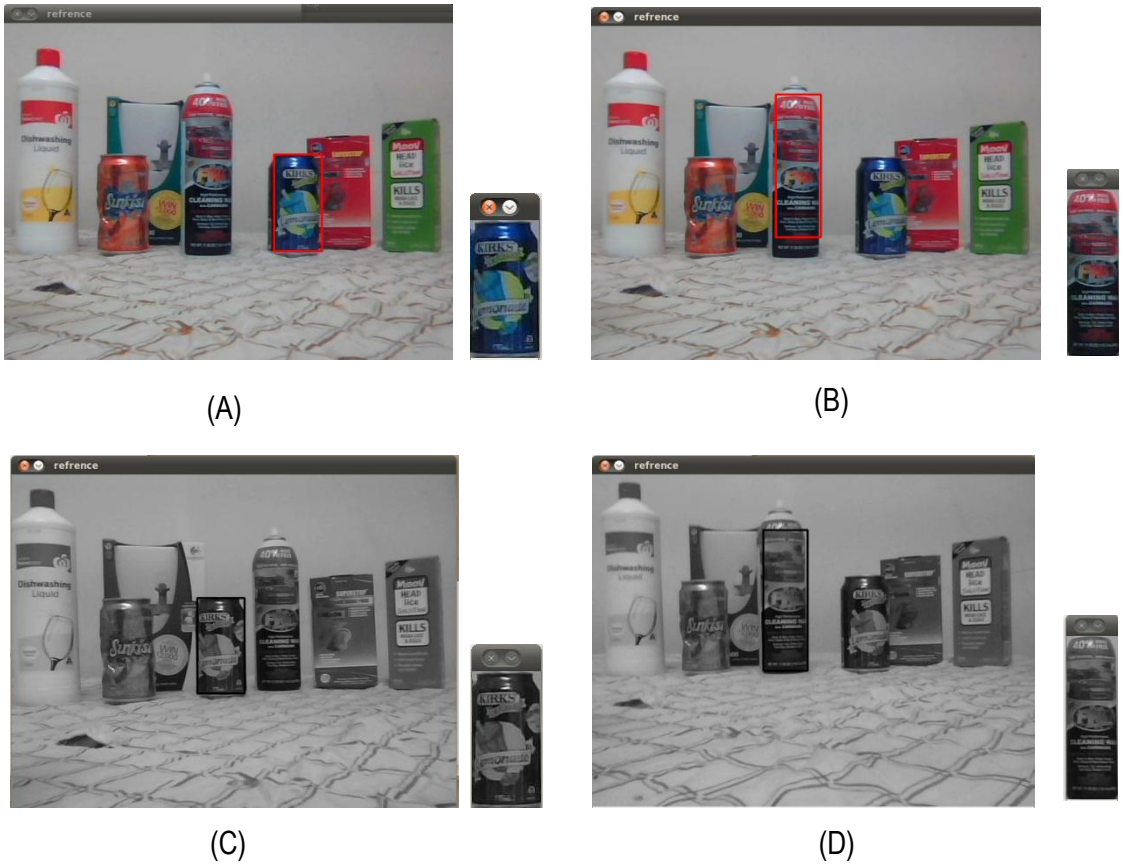


Figure 6.10. Template matching results, (A and B) using a colour images and (C and D) using a grey scale images: original images are on the left and template images are on the right

6.3.3 Detecting object by Speeded Up Robust Features (SURF)

This strategy has been successfully implemented and tested. Figure 6.11 illustrates the results of detecting and recognising two objects by using this method. The small images are the templates of the objects and the big images are the captured images in real-time. First, the template of an object to be found is taken; then the interesting points and descriptors of the template image are extracted and calculated. The small circles, which represent the SURF interesting points, can be clearly seen in Figures 6.11B and D. Next, the interesting points and descriptors of the environment (in the real time image) are extracted and determined. Then, the matching process is undertaken by comparing the interesting points and descriptors of both the template and the real-time image. The results of the detection process are shown Figures 6.11A and C; the green rectangles represent the objects that are found in the real-time images. The number of interesting

points, which is the same in both images, is determined and the average of their horizontal centre coordinates in the captured image is calculated and fed to the controller. If the number of matching interesting points is equal or greater than a threshold number, then the object is detected. This method has proved to be quicker than template matching (see section 6.5). Figure 6.12 presents the pseudocode for the SURF process.

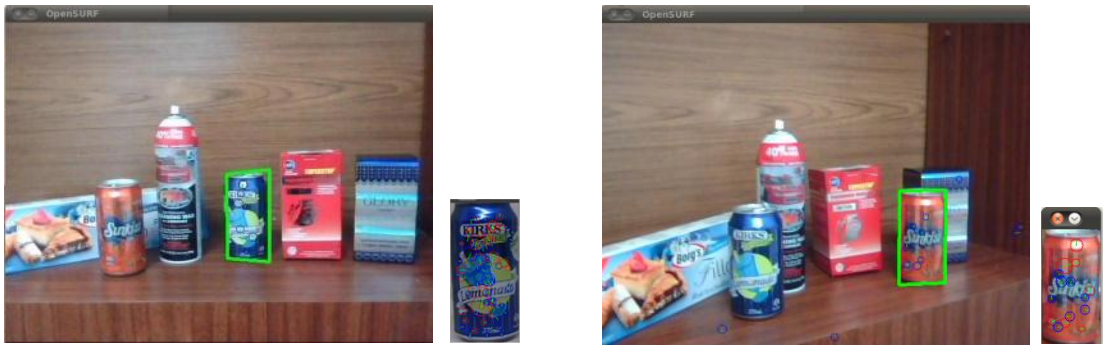


Figure 6.11. Results of SURF method; the small images are the templates of the objects and the large images are the captured images

Name: Object detection by SURF
Input: Image = Capture Image (), Template = Template Image ()
Output: The object features (Matched points and Central coordinates), or Null if the object is not found.

1. Matched points= Null, X-axis = Null, Y-axis = Null;
2. SURF (Template);
3. Find Interest points and Descriptors;
4. SURF (Image);
5. Find Interest points and Descriptors;
6. Match (Image (Interest points and Descriptors) & Template (Interest points and Descriptors));
7. **if** (Matched points $\geq$ threshold) **then**
8. Determine X-axis, Y-axis ;
9. **end if**;
10. **return** Matched points, X-axis and Y-axis;

Figure 6.12. Algorithm for object detection using SURF

6.4 The landmark design

In order to identify the final (delivery) location (the same as the start point) in an environment, a mobile robot needs to observe characteristics of this location. Fast determination of the location's features by the camera is crucial in real-time navigation. Therefore, a cylindrical shaped artificial landmark that has two solid colours (green and blue) was used, as shown in Figure 6.14A. The main advantage of using the cylindrical shaped landmark is that it appears the same from any side direction. The two-colour landmark pattern was chosen because it is less likely to be confused with the background environment and can also provide a more accurate detection process.

Detecting the landmark begins by performing colour segmentation on the captured image to find the green colour component of the landmark (Figure 6.14B). If the green part is recognised, its area and central coordinates are determined. Then, the captured image is segmented to detect the landmark's blue component (Figures 6.14C and D), followed by the determination of its area and central coordinates. If the areas of both colours and their horizontal coordinates are similar and vertical, the centre of the green area is above that of the blue, and this indicates that the location is found. In this case the robot must detect the landmarks, as shown in Figure 6.14E. When the landmark is detected, the main task for the robot is to find out the landmark's location and then approach it by keeping its image within the centre of the image plane (as will be explained in Chapter 7). However, if any of the previous conditions are not met, the robot continues its search for the location.

The same landmark design was used to mark the entrance (the doors if they are open) but different colours (red and yellow) were used. This landmark enables the robot to finds out that either it encounters the entrance of the environment or not. If so, the robot steers and moves beside that entrance (see Chapter 7). The same strategy as previously outlined is adapted to detect and recognise these landmarks. Figure 6.13 presents the pseudocode for the landmark detection procedure.

Name: Landmark detection by colour segmentation
Input: Image = Capture Image ();
Output: The landmark features (areas and Central coordinates), or Null if the landmark is not found.

```
Area1 = Area2 = Null, X-axis1 = X-axis2 = Null, Y-axis1 = Y-axis2 = Null;
Segment Image (Image) for the landmark's upper part;
Apply morphological opening operation (to reduce image's noise);
if (upper part detected (Area1)) then
    Segment Image (image) for the landmark's lower part;
    Apply morphological opening operation (to reduce image's noise);
    If (lower part detected (Area2)) then
        Determine Area1, X-axis1, Y-axis1, Area2, X-axis2 and Y-axis2;
    end if;
end if;
return Area1, X-axis1, Y-axis1, Area2, X-axis2 and Y-axis2;
```

Figure 6.13. Algorithm for landmark detection

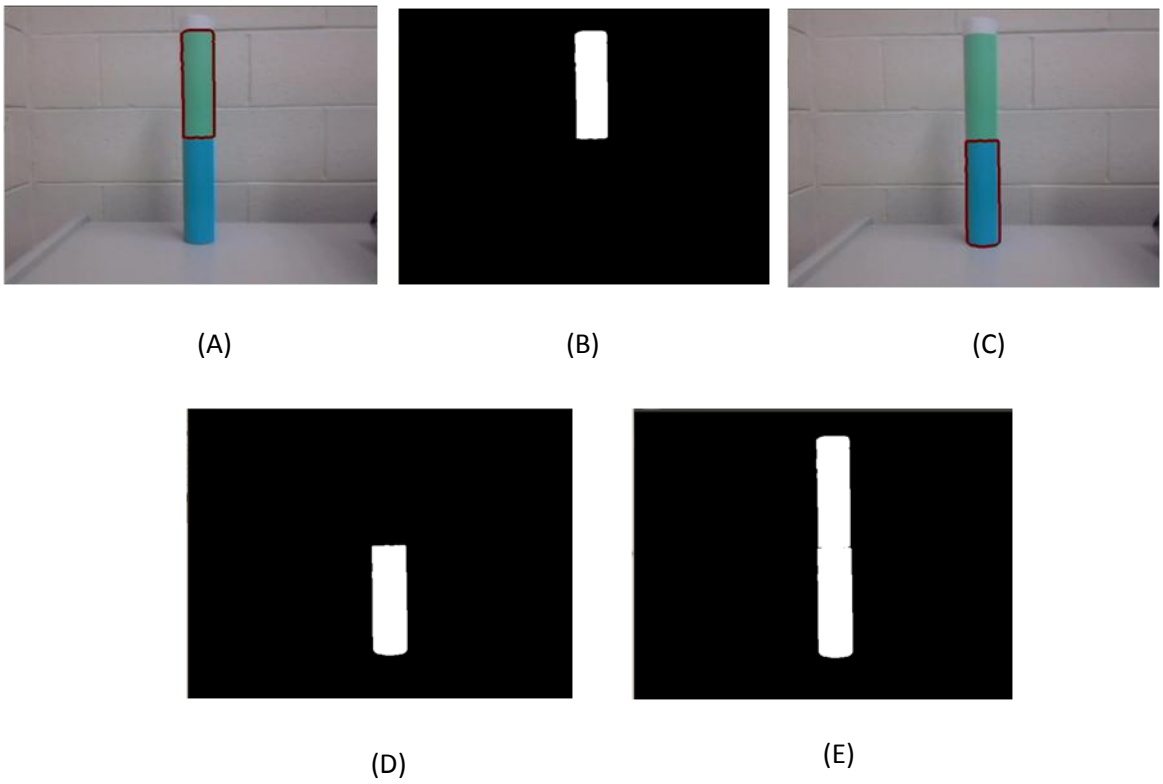


Figure 6.14. Landmark feature pattern and its segmentation

6.5 Experimental Results

The proposed algorithms for image processing were implemented using C++ and some libraries were used, including OpenCV [81] and OpenSURF [82]. They were initially tested by using a PC with a 1.7 GHz microprocessor and 1 GB of RAM. A Linux operating system (Ubuntu 10.04 LTS [80]) was installed in the computer. A webcam, model C200 was used as the vision sensor. This takes images with a maximum resolution of 640 by 480 pixels with an image capture rate of 30 frames per second. The camera was connected to the PC via a USB port.

The experiments were done off-line and then in real-time. In the former, environmental images were taken and processed to extract the object's features. In this scenario, it was possible to control the environmental factors, such as light intensity and background colours. Therefore, optimal results were obtained in these experiments. However, search robots execute their tasks in dynamic environments; therefore, the object detection algorithms must be evaluated under these conditions. Thus, the object's features were extracted from the images captured in real-time.

In order to evaluate the presented algorithms, a Coca Cola can, which is 6 cm in radius and 13 cm in height, was used and it was located at 60 cm from the camera (see Figure 6.15A). First, the colour segmentation techniques were tested to determine the processing time, and the can's area and its horizontal location in the image as shown in Figures 6.15B to D. Table 6.1 illustrates the experimental results. Then, the template matching and SURF methods were evaluated (see Table 6.2). It is important to note that the time was determined after processing ten frames while the area, centre and match of the can were calculated for the last frame. From the experimental results, it was concluded that:

- In both RGB and HSI colour segmentation methods, some image noise might appear in the binary image background because of the segmentation process; therefore, this noise must be removed or reduced. This was done using some morphology operations (see section 6.3.1.3).
- In RGB colour space, two methods have been used. The first method used the original 24-bit image directly to segment the desired colour. The second split the

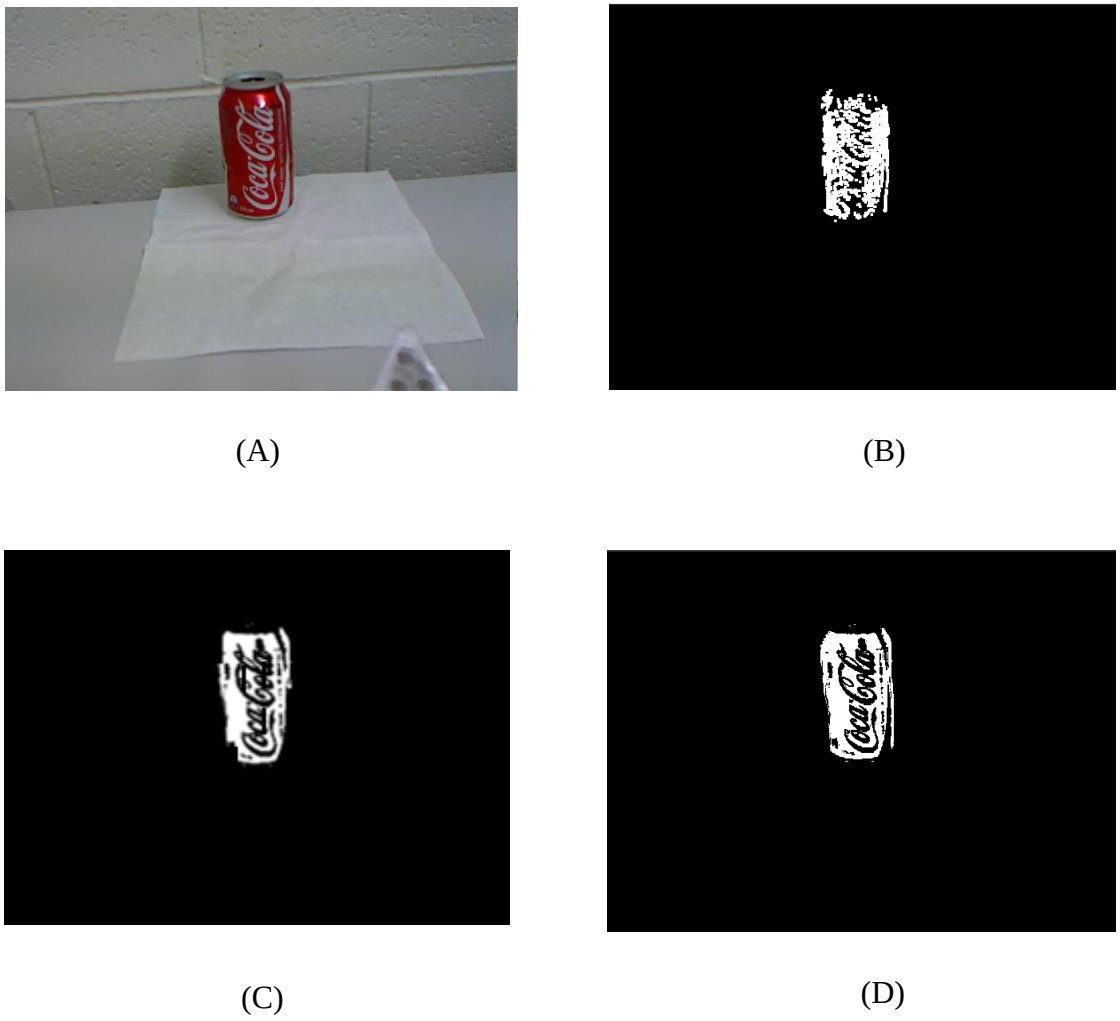
24-bit colour image into three 8-bit images; one for each colour intensity channel (see Section 6.3.1.1). The second method proved to be more efficient (the detected object is shown within more details in the image, see Figures 6.15B and C) because the segmentation process for each colour weight was not affected by the two other colour weights. Then, these channels were re-combined to achieve the binary image. Conversely, in the first method, each pixel in the image was tested for three weights, which strongly relates them to each other [41]. However, the number of calculations needed by the second method was greater than for the first and therefore, the processing time was slightly longer (see Table 6.1), and this affected the real-time processing.

- The HSI technique was more effective than the RGB colour space for finding an object by its colour (see Figure 6.15). This is because each uniform colour in HSI, from the darkest to the brightest, is assigned a particular period of Hue values, whereas the Saturation and Intensity periods specify only the amount and brightness of the colour, respectively. It was also concluded that the HSI technique was less sensitive to changes in light intensities in agreement with [44]. However, the segmentation process in HSI took longer than in RGB (see Table 6.1). This was because the output of the camera was in the RGB colour space and thus, the colour space transformation between RGB and HSI incurs computational cost, which affects real-time behaviour [41].
- When a target object with a single colour is used, optimal results are achieved by using the colour segmentation method. In this case, the segmentation techniques are ideal for detecting the target from any viewpoint because the object is specified by its uniform colour. The detection task depends on the number of pixels that follow the object pixels in the image and this is not affected by the object's rotation. However, the number of object's pixels is influenced by the object's distance from the camera. In this case, if the object is close to the camera, it will cover a large area in the image. Conversely, if it is far away from the camera, it appears in the image as only a few pixels.
- The background colours of the environment affected the segmentation process.
- The template matching and SURF methods achieved better results when multiple-coloured objects were used. In this case, an accurate matching process is attained because the templates that represent the objects' images have the highly detailed

and unique regions [49]. The SURF method and specifically the associated calculation process, was quicker because only one object template was used, instead of the 16 used in template matching (see Table 6.2).

- In all the above-mentioned techniques, the coordinates at the centre of the target in the image must be calculated. This information is then supplied to the robot's controller, which influences its motion as will be explained in Chapter 7.

The image processing algorithms were then implemented on the real robots for validation.



(A) (B)

(C) (D)

Figure 6.15. The Coca Cola detection by the colour segmentation technique: (A) the original image; (B) and (C) the image segmentation using RGB (methods 1 and 2, respectively), and (D) the image segmentation using HSI colour space

Table 6.1: *Experimental results for the colour segmentation methods by using the PC. HSI was more efficient than RGB (the detected object's details in the image have a high quality); however, the processing time was longer. The numbers of pixels in object area were different because the morphological process levels (Number of iterations) and the threshold values that were used in the segmentation methods were different. Therefore, the centre horizontal coordinate (X axis) was slightly different; however, that was not affecting the controlling process (see Chapter 7).*

Item	RGB (method 1)	RGB (method 2)	HSI
Time (s)	1.641	1.653	1.901
Area (pixel)	6468	7527	8297
Centre, X axis (pixel)	343	340	341

Table 6.2: *Experimental results for template matching and SURF (the match was determined by the number of the matched points that exist between the template and the captured image in SURF method while it was calculated by the correlation number in the template matching) by using the PC. Note that the time is for processing ten frames while the robot only needs to process one frame; therefore, the processing time will be reduced to 0.4 and 3 seconds for the SURF and template matching, respectively*

Item	SURF	Template matching
Time (s)	4.244	30.171
Match (matched points for SURF and correlation for template matching)	8	0.785
Centre, horizontal coordinate, X axis (pixel)	339	337

6.5.1 Image Processing for the Hexapod

As mentioned in Chapter 3, the hexapod has a computer-based Roboard controller RB-100 that has a Vortex86DX and a 32 bit x 86 CPU running at 1 GHz with 256 MB on-board memory. The main control program and all image processing functions were implemented on-board. The main drawback was that this board has a processor with a

low computation capacity (see Tables 6.3 and 6.4). For instance, Table 6.3 shows that although the PC's processor used is 1.7 times faster than the Roboard processor, the image processing time within PC was approximately four times shorter than that determined within the hexapod. The experimental results also showed that this board sometimes failed in the segmentation using the HSI process and consequently, the robot stopped moving at the location where the failure occurred. The board was also extremely slow in processing the template matching and SURF calculations (see Table 6.4), which meant that the robot moved extremely slowly. Therefore, the colour segmentation methods were adapted for the hexapod to detect both the target and the delivery location.

Table 6.3: *Experimental results for the colour segmentation methods by using the hexapod's Roboard, the time was determined after processing ten frames while the area and centre of the can were calculated for the last frame (see Table 6.1)*

Item	RGB (method 1)	RGB (method 2)	HSI
Time (s)	7.043	7.105	7.315
Area (pixel)	6279	7595	7784
Centre, X axis (pixel)	335	332	333

Table 6.4: *Experimental results for template matching and SURF by using the hexapod's Roboard, the time was determined after processing two frames while the match and centre of the can were calculated for the last frame (see Table 6.2)*

Item	SURF	Template matching
Time (s)	8.311	93.171
Match (points for SURF and correlation for template matching)	6	0.711
Centre (X axis)	331	335

6.5.2 Image Processing for the Wheeled Robot

It was decided to use a PC motherboard with a high computational capacity to perform the image processing calculations. Therefore, a wheeled robot was implemented, as explained in Chapter 4. An IBM motherboard, which has Intel Pentium 4 CPU 3.00 GHz and 1.00GB of RAM was used. The experimental results showed that this board was much faster for image processing (see Tables 6.5 and 6.6).

Table 6.5: *The colour segmentation methods' results by using the wheeled robot's motherboard; the processing time for ten frames was much shorter than other two boards (see Tables 6.1 and 6.3 for comparison)*

Item	RGB (method 1)	RGB (method 2)	HSI
Time (s)	0.939	0.942	1.001
Area (pixel)	6796	7674	8303
Centre (X axis)	329	328	328

Table 6.6: *The template matching and SURF's results within the motherboard, the time was determined after processing ten frames. As mentioned above, the robot only needs to process one frame; therefore, the processing time has been dramatically reduced by using the new processor to be 0.2 and 1.5 seconds for the SURF and template matching, respectively*

Item	SURF	Template matching
Time (s)	2.458	15.906
Match (points for SURF and correlation for template matching)	9	0.795
Centre (X axis)	327	330

The experimental results demonstrated that the algorithms used for object detection were highly efficient providing adequate processing capacity (Intel Pentium 4 CPU 3.00

GHz and 1.00GB of RAM). Finally, for the hexapod, the colour segmentation methods were used to detect both the target and the delivery location, whereas for the wheeled robot they were only employed to find the delivery location.

6.6 Conclusion

Different image processing techniques have been employed in order to achieve optimal results for object detection. It has been concluded that when an object is specified by unicolour, segmentation techniques are ideal for detecting it from any direction. The detection process is not affected by scale and rotation of the object. However, it is affected by the environmental conditions, particularly the colours of the background. The template matching and SURF methods achieve better results when multiple colours are used. The SURF method has proved to be quicker than the template matching method. The image processing algorithms need processors with high computational performances. Therefore, when they were rebuilt in the hexapod robot processor, the robot proved to be extremely slow. Consequently, a new robot was implemented that uses a PC motherboard. In this study, the SURF and template matching methods are used to detect and recognise the target object and the colour segmentation method is employed to detect the landmark that defines the final location (delivery location). This is because the conditions of the environment near the delivery location can be controlled. The landmarks are cylindrical in shape and they have two colours to improve the detection process.

Chapter 7 The Control System

7.1 Introduction

In Chapter 1, the problem statement identified the challenges in constructing a navigation system for an indoor search robot. One of the main objectives of this work is to design a system that is capable of overcoming the challenges using a small number of sensors. The previous chapters have described the main tools that the control system needs to enable the robot to search its environment for a target object. The proposed control system enables the mobile robot to collect environmental information and transfer it to the navigation algorithm in real-time; generate an exploration path that the robot can navigate without collisions; and control the robot throughout its motion. The robot's navigation strategy was introduced and explained in Chapter 5 and this enables the robot to explore the entire search area of an indoor environment. As the robot executes the visual search, it needs tools that enable it to perform this task. In Chapter 6, some of the object detection techniques were presented. This chapter explains the implementation of the entire control system for both the robots used in this study (the hexapod and wheeled mobile robots, see Chapters 3 and 4).

7.2 The Main Program of the Hexapod

The navigation system relies on the code that controls the robot's motion. The system gathers sensor information in real-time and translates it into control commands. In the proposed navigation system, the robot starts its motion by searching the environment in the start point with the objective of finding the target object. If the robot does not find the target, it will navigate and explore its environment during its motion. Figure 7.1 depicts the mobile robot navigation strategy in which the robot keeps the wall on its left (as explained in Chapter 5) when moving forward. Any obstacle that exists in the robot's path will be considered as a structure that is similar to the wall and the robot turns right and navigates along this obstacle. While the robot navigates, it continues searching for the target and correcting its location beside the wall.

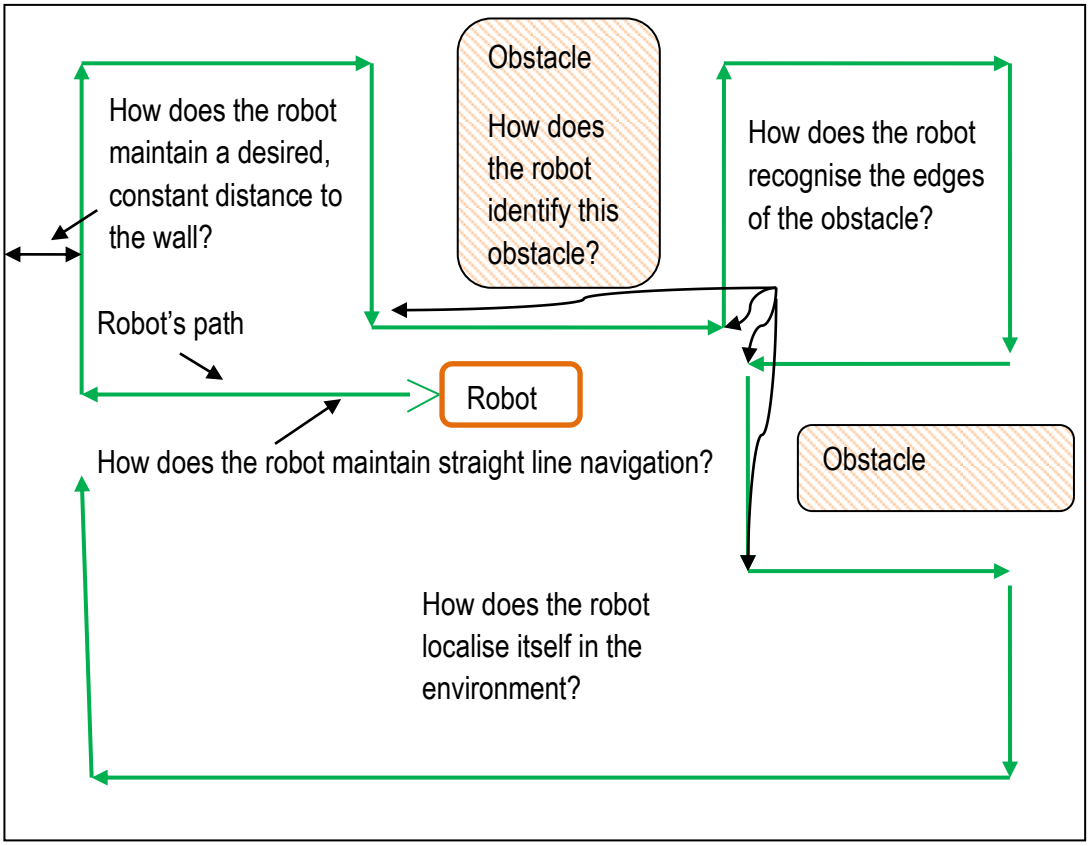


Figure 7.1. The robot navigation strategy

Figure 7.2 represents the flowchart segments of the navigation process of the hexapod mobile robot. First, the robot executes searching mode (1), which enables the robot to search 360° in the current field (see section 7.2.2). If the target object is found in this process, then time and effort included in the searching process will be saved. However, if the robot does not find the object, it will move to the nearest wall (or similar obstacle). The nearest wall can be found either by the vision system (as such, it is defined by a specific artificial landmark), or by the ultrasonic range sensors. When the robot reaches the pre-defined distance from the wall (which is defined by the user and it must enable the robot to rotate without colliding with the walls or obstacles), it will turn 90º to the right and increase the termination counter (counter (1)), which helps the robot decide when the searching process is completed (i.e., when it has covered the whole searching area). The robot will compare the current counter value with the pre-defined value that has been decided by the user. If it is equal to or greater than the pre-defined value, the robot will terminate the navigation and the search process. Otherwise, the robot stops moving and performs the searching mode (4) function that enables the robot

to search for the target object in the current direction. If the robot finds the target at any time, it will leave the wall and finish the searching process. Otherwise, the robot will position itself beside a wall and continue the search.

The locating process is initiated by reading the front ultrasonic range sensors' information. If the distance between the robot and the wall or the obstacle is equal to or less than the pre-defined value, this means there is not enough free space for the robot to navigate and correct its location beside the wall. In this case, the robot stops moving and executes searching mode (3), which enables the robot to find that either the encountered obstacle is the starting point or not (see section 7.2.2). If so, the robot terminates its motion; otherwise, it turns 90º to the right. If there is enough space to correct the location, the robot will read the information from both left-side range sensors. If both measurements are equal to or less than the maximum pre-defined value of the distance interval (which specifies the largest distance from the adjacent wall and it also specified by the user as explained in Chapter 5), the robot will locate itself beside the wall (see section 7.2.1) and then move to a specific distance before resuming the search process. The hexapod robot walks six steps (30 cm) and then stops to update the sensors' information. A counter (2) is added to the program to specify the distance that the robot walks without searching for the target object (performing the searching mode (2)) as will be explained in section 7.3.

However, if both left-side range sensor measurements are greater than the maximum pre-defined distance value, this will mean that the robot moves beside the obstacle, having reached the boundaries of this obstacle. In this case, the robot turns 90º to the left in order to keep the obstacle on its left side. Then, it executes searching mode (5), which enables the robot to finds out that either it encounters entrance (door) or not (see section 7.2.2). If so, the robot turns 90º to the right and moves beside that entrance. Otherwise, it keeps walking and reading the sensor measurements until it detects the boundaries of the obstacle again. This enables the robot to navigate around the obstacle. In this process, the robot keeps searching the environment for the target object as explained before. However, if one of the left-side range sensor measurements is greater than the maximum pre-defined distance value, the robot seeks to detect the boundaries of the obstacle and therefore, it walks a specific distance and then checks the sensors' information. The robot navigation process continues until either the robot finds the

target, or the counter value reaches the pre-defined termination value. The robot also terminates the searching process if the start point is re-encountered. If the robot finds the target object, it will move to approach it, then grasp and relocate it to a new location (see sections 7.2.3 and 7.2.4).

7.2.1 Locating the Robot beside the Wall

When the robot follows the walls or obstacles boundaries, it must move within a desired distance from these objects to avoid collision. The flowchart in Figure 7.3 explains the implementation of the code that uses the side range sensors' information to control the hexapod mobile robot motion and to locate it beside the wall. When the robot wants to walk along the wall, it starts ranging the distance between its current location and the wall. If one or both of the range sensors' measurements is greater than the maximum pre-defined distance interval (see section 5.5), the correction loop is ended. Otherwise, the loop is continued by comparing both measurements. If both measurements are not equal, the robot will be either moving away from the wall or risking collision with the wall. Therefore, the robot needs to correct its direction to be parallel to the wall. For instance, if the side front sensor measurement is greater than the rear side sensor measurement, the robot will correct its direction by rotating a specific angle to the left. This process will be continued until the minimum difference between the two sensors' measurements, which is specified by the user to enable the robot to move within a minimum error, is reached. This process helps the robot to move along the wall in a straight line.

The robot in this loop also ensures that its distance from the wall is equal to the pre-defined interval values. If the distance is equivalent to the pre-defined values, the correction loop is ended. Otherwise, the robot needs to move away from or closer to the wall. For instance, if the distance is greater than the maximum pre-defined distance interval, the robot will correct this and move to be closer to the wall. This process is repeated until it achieves the desired condition. This process keeps the robot moving along the wall at a specific distance without any collisions with that wall. While the robot performs this process, it keeps checking the front ultrasonic range sensors' information. If the measured distance is equal to or less than pre-defined value, the robot terminates this process.

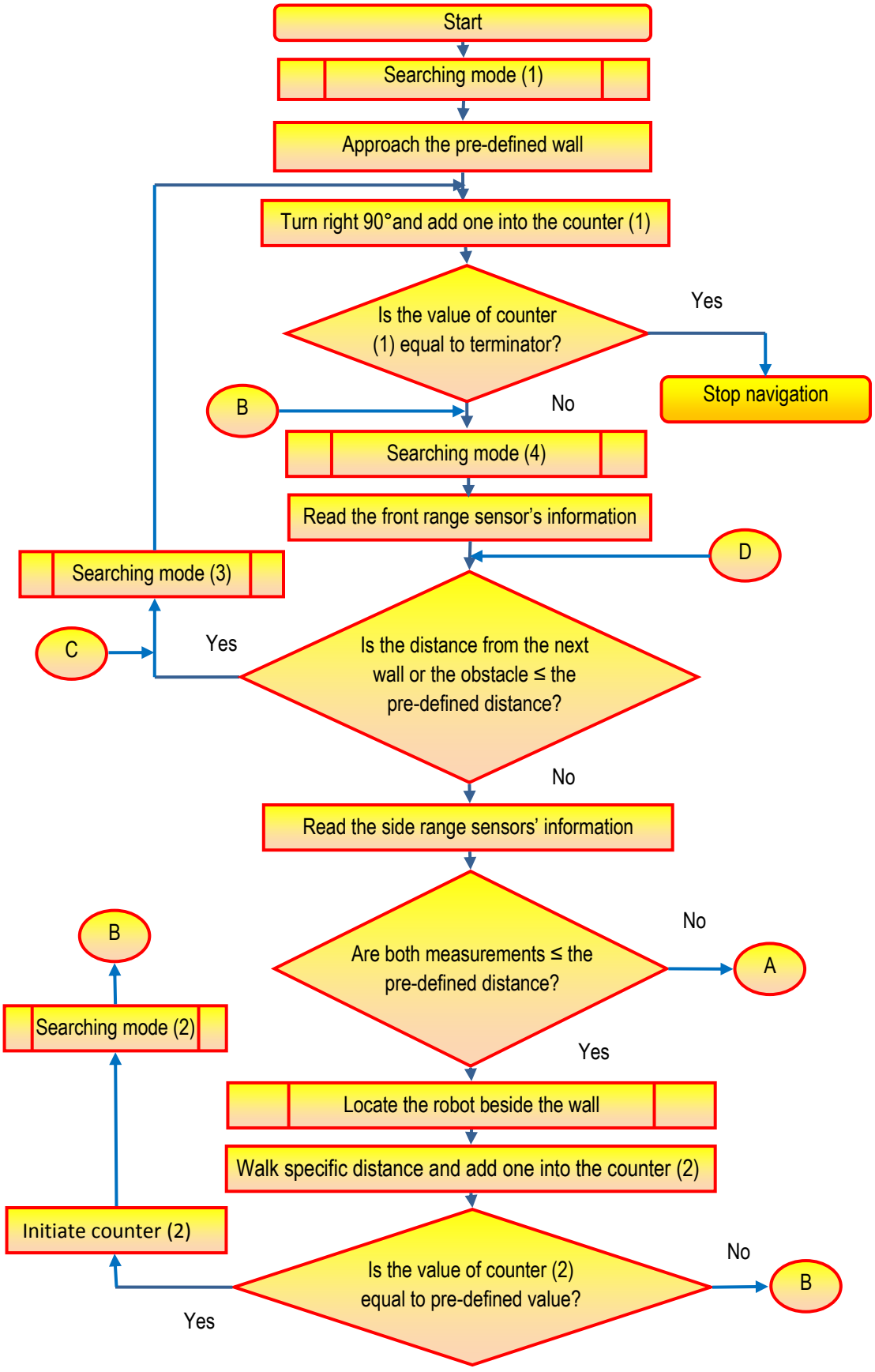


Figure 7.2. The navigation process flowchart

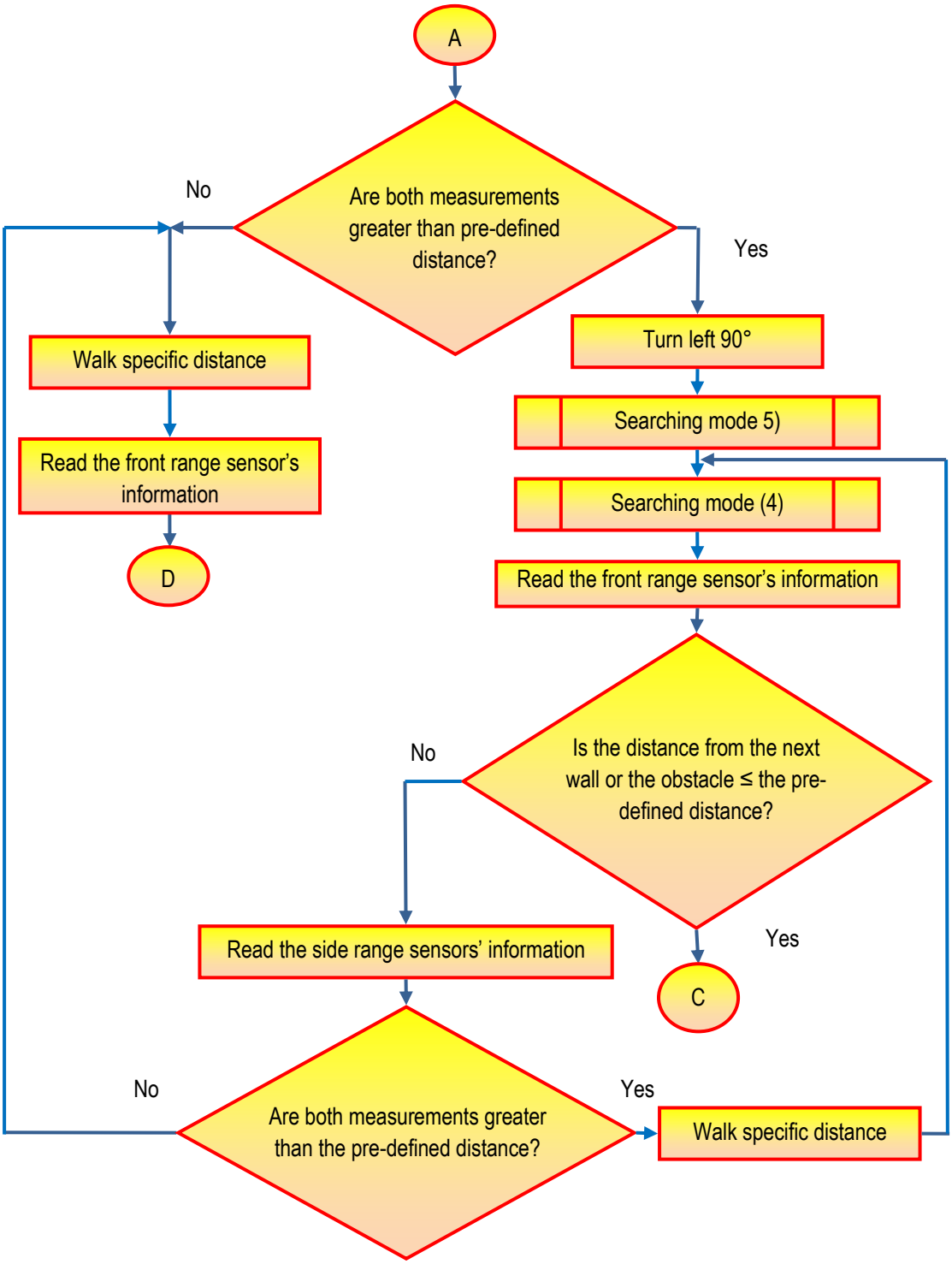


Figure 7.2. The navigation process flowchart (continued)

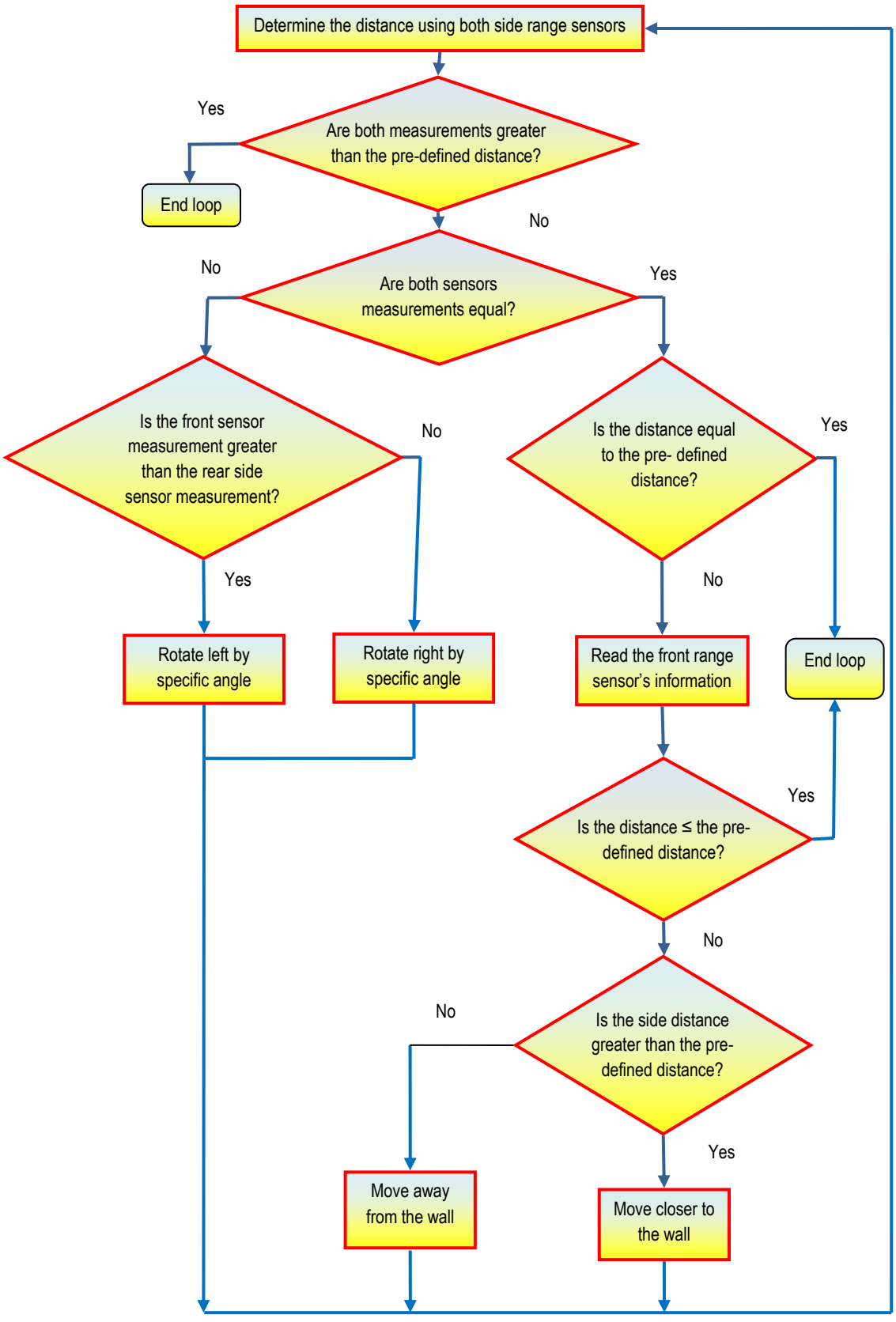


Figure 7.3. Flowchart for locating the robot beside the wall

7.2.2 Searching Mode

Searching mode means that the robot uses its vision system to find a target that is a known object in an unknown 3-D environment; this task requires the robot to control the camera's action. As mentioned in Chapter 2, this process includes two stages: 'where to look next' and 'where to move next' [69]. In this study, both robots are equipped with cameras that do not have zoom capabilities. Therefore, the target's size in the image depends on its distance from the robots. In the first stage, the robot stops its movement and fixes the camera in a current position (viewpoint) for a length of time sufficient for acquiring and processing the image and updating the robot's knowledge about the environment. If the target object is not detected in the current viewpoint, the robot executes the second stage in which it moves the camera to the next optimal viewpoint. The next viewpoint should be reachable with high probability of detecting object. It should also bring other hidden search areas into the camera's view [69].

The selection of the next viewpoint includes choosing the position and direction of the camera (the new sight angle) relative to the previous viewpoint of the camera. The view angle of the vision sensor plays an important role in the search process because it decides the area that will be searched each time. It also decides the maximum pan angle that must be used to rotate the camera each time. For instance, the blue object that is shown in Figure 7.4A is not detected and recognized by the vision system because portions of the object area are hidden from the camera's view. In this case, if the camera is rotated by an angle that is the same as the sensor's sight angle, the same problem will continue to appear, as shown in Figure 7.4B. Moreover, if the pan angle is greater than the sensor's vision angle, even the object that is on the area between the new field (cyan area) and the previous one (yellow area) will be ignored, as shown in Figure 7.4C. Therefore, in order to solve this problem the camera is rotated by pan angles that are less than the sensor's view angle, as shown in Figure 7.4D.

The probability of finding the object in the field increases if the robot searches each single area from two different viewpoints. This is because the camera might be rotated to a viewpoint in which the camera acquires the object's image with higher contrast features. In this research, the sight angle of the vision sensor used was 36°. The decision taken was to use a 20° pan angle, which it is almost equal to half of the vision angle and

satisfies all the above conditions. The hexapod mobile robot has a fixed camera; therefore, the pan rotation is provided by revolving the robot about its central axis. As such, the searching time is increased by the duration of the robot's rotation. This problem has been solved for the new, wheeled robot by attaching the camera to a servo motor that turns the camera by a specific pan angle.

The hexapod robot has five searching modes (see Figure 7.5). First, searching mode (1) is used to search for the object in the starting location. If the target is not detected in the first image taken by camera, the robot turns 20° clockwise (in case of wheeled robot, the camera is rotated) about its central axis and another image is taken. This process is repeated until the target is detected. After a full rotation of 360°, if the target has not been detected, this loop is terminated and the robot starts the navigation process.

Second, the searching mode (2) is used when the robot is navigating, i.e., it locates the wall at one of its sides; therefore, the robot needs to search the space that is directly ahead and also the space that is inside the environment (the middle of the room). In this case, the robot rotates the camera 180° instead of the 360° that is used in mode (1). As mentioned above, the hexapod mobile robot had a fixed camera so that it performed searching modes by rotating itself about its central axis. Therefore, in both modes the robot has been programmed to execute the object approaching function (by moving straight forward in the current direction) once the target is found, or to continue the search process. In case of the wheeled robot, the camera was attached on a servo motor to rotate it. Therefore, if the target is found in any viewpoint, the robot will revolve its body in that direction and then perform the object approaching function.

Third, the searching mode (3) is employed to terminate the robot motion if the searching process is completed without finding the target. In this case, the robot finds out that either the encountered obstacle is the start location or not? If so, the robot terminates its motion; otherwise, it continues the navigation process. The searching mode (4) is used to search the area, which is directly ahead, for the target object while the robot moves forward. Finally, the searching mode (5) is employed within the navigation system in order to enable the robot to finds out that either it encounters the entrance of the environment or not. If so, the robot steers 90° to the right and moves beside that entrance.

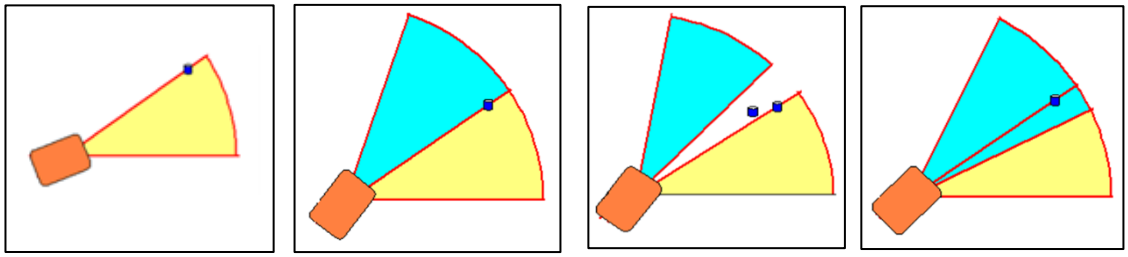


Figure 7.4. Viewpoints of the camera: (A) the state when the blue object is on the boundary of the camera's view area (part of the object cannot be seen by the robot); (B) the camera is rotated to the new viewpoint (cyan area) by an angle that is the same as the camera's old viewpoint (yellow area); (C) the state when the camera is rotated by a pan angle is greater than the camera's sight angle; and (D) shows the desired state when the pan angle is less than the camera's sight angle

```
Name: searching mode (1)
Input: Image processing results
Output: Camera rotation, Robot motion
    for (angle = 0 to 360° step 20°) do
        Image processing;
        if (target object found) then
            go to the object approaching;
        end if
    end for
    go to navigation process
```

```
Name: searching mode (2)
Input: Image processing results
Output: Camera rotation, Robot motion
    for (angle = 0 to 180° step 20°) do
        Image processing;
        if (target object found) then
            go to the object approaching;
        end if
    end for
    go to navigation process
```

```
Name: searching mode (3)
Input: Image processing results
Output: Robot motion
    Image processing;
    if (start location encountered) then
        terminate the robot motion;
    end if
```

```
Name: searching mode (4)
Input: Image processing results
Output: Robot motion
    Image processing;
    if (target object found) then
        go to the object approaching;
    end if
```

```
Name: searching mode (5)
Input: Image processing results
Output: Robot motion
    Image processing;
    if (environment's entrance's landmarks encountered) then
        turn right 90°;
    end if
```

Figure 7.5. The pseudocodes of the searching mode algorithms

7.2.3 Approaching the Object

When the target object is detected, the main task for the robot is to find out the target's location and then execute a specific duty [98]. In this study, the robot must grasp and relocate the target to the delivery location (starting point). In this scenario, the robot must approach the object by keeping its image within the centre of the image plane [44]. As such, the object's centre (horizontal coordinate) in the image is determined, as explained in Chapter 6, and forwarded as input to the robot controller. In this process, the captured image is divided into three areas (see Figure 7.6); consequently, there are three logical cases that can be fed to the controller. The first is that the object's centre is on the right of the middle area (green area); therefore, the robot has to rotate to the right using a specific rotation angle, dependent on the amount of the error value, in order to correct the error and bring the object's centre to the middle area. The second is that, if the centre is on the left side, the robot will rotate to the left side and the third, if the centre is in the middle, the robot moves forward to approach the object.

During the approach process, the robot controller continuously updates the position of the robot relative to the object by using the information coming from the front ultrasonic sensors. When the robot reaches a pre-determined distance from the object, which enables the robot to grasp it, the robot does this. In the grasping action, the robot opens its gripper, moves it down to place the object in an appropriate grip point (in the middle of the object), closes the gripper to grasp the object firmly, and then moves it up a desired distance. As mentioned in Chapter 5, a force sensor was attached in the gripper to detect the amount of the grip force that is applied to the object. This sensor is employed as on/off switch. If the force exceeds the pre-defined maximum value, which depends on the grasped object's weight, the robot will stop the gripper servo motors' shafts in the current locations. Once the grasping operation is accomplished, the robot then executes the relocating process (as will be explained in the next section). If the robot loses the object in the image during the approaching process, it will again carry out the searching mode (1). Figure 7.7 presents the pseudocode of the object approaching process.

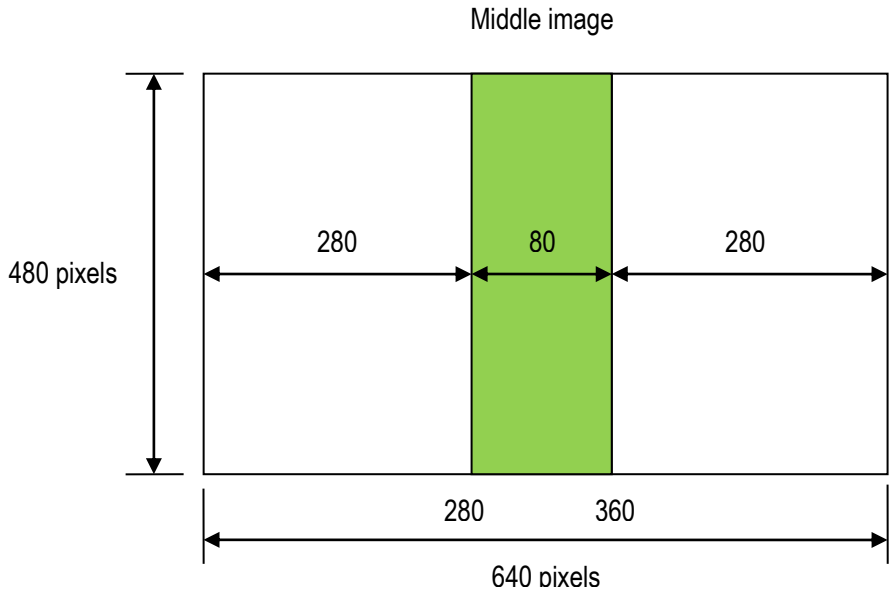


Figure 7.6. The desired area (green area) location in the image

Name: Object approaching
Input: Image processing results, Ultrasonic Range sensors' measurements ()
Output: The robot motion.

```
if (the object is not approached) do
    Image processing results;
  if (Target object found ) then
      Determine the object's central coordinates (X-axis, Y-axis);
      if (the object's centre is in the left of the image plane) then
          Turns left;
      else if (the object's centre is in the right of the image plane) then
          Turns right;
      else if (the object's centre is in the middle of the image plane) then
          Read the front ultrasonic range sensors' measurements;
          if (distance, measured from the robot to the object ≤ the pre- defined distance) then
              grasp the object;
              go to the relocating process;
          else
              Move forward (approaching the object);
          end if
      end if
  else
      go to the searching mode (1);
  end if
end if
```

Figure 7.7. The object approaching algorithm

7.2.4 Relocating Process

Once the target has been approached and grasped, the robot starts searching for the delivery location that is defined by a landmark (see Chapter 6). As such, the robot performs the same codes that were used initially to find the target object, but this time it searches for the delivery location. At the location where the object is grasped, the robot searches the environment for the delivery location by performing the searching mode (1). If the location is found, the robot will approach it and deliver the object. In this case, the robot carries out the approaching function (see section 7.2.3) except that the image processing is performed to find the location. When the robot reaches the delivery location, it moves its gripper to a point above the delivery location surface. Then, the gripper is opened and moved up to release the object on the surface. However, if the location is not detected in the previous process, the robot will carry out the navigation and searching functions as explained in section (7.2).

7.3 Experimental Result

The experiments have been conducted on the hexapod. The main control program and all image processing algorithms were written using C++. The experimental environment was the office room with a total area of 3200 × 4000 mm. Figure 7.8 represents the environmental map of the room. The brown rectangle represents the robot and the green rectangle denotes the starting point (delivery location) to which the robot must relocate the target object. The letter scripts (in Figures 7.8A and B) identify the locations of the robot in Figures 7.9 and 7.10, respectively, and the green arrows represent the robot's navigation path. The experiments were carried out to evaluate the hexapod's control system. First, the system was tested without the robot finding the target object. The objective of the robot was to navigate along the walls, starting from the starting point and to return to this location, avoiding collisions with obstacles in its path. The obstacles were rectangular shaped objects of various sizes (see Figure 7.8A).

In the initial location, the robot started searching for the target object by executing searching mode 1. When the object was not found, the robot started moving from its current location and approached the nearest wall (see Figure 7.9A). After approaching the wall, the robot turned in the desired direction (for example, turned right). When the

robot reached the first obstacle, it viewed this obstacle as a wall, turned right and navigated along it. The robot continued moving until it reached the boundaries of the obstacle. In this case, the robot turned left, in order to keep the obstacle on its left side. The robot kept looking for the object while it moved forward by performing searching modes 2 and 4. These processes were repeated for other obstacles until the robot went back to the wall (see Figures 7.9B and C). The robot continued moving along the walls or obstacles until it encountered the start location (delivery location), where the robot terminated its motion (see Figure 7.9D).

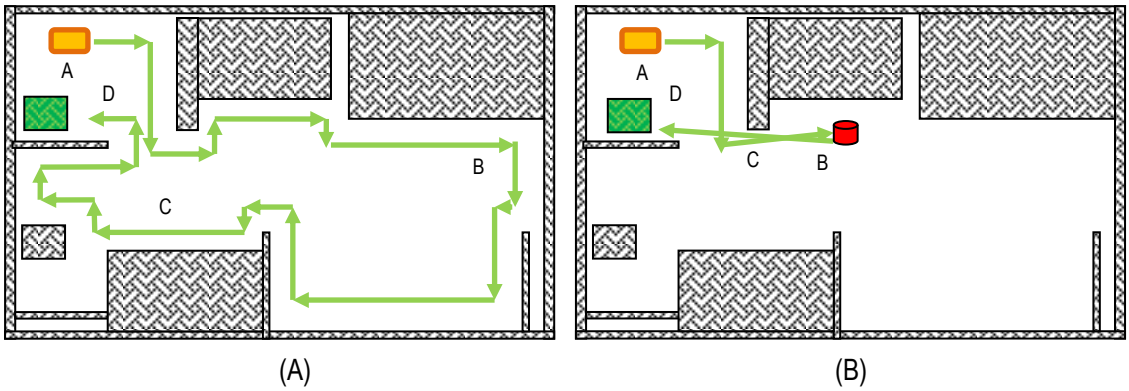


Figure 7.8. Simulation of the environment

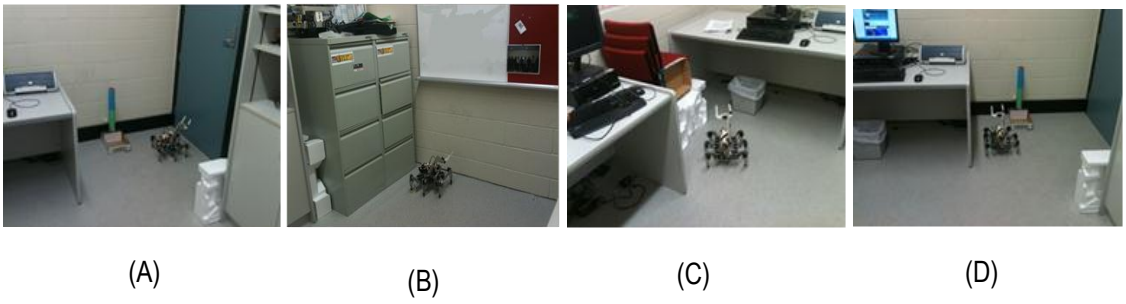


Figure 7.9. The hexapod motion (without finding the target object)

Two heuristics, which are characterised by selecting the points that the robot must stop and search its surrounding environment [30], were used to examine the relative effect of varying search and travel cost (the operating time needed). In the first heuristic, the robot travels 1.2 m, which is specified by counter (2), and then stops to execute searching mode (2) to search the environment for the target. In the second heuristic, the robot travels 1.5 m before performing searching mode (2). Table 7.1 describes the results of both searching heuristics.

The hexapod walked six steps (30 cm) forward and then stopped to update the sensory information, which consists of the ultrasonic sensors and camera information (performing searching mode 4). In this study, the robot required 12.8 s for travelling (30 cm) and 2.9 s for updating the environmental information. In the first heuristic, the robot repeated this process four times to travel 1.2 m so it required 62.8 s (51.2 s travelling and 11.6 s updating sensors' information) to perform this task. Then, the robot stopped to execute searching mode 2, which required 84 s (26.1 s updating camera information and 57.9 s rotating the camera) to be accomplished. The robot navigated in the previous environment (Figure 7.8A) 10.8 m to complete searching the whole area. In this scenario, the search process consumed 67% of the operating time. This involves the time of rotating the robot's body (pan the camera) (779 s) and image processing (365.4 s). The travelling time represent of 33% of the total time and it only consists of moving forward and updating the sensory information. In the second heuristic, the robot travelled 1.5 m without performing searching mode (4); as such it only required 64 s. Then, it carried out searching mode (2). Although the travelling time was staying at 460.8 s, it increased to 44% of the operating time. This was because the robot travelled the same distance; however, it consumed less time within the search process.

Table 7.1: *Experimental results of two searching heuristics: the hexapod robot spent 67% or less from the operating time for searching the environment*

Item	Searching time (s)	Travelling time (s)	Total time (s)	Searching cost (%)	Travelling cost (%)
Heuristic 1	944.40	460.80	1405.20	67	33
Heuristic 2	588.00	460.80	1048.80	56	44

In both heuristics, the robot was extremely slow because it had a fixed camera so that it performed searching modes by rotating itself about its central axis. This contributed with the image processing and calculations of the robot's gait signals for making the navigation process extremely slow. The travelling time will dramatically increase if the robot travels in an environment that has more obstacles. The experimental results also demonstrated that with the decrease of the number of the heuristic's points that the robot stops and searches the surrounding area, the searching cost was reduced from 67% to

56% (see Table 7.1). However, this might reduce the probability of finding the target object. Therefore, the search process must be carefully planned, which means the heuristic should have a smallest number of points that can cover the entire environment.

Then, the control system was tested by the robot finding and relocating the desired object (see Figure 7.8B). In this scenario, the robot executed the above process until it detected and recognised the target object. In this case, the robot abandoned the wall and moved towards the target. In doing so, it kept the image of the target within the image plane. When the robot reached the pre-determined distance from object, it grasped it and put it in the delivery location (see Figures 7.10A to D). A specific artificial landmark (as explained in previous sections) defined the delivery location. The system was executed successfully, which proves the effectiveness of the methodology used.

Figure 7.10. Searching for, finding and relocating the object using the hexapod

Figure 7.11 illustrates the hexapod motion in an environment where the target is not placed close to the wall (here the target cannot be seen from the starting point because it is covered by the obstacle Figure 7.11A). Therefore, the robot moved along the walls and continued executing the searching modes (as explained before) until it could find the object. In this case, the robot left the wall and approached the object to grasp it (Figure 7.11B). Then, the robot performed searching mode (1) in order to find the delivery location; and because this location could not be seen from the current location, the robot moved to the closest wall (in this case the obstacle). Then, it followed the obstacle's boundaries until it detected the delivery location (Figures 7.11C to E). The robot then approached this location to put the object there (Figure 7.11F). More environmental states are given in [1].

The hexapod readily analysed the range sensors' information. However, the main drawback is that the robot has a processor with low computational speed (Roboard). The experimental results show that this board failed during segmentation with the HSI process and consequently, the robot stopped moving in the location where the failure occurred. The board was also extremely slow with processing the template matching and SURF calculations, which resulted in the robot moving extremely slowly. Therefore, the colour segmentation with the RGB was employed for the object detection, and this greatly restricted the robot's applications in respect of the environmental conditions that affect the image segmentation process (see Chapter 6).

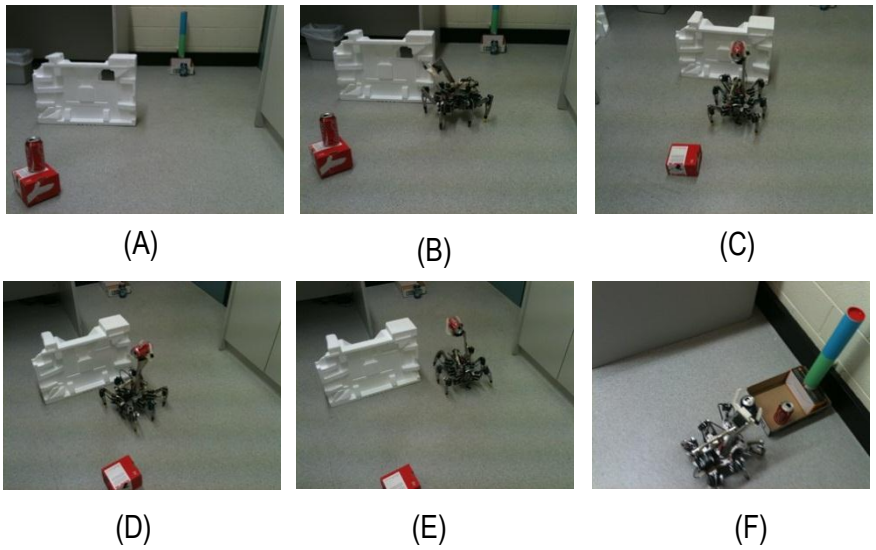


Figure 7.11. The hexapod motion when the object is not close to the wall

7.3.1 Problems and Limitations

The hexapod used had inherent hardware problems and limitations that affected its performance.

- The board did not have sufficient computation capacity for image processing calculations. Therefore, the experiments showed that image segmentation using the HSI failed and the robot terminated its motion in the location where the failure occurred. The robot's controller board also had computational speed limitations; therefore, it processed the template matching and SURF calculations slowly, which caused the robot to move slowly.

- It had a fixed camera so that it performed searching modes by rotating itself about its central axis. This contributed to making the navigation process extremely slow.
- It is small and therefore, the ultrasonic range sensors used must have narrow beams. Otherwise, they will detect some of the robot's components, such as its legs, as explained in Chapter 5. This will affect the method that enables the robot to follow the obstacle boundaries.

7.4 The Main Program of the Wheeled Robot

The decision was taken to test the methodology on a robot with a processor with higher computational capacity; therefore, the wheeled robot was designed and implemented (see Chapter 4). The wheeled robot was designed to have two boards. The first was the PC motherboard, which had an Intel Pentium 4 CPU 3.00 GHz processor and 1.00 GB of RAM to execute the image processing calculations. The second was the microcontroller for processing the sensory signals and performing the movement decisions. As such, the robot could capture images and process them while it navigated. In the final design, the robot's microcontroller sent the digital signals in two bits to the motherboard to process the images (Table 7.2); then, two signals were received about the states of the object in the image (Table 7.3). For instance, if the robot grasps the detected object and wants to relocate it, the microcontroller sends digital code (10) to the motherboard to initiate the search for the delivery location. In this example, if the robot does not find the location, then the motherboard sends digital code (00) to the microcontroller, which means that there is no object in the image.

Table 7.2: *The states of the searching process sent to the motherboard*

Bin (2)	Bin (1)	The search state
0	0	Stop searching
0	1	Search for the target object
1	0	Search for the delivery location
1	1	Search for environment's entrance

Table 7.3: *The states of object location in the images sent to the micro-controller*

Bin (2)	Bin (1)	The object state
0	0	**No object**
0	1	**In the Left**
1	0	**In the Right**
1	**1**	**In the middle**

When the microcontroller received the image processing results, it accordingly decided the robot's motion. The robot could update the environmental information in each of its movement cycles without stopping its motion. Therefore, the robot's reaction for any change in the environment was much faster than the hexapod. The wheeled robot was also equipped with ultrasonic range sensors that had wide beam widths and were more precise than those used with the hexapod. The wheeled robot was programmed to perform the same navigation algorithm that was used with the hexapod but it differed slightly in the way that it followed the walls. For example, the hexapod walked along the wall that was on the left, while the wheeled robot located the wall to its right.

The main motion control program was programmed in C and the image processing algorithms and parallel port's control programs were developed using C++. The experimental environment was the same as is in Figure 7.8. However, some obstacles were positioned at different locations, as shown in Figure 7.12. The experiments were performed in two stages. In the first, the robot accomplished the search process without the necessity of finding the target object (Figures 7.12A and 7.13A-F). The robot carried out the same heuristics that was executed by the hexapod. In the first heuristic, the robot required 11.3 s for travelling 1.2 m and then it stopped to execute searching mode (2), which required 17.83 s (this time is for rotating the camera, for acquiring and processing the images, and for updating the robot's knowledge about the environment) to be accomplished. In the second heuristic, the robot needed 14 s in order to travel 1.5 m, and then it carried out searching mode 2. Table 7.4 shows the operating time to travel and explore the environment in which the robot also travelled 10.8 m to search the whole area. The results demonstrates that the operating time depends on the searching time (the time of rotating the camera and processing the images) because the

travelling time is constant in both heuristics used. As such, the heuristic must be carefully planned to have a minimum number of points that cover the entire environment.

Table 7.4: *Experimental results of two searching heuristics: the wheeled robot spent 64% or less from the operating time for searching the whole environment*

Item	Searching time (s)	Travelling time (s)	Total time (s)	Searching cost (%)	Travelling cost (%)
Heuristic 1	178.30	101.70	280.00	64	36
Heuristic 2	124.81	101.70	226.51	55	45

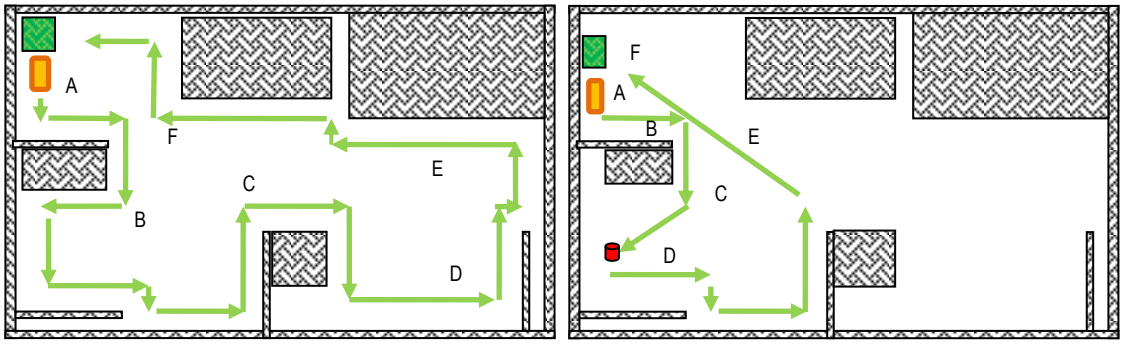


Figure 7.12. Simulation of the first environmental experiments of the wheeled robot

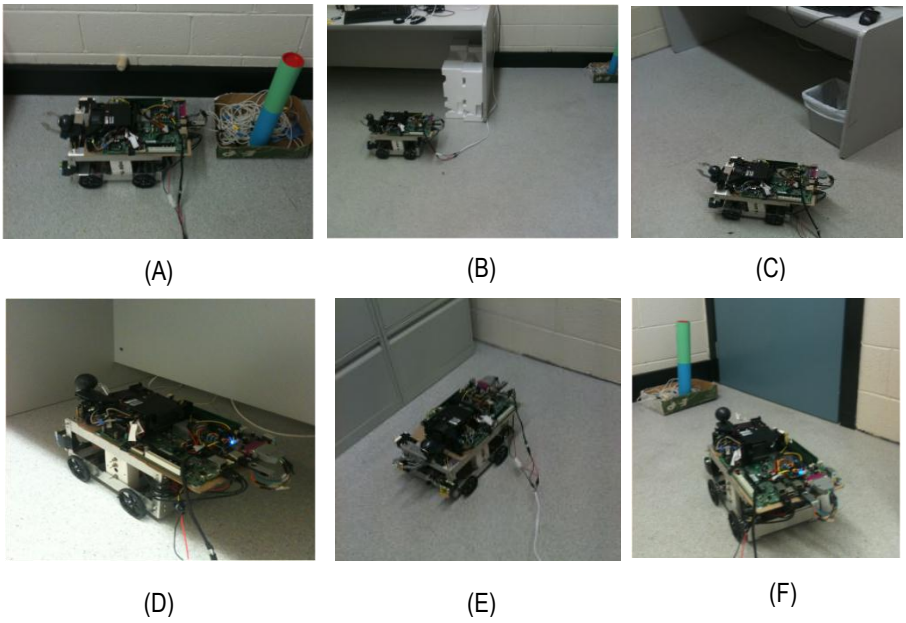


Figure 7.13. The wheeled robot motion (without detecting the target)

In the second stage, the robot found the object and relocated it to the delivery location, as shown in Figures 7.12B and 7.14A-F. The time needed to complete the search task was dramatically reduced with the wheeled robot. For instance, the hexapod needed about 25 minutes to search the entire area in the first heuristic described above, whereas the wheeled robot required less than 4.40 minutes to search the same area. The limitations of the hexapod are described in section 7.3.1.

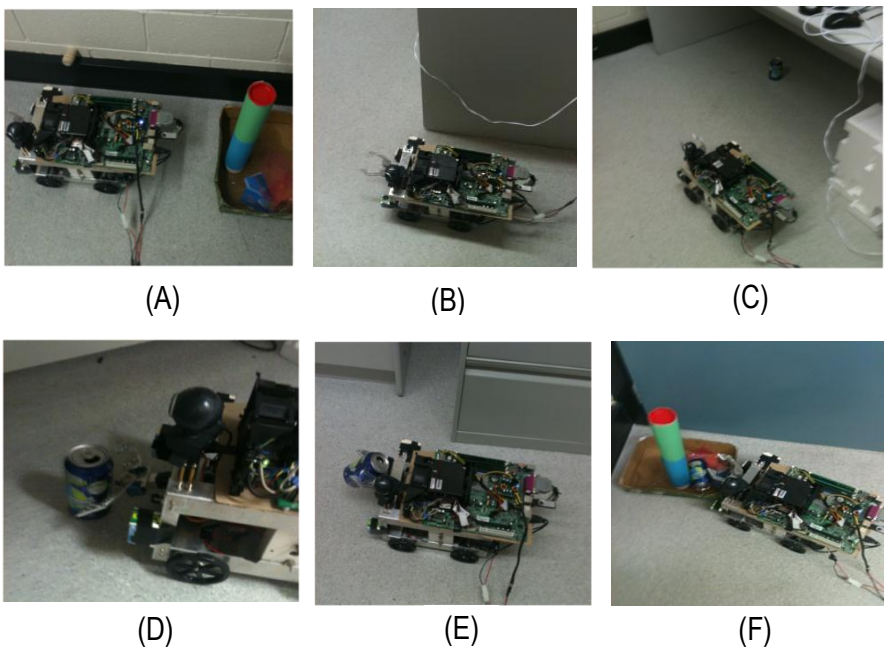


Figure 7.14. The wheeled robot motion (finding and relocating the target)

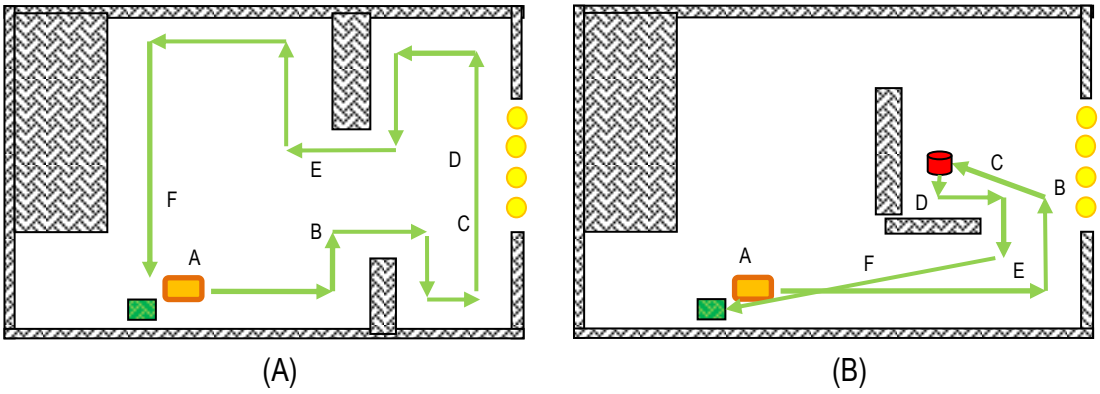


Figure 7.15. Simulation of the second environmental experiments of the wheeled robot

The experiments were then conducted in an environment that has a total area of 3200 × 3800 mm (see Figure 7.15) in order to evaluate the wheeled robot motion within different situations. The yellow circles represent the landmarks used to identify the

environment's entrance. The objective of the robot in Figures 7.15A and 7.16A-F was to navigate along the walls from the starting point and return to this point. When the robot reached and recognised the entrance's landmarks, it considered them as a wall and navigated along them, as shown in Figures 7.16C and D.

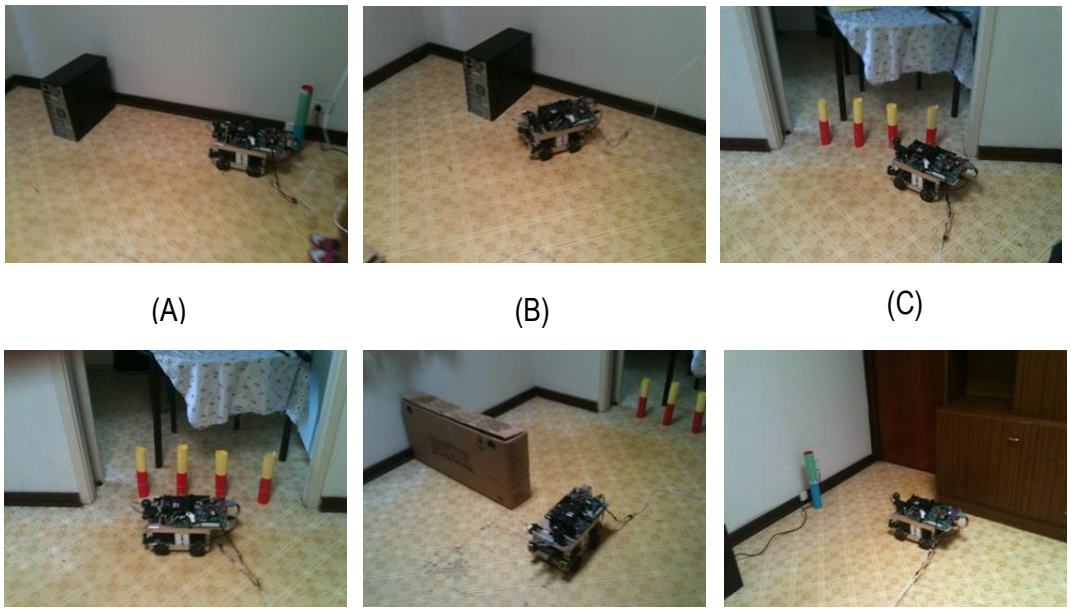


Figure 7.16. The wheeled robot motion in the second environment (without detecting the target)

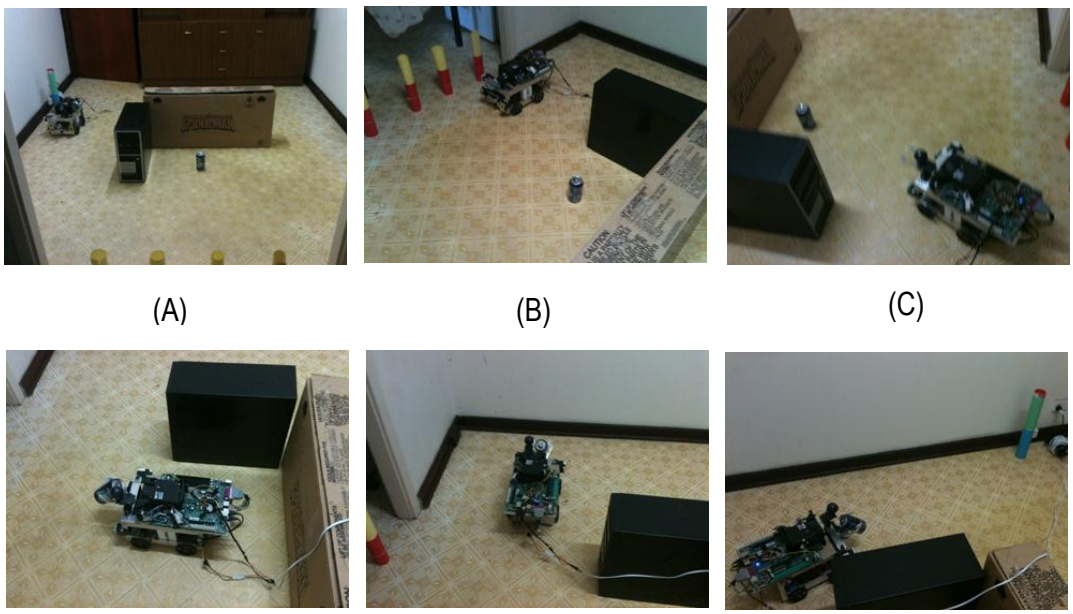


Figure 7.17. The wheeled robot motion in the second environment (finding and relocating the target)

Figures 7.15B and 7.17A-F explain the robot motion in the environment where the target is not placed close to the wall. In this scenario, the robot navigated beside the walls and continued performing the searching modes 2, 3, 4 and 5. When it discovered the object (Figure 7.17B), it left the wall and approached the object to grasp it (Figures 7.17C and D). The robot then moved to the closest wall (in this case the obstacle), and followed the obstacles' boundaries until it detected the delivery location (Figures 7.17E and F). In this case, the robot started approaching this location to place the object there. The system was also executed successfully, which proves the effectiveness of the methodology used.

7.5 Conclusions

In this chapter, the navigation system that enabled the mobile robot to search and find the object in the environment has been implemented and tested within two types of mobile robots, legged and wheeled. The robots' performances were greatly dependent on the electronic system. The wheeled robot was equipped with two processors. One was for executing vision task while the other was for the motion control. Therefore, its reaction for any change in the environment was much faster than the hexapod that used one board for both tasks. The hexapod had inherent hardware problems and limitations that also affected its performance. First, it belongs to the legged robots that are inherent to be slower than the wheeled ones. Second, it had a fixed camera so that it performed searching modes by rotating itself about its central axis. This contributed to making the navigation process extremely slow.

It was concluded that if the number of the heuristic's points that the robot must stop and search the surrounding area is reduced, the searching cost is dramatically decreased. However, this reduces the probability of finding the target object. Therefore, the search process must be carefully planned which means the heuristic should have a minimum number of points that can cover the entire environment. It was also concluded that it is possible to implement a navigation system within a smallest number of sensors if they are positioned and used effectively on the robot's body. Experiments proved that the methodologies used and that the codes developed made the robots capable of performing their tasks of finding, approaching and relocating objects as planned.

Chapter 8 Conclusions and Future Work

8.1 Conclusions

This thesis focused on mobile robot's navigation system. More specifically, it considered the challenges of designing and operating a mobile search robot used to search for, find and relocate a target object in an indoor environment. The problem was divided into three sub-problems: motion planning; visual-search; and relocation of objects. The first enables the robot to navigate in a planned path and to continue searching and discovering its environment. The second relies on using the vision sensor (camera) and is supported by some object-detection techniques. The third involves using a robot's gripper to grasp the desired object and then move it to the final location. The experiments were carried out with two types of robots, hexapod and wheeled robots, to demonstrate that:

- the methodology used is suitable for a search robot that works autonomously in unknown indoor environments;
- the way used is efficient to identify the location of orientation of robot for an effective control mechanism.
- the technique used is ideal to position the sensors on the robot's body for autonomous navigation.
- the techniques for image processing that are used are optimal for object detection within robot exploration applications;
- the proposed exploration path covers the entire environment for a search robot;
- the proposed strategy of searching is also suitable for the designed environment; and
- it is possible to construct an optimal motion control system that employs a camera and a minimum number of ultrasonic range and force sensors.

The conclusions of the proposed work are detailed as follows:

Mobile robot construction

Although designing and implementing of the mobile robot was not a closed problem, this work proposed a construction method of two different types of mobile robots. In Chapter 3, the modelling of the hexapod mobile robot was explained in detail. Then, the

control system for that robot was proposed. Chapter 4 explained the construction of the wheeled robot.

The exploration path

For the exploration path, the same Bug algorithm strategies were used, in which, the robot navigated by following the walls or obstacle boundaries. While the robot moved beside the wall, it had to execute three tasks. The first was to correct its location beside the walls and avoid the obstacles that were in its path. The second was to carry out the visual-search process for the target object, by rotating the camera to explore the surrounding environment. If the robot found the target at any time, it executed the third task, in which it approached the object, then grasped and relocated it to the target location. The target location was defined using the artificial landmark.

Object detection

Three techniques were employed for object detection and recognition; colour segmentation; template matching; and Speeded-Up Robust Features (SURF). There were two problems that appeared when implementing these techniques on the hexapod's controller board. First, the board could not support image-processing techniques that require high computation capacity. Second, the computation process was extremely slow, which made the robot slower than expected. These problems were solved by designing a new robot, which had a PC motherboard for the image processing.

8.2 Results

The main findings of these research questions can be summarized as follows:

- The problem statement identified the challenges in constructing a navigation system for an indoor search robot. Part of this research involved designing systems that were capable of overcoming the challenges. The general question was: How should a self-navigating mobile robot control system be designed?

Generally, the optimal navigation system must support three skills for the robot. The first is path-planning skill, which is the robot's ability to find the shortest way from starting point to the target position, while avoiding obstacles. All the existing navigation

algorithms assume that the robot knows the exact location of both places. However, in case of the search robot, it does not know the target position. As a result, the robot must navigate to search for the target. In this scenario, the robot has to explore the entire environment. The second skill is the ability to position itself in its environment. In this case, the robot must be trained to understand some of the environmental aspects. The third skill is the ability to control its motion, which relies on actuators or motors that move the robot from one place to another.

In the proposed navigation system, the robot starts its motion from the initial location (starting point) that is defined by a visual mark (landmark). The robot's objective is to search for, find, and relocate the target object to the delivery location, which is (in this study) the same as the starting point. To simplify the problem, the robot is used to search the indoor environment that is defined by external boundaries (walls). The robot navigates along the walls and considers all obstacles as structures similar to the wall. While the robot moves, it continues to orient itself beside the wall. It also keeps searching for the target and the starting point. If it encounters the starting point again, it will consider the search process accomplished and that the target is unreachable.

- Another aspect of the research was: How should the different types of sensors be integrated within the control system for the search robot?
- The accuracy of a measurement system will dramatically rise if the robot is equipped with the high number of sensors. However, this increases the robot's price and leads to a more complex control system in its implementation [7]. Therefore, the number of sensors must be reduced without affecting the effectiveness of the robot motion. The question is: How should a minimum number of sensors be attached on the robot's body for autonomous navigation?
- Which custom built instruments are needed for the robot's navigation system, which is based on the vision system and range sensors, to function optimally in the intended simulation of an industrial environment?

Various sensors can be used to enable the robot to sense its surrounding environment and then decide on its behaviour. These sensors provide the environmental information by means of electrical signals, which must be processed in the robot's processor to generate meaningful information for influencing the robot's motion. Some types of

sensors (such as visual sensors) need processors with a high computation capacity to analyse their signals. Typically, the robot is equipped with the appropriate types of sensors based on its application, and the robot's environmental situation. In this study, the robot executed a visual search; therefore, it was equipped with a vision sensor (camera). This sensor was used to perform three tasks: searching for the target object; finding the delivery location; and detecting the room's (indoor environment) entrance.

The robot needed to navigate along the walls and avoid the obstacles located in its path. In this scenario, the robot had to maintain a desired distance from the walls or the obstacles so it was equipped with ultrasonic range sensors. These were attached on the robot's body in a way that made each sensor perform a specific task. For example, two sensors were used for each side of the robot side to locate the robot beside the wall, or to detect the obstacle's boundary. The anterior sensors were used to detect the walls or obstacles that were in the robot's path. They also helped to control the robot's motion when it approached the object or the delivery location.

The robot was also used to relocate the target object. Therefore it was equipped with a gripper to perform the relocating task. The gripper was equipped with a force sensor to sense and control the force that was applied to the object. In the case of the hexapod robot, each leg was equipped with a force sensor that could be used to help control the robot's movement.

- Although there are many options, how should vision technology be integrated with robot technology for autonomous navigation? (This investigation is expected to lead to optimal mobile robot navigation system.)

Vision sensors enable the robot to execute its tasks autonomously; therefore, their use has been investigated for several decades. Accordingly, various techniques, including visual tracking, localisation and mapping, have been developed and used to facilitate navigation. However, vision sensors generally need processors with high computation capacity to process their output signals. In addition, their outputs are influenced by environmental conditions (lighting, texture and background colour) and objects features (i.e. texture and contrast features). Furthermore, the calculations of objects' distances from the camera are still a major challenge in computer vision. Therefore, the robot navigation is achieved by a combination of various sensors.

This study presents a vision-based technology for a robot that executes the visual search for a target object. If it is found, the robot will implement visual tracking to approach the target object before grasping it. When this is completed, the robot will again execute the visual search; however, this time, it will search for the target location where the robot will release the object. If the robot returns to the start location, which is defined by a visual landmark, without finding the target object, the search process has been accomplished.

- Which algorithms should a mobile robot use to search and locate objects in the visual field?

Generally, the robot navigates within its environment, which contains a starting location, target location and number of arbitrarily sized and shaped obstacles. Its objective is to move from the start to the target, without any collision. In the case of the search robot, the robot uses the vision system to detect, recognise and locate the target object, and also some other objects that might be used for its localisation. As a result, the robot has to select the best visual field (camera's viewpoint).

In this study, the method that is in [69] was employed and adapted to perform this task. The two stages were 'where to look next?' and 'where to move next?' In this thesis, the camera did not have zoom capability and only rotated by pan angles (there are no tilt angles). Therefore, these two stages seem to be combined in one stage. However, this method proved to be effective in performing the search process.

- How can a robot be enabled to perform its tasks in different terrains? (The research has the potential to identify limitations, owing to terrain, on the navigation system and path-planning methods.)

Wheeled robots are re-used in most industrial applications, however, some objects may be dropped on the ground (in the robot's path) and obstruct the robot's motion. Even if these obstacles are small and the robot can navigate over them, the robot will consume high energy. Conversely, if the robot follows the obstacles' boundaries, this makes the navigation path and travel time longer. Wheeled robots are also inefficient on very soft or rough surfaces, such as outdoor, unpaved terrains. In contrast, legged robots provide superior mobility in soft and unstructured terrains. They can also move over and

overcome small obstacles more easily than wheeled robots. In this study, both types of robots, hexapod and wheeled robots, were studied and employed to test the methodology used. The hexapod proved to be more flexible in its motion than the wheeled robot. However, it was much slower than the wheeled one. The design and implementation of this type of robot is a more complicated task; particularly, the calculations of the torques and coordinating systems of its joints.

The algorithms of the navigation and path-planning problems are divided, based on the environmental information needed, into global and local formulations. In the former, the terrain map is provided as an input for the robot, and navigation planning involves finding a suitable path between the start and target locations. Consequently, for any changes in the environment the robot needs to update the comprehensive map. These methods have three basic limitations; expensive computation; complex to construct; and difficult to get an accurate graph model. Conversely, in the local formulations, the environment is unknown and the robot needs to acquire the environmental information directly, by using a sensory system. These methods are appropriate for a robot that is designed to navigate in an unknown environment that is constantly changing. They are also easy to construct and optimal for real-time applications.

In the proposed navigation system, the robot used a local method similar to the Bug formulations. The robot used its sensor (vision and range) system to obtain the local environmental information. However, the robot had extra information about its surrounding terrain such as: walls defined the environment; and all obstacles were located close to the walls.

- Finally, the issue of scalability with regard to the size of the robot must be addressed as, in general, this issue has not received sufficient attention from researchers. While no large-scale robots were designed in this project, what are the theoretical challenges of scaling the model robots used to an industrially useful size?

The robotics system mainly involves mechanical and electronic components. The former includes the robot’s chassis and movement tools, such as wheels, legs and motors. The size and weight of these components depend on the required robot size. Conversely, the electronic system mostly consists of the main processor board and the

motor-driver board, which controls and powers the motors. In this study, two small prototypes of robots were used, hexapod and wheeled robots. The major difficulty with designing and implementing similar large robots is that the robot's weight increases dramatically. As such, large and more powerful motors, which provide high torque, will be needed to drive the robot. Typically, these types of motors consume a large amount of energy when operating. The electronic system will be the same, although motor-drive amplifiers that provide the motor with high current are needed.

Other scalability issues become apparent when the robot has to execute different tasks. These include size and weight, as well as the controller's capacity. For instance, if the robot is employed in industry to transport objects of various sizes or shapes, it must be able to adjust its gripper as required. In this case, the robot needs three aspects: the ability to change its size; various sensors in order to discover the surrounding environment; and an intelligent controller to deal with different situations.

8.3 Future Work

This study investigates the search mobile robot, to search for, find and relocate the target object in an indoor environment. Future relevant work could include:

- Robot navigation (exploration path): this includes implementing and simulating other navigation algorithms for comparison, and validating their results from experimental testing. It also involves investigating the robot's localisation techniques, such as simultaneous localization and mapping (SLAM) [38] and Kalman Filters [3].
- Object detection: this includes using a vision sensor that has zooming capabilities and is mounted on a pan-tilt system to increase the robot's ability to search large areas.
- Robot construction: it will be interesting if a large-scale robot is built and then used to search a large area, to relocate objects. More sensors must be added to this robot to increase the robot's capability to explore its environment.
- Robot intelligence: it is important to implement an autonomous system using an artificial intelligent controller, such as a Fuzzy controller or a neural network control system.

References

[1] A. S. Khusheef, G. Kothapalli, and M. Tolouei-Rad, "Simulation of a mobile robot navigation system," presented at the 19th International Congress on Modelling and Simulation, pp. 318-323, Perth, Australia, 12–16 December, 2011.

[2] J. Ng and T. Bräunl, "Performance Comparison of Bug Navigation Algorithms," *Journal of Intelligent and Robotic Systems,* vol. 50, pp. 73-84, 2007.

[3] R. Negenborn, "Robot localization and kalman flters," Master's thesis, Utrecht University, Netherlands, 2003.

[4] Y. Fukazawa, C. Trevai, J. Ota, H. Yuasa, T. Arai, and H. Asama, "Controlling a mobile robot that searches for and rearranges objects with unknown locations and shapes," in *proceedings of the IEEE/RSJ International Conference onIntelligent Robots and Systems (IROS)* vol.2, pp. 1721-1726, Las Vegas. hevada, 27-31 October, 2003.

[5] J. Taheri and N. Sadati, "A fully modular online controller for robot navigation in static and dynamic environments," in *proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation* vol. 1, pp. 163-168, 16-20 July, 2003.

[6] T. Sogo, H. Ishiguro, and T. Ishida, "Mobile robot navigation by a Distributed Vision System," *New Generation Computing,* vol. 19, pp. 121-137, 2001.

[7] A. Mokhtari, A. Benallegue, and Y. Orlov, "Exact linearization and sliding mode observer for a quadrotor unmanned aerial vehicle," *International Journal of Robotics and Automation,* vol. 21, pp. 39-49, 2006.

[8] D. Sadhukhan, "Autonomous Ground Vehicle Terrain Classification Using Internal Sensors," Master's thesis, Department of Mechanical Engineering, Florida State University, USA, 2004.

[9] G. N. Desouza and A. C. Kak, "Vision for mobile robot navigation: a survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, pp. 237-267, 2002.

[10] U. W. E. Bristol. (last visited in April, 2011). *Robotics in University of the West of England*. Available: http://courses.uwe.ac.uk/h671/2011

[11] M. F. Silva and J. A. T. Machado, "A Historical Perspective of Legged Robots," *Journal of Vibration and Control,* vol. 13, pp. 1447–1486, 2007.

[12] M. H. Raibert, "Legged robots," *Commun. ACM,* vol. 29, pp. 499-514, 1986.

[13] A. A. Transeth and K. Y. Pettersen, "Developments in Snake Robot Modeling and Locomotion," in *the 9th International Conference on Control, Automation, Robotics and Vision (ICARCV '06)*, pp. 1-8, Singapore, 5-8 December, 2006.

[14] T. W. Mather and M. Yim, "Modular configuration design for a controlled fall," in *the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* pp. 5905-5910, St. Louis, MO, 10-15 October, 2009.

[15] G. E. Jan, C. Ki Yin, and I. Parberry, "Optimal path planning for mobile robot navigation," *IEEE/ASME Transactions on Mechatronics,* vol. 13, pp. 451-460, 2008.

[16] C. Silva and B. Ribeiro, "Navigating mobile robots with a modular neural architecture," *Neural Computing and Applications,* vol. 12, pp. 200-211, 2003.

[17] N. Sariff and N. Buniyamin, "An Overview of Autonomous Mobile Robot Path Planning Algorithms," in *proceedings of the 4th Student Conference on*

Research and Development (SCOReD), , pp. 183-188, Selangor, Malaysia, 27-28 June, 2006.

[18] A. G. N. C. d. Santos, "Autonomous mobile robot navigation using smartphones," Master's thesis, Instituto Superior Tecnico, 2008.

[19] M. Salem, A. R. Abbasi, and Q. S. M. Zia-ul-Haque, "On indigenous design of prototypic tele-operated mobile robot for ventilation duct inspection in nuclear power plants," in *the 6th International Conference on Emerging Technologies (ICET)* pp. 94-99, Islamabad, Pakistan, 18-19 October, 2010.

[20] K. Nasahashi, T. Ura, A. Asada, T. Obara, T. Sakamaki, K. Kim, and K. Okamura, "Underwater volcano observation by autonomous underwater vehicle "r2D4"," in *Oceans 2005 - Europe*, Vol. 1, pp. 557-562, 20-23 June, 2005.

[21] NASA. (last visited in July, 2012). *Mars exploration rovers*. Available: http://marsrovers.nasa.gov/gallery/press/opportunity/20120117a.html

[22] N. Ruangpayoongsak, H. Roth, and J. Chudoba, "Mobile robots for search and rescue," in *proceedings of the 2005 IEEE International Workshop on Safety, Security and Rescue Robotics*, pp. 212-217, Kobe, Japan, 6-9 June, 2005.

[23] M. Tomono, "Planning a path for finding targets under spatial uncertainties using a weighted Voronoi graph and visibility measure," in *proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* vol. 1, pp. 124-129, Las Vegas, Nevada, October, 27-31 October, 2003.

[24] H. Miao, "Robot path planning in dynamic environment using a simulated annealing based approach," Master's thesis, Science and Technology, Queensland University, Queensland, 2009.

[25] Y. Zhu, T. Zhang, J. Song, and X. Li, "A New Bug-type Navigation Algorithm Considering Practical Implementation Issues for Mobile Robots," in *proceedings of the 2010 IEEE International Conference on Robotics and Biomimetics*, pp. 531-536, Tianjin, China, 14-18 December, 2010.

[26] D. Huiying, L. Wenguang, Z. Jiayu, and D. Shuo, "The Path Planning for Mobile Robot Based on Voronoi Diagram," in *the 3rd International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, pp. 446-449, Shenyang, China, 1-3 November, 2010.

[27] D. Bodhale, N. Afzulpurkar, and N. T. Thanh, "Path planning for a mobile robot in a dynamic environment," in *proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 2115-2120, Bangkok, Thailand, 21 - 26 February, 2009.

[28] V. Nguyen Hoang, V. Ngo Anh, L. SeungGwan, and C. TaeChoong, "Obstacle Avoidance Path Planning for Mobile Robot Based on Multi Colony Ant Algorithm," in *the First International Conference on Advances in Computer-Human Interaction*, pp. 285-289, Sainte Luce, France, 10-15 February, 2008.

[29] A. Sarmiento, R. Murrieta, and S. A. Hutchinson, "An efficient strategy for rapidly finding an object in a polygonal world," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, vol. 2, pp. 1153-1158, Las Vegas, Nevada, 27-31 October, 2003.

[30] Haye Lau, Shoudong Huang, and G. Dissanayake, "Optimal Search for Multiple Targets in a Built Environment," presented at the proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3740 - 3745, Edmonton, Alberta, Canada, 2-6 August, 2005.

[31] B. Tovar, S. M. LaValle, and R. Murrieta, "Optimal navigation and object finding without geometric maps or localization," presented at the Proceedings of

the 2003 IEEE lnternaliooal Conference on Robotics & Automation, pp. 464-470, Taipei, Taiwan, September 14-19, 2003.

[32] J. Ng and T. Bräunl, "Robot navigation with a guide track," in *Fourth International Conference on Computational Intelligence, Robot and Autonomous Systems*, pp. 37-43, Palmerston North, New Zealand, 28-30 November, 2007.

[33] D. H. Kim and S. Shin, "Local path planning using a new artificial potential function composition and its analytical design guidelines," *Advanced Robotics,* vol. 20, pp. 115-135, 2006.

[34] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and control*. USA: Wiley, 2006.

[35] G. Macesanu, T. T. Codas, C. Suliman, and B. Tarnauca, "Development of GTBoT, a high performance and modular indoor robot," in *the IEEE International Conference on Automation Quality and Testing Robotics (AQTR)*, vol. 1, pp. 1-6, Cluj-Napoca, Romania, 28-30 May, 2010.

[36] A. K. Ray, L. Behera, and M. Jamshidi, "GPS and sonar based area mapping and navigation by mobile robots," in *the 7th IEEE International Conference on Industrial Informatics (INDIN)* pp. 801-806, Cardiff, Wales, 23-26 June, 2009.

[37] C. Hyeyeon, C. JongSuk, and K. Munsang, "Experimental research of probabilistic localization of service robots using range image data and indoor GPS system," in *the IEEE Conference on Emerging Technologies and Factory Automation (ETFA '06)*, pp. 1021-1027, Prague, Czech, 20-22 September, 2006.

[38] W. Dae Hee, L. Young Jae, S. Sangkyung, and K. Taesam, "Integration of Vision based SLAM and Nonlinear Filter for Simple Mobile Robot Navigation," in *the IEEE National Aerospace and Electronics Conference (NAECON)* pp. 373-378, Dayton, OH, 16-18 July, 2008.

[39] M. Marron, M. A. Sotelo, J. C. Garcia, D. Fernandez, and I. Parra, "3D-Visual Detection of Multiple Objects and Structural Features in Complex and Dynamic Indoor Environments," in *the 32nd Annual Conference on IEEE Industrial Electronics (IECON)*, pp. 3373-3378, Paris, France, 6-10 November, 2006.

[40] M. Elmogy and Z. Jianwei, "Robust real-time landmark recognition for humanoid robot navigation," in *proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 572-577, Bangkok, Thailand, 21 - 26 February , 2009.

[41] H. Xiaovvei, L. Junsheng, L. Yanping, and X. Xinhe, "An approach of color object searching for vision system of soccer robot," in *the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 535-539, Shenyang, China, 22-26 August, 2004.

[42] R. M. Jusoh, "Application of Vision Target Localization for Mobile Robot," in *the 4th Student Conference on Research and Development (SCOReD)* pp. 144-146, Selangor, Malaysia, 27-28 June, 2006.

[43] M. Mata, J. m. Armingol, J. Fernndez, and A. D. l. escalera, "Object learning and detection using evolutionary deformable models for mobile robot navigation," *Robotica,* vol. 26, pp. 99-107, 2008.

[44] M. Abdellatif, "A vision-based navigation control system for a mobile service robot," presented at the SICE, 2007 Annual Conference, p.p 1517-1522, Kagawa University, Japan, 17-20 September, 2007.

[45] P. Jae-Han, B. Seung-Ho, K. Jaehan, P. Kyung-Wook, and B. Moon-Hong, "A new object recognition system for service robots in the smart environment," in *the International Conference on Control, Automation and Systems (ICCAS '07)* pp. 1083-1087, Seoul, South Korea,17-20 October, 2007.

[46] R. Lin, Z. Du, F. He, M. Kong, and L. Sun, "Tracking a moving object with mobile robot based on vision," in *the IEEE International Joint Conference on Neural Networks (IJCNN) (IEEE World Congress on Computational Intelligence)*, pp. 716-720, Hong Kong, 1-8 June, 2008.

[47] B. Browning and M. Veloso, "Real-time, adaptive color-based robot vision," in *the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3871-3876, 2-6 August, 2005.

[48] X. Guihua and X. Zhuoyi, "A New Algorithm for Target Recognition and Tracking for Robot Vision System," in *the IEEE International Conference on Control and Automation (ICCA)* pp. 1004-1008, Guangzhou, China, May 30-June 1, 2007.

[49] L. Ding, A. Goshtasby, and S. M., "Volumetric image registration by template matching," *Image and Vision Computing*, vol. 19, pp. 821-832, 2001.

[50] R. Takei. (2003). *A new grey scale template image matching algorithm using the cross-sectional histogram correlation method*. Available: http://www.math.ucla.edu/~rrtakei/prevRsrch/workreport03.pdf

[51] S. Omachi and M. Omachi, "Fast Template Matching With Polynomials," *IEEE Transactions on Image Processing*, vol. 16, pp. 2139-2149, 2007.

[52] Q. Do and L. Jain, "Application of neural processing paradigm in visual landmark recognition and autonomous robot navigation," *Neural Computing and Applications*, vol. 19, pp. 237-254, 2010.

[53] D. G. Lowe, "Object recognition from local scale-invariant features," in *proceedings of the Seventh IEEE International Conference on Computer Vision* vol.2, pp. 1150-1157, Kerkyra, Greece, 20-27 September, 1999.

[54] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int. J. Computer Vision*, vol. 60, pp. 91-110, 2004.

[55] S. Se, D. Lowe, and J. Little, "Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks," *The International Journal of Robotics Research*, vol. 21, pp. 735-758, 2002.

[56] L. Juan and O. Gwun, "A Comparison of SIFT, PCA-SIFT and SURF," *International Journal of Image Processing (IJIP)*, vol. 3, pp. 143-152, 2009.

[57] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded Up Robust Features." vol. 3951, A. Leonardis, H. Bischof, and A. Pinz, Eds., ed: Springer Berlin / Heidelberg, pp. 404-417, 2006.

[58] A. Alipoor and M. Fesharaki, "Multi objective optimization for object recognition," in *the 2nd International Conference on Education Technology and Computer (ICETC)*, , vol. 2, pp. 70-73, Shanghai, China, 22-24 June, 2010.

[59] S. Ekvall, P. Jensfelt, and D. Kragic, "Integrating Active Mobile Robot Object Recognition and SLAM in Natural Environments," in *proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5792-5797, Beijing, China, 9-15 October, 2006.

[60] S. Ekvall, D. Kragic, and F. Hoffmann, "Object recognition and pose estimation using color cooccurrence histograms and geometric modeling," *Image and Vision Computing*, vol. 23, pp. 943-955, 2005.

[61] Z. Qingjie, D. Hongbin, Z. Wenyao, and M. Aixia, "Selection of Image Features for Robot Vision System," in *the IEEE International Conference on Automation and Logistics*, pp. 2622-2626, Jinan, China, 18-21 August, 2007.

[62] M. R. Blas, M. Agrawal, A. Sundaresan, and K. Konolige, "Fast color/texture segmentation for outdoor robots," in *proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4078-4085, Nice, France, 22-26 September, 2008.

[63] L. C. Lulio, M. L. Tronco, and A. J. V. Porto, "Pattern recognition structured heuristics methods for image processing in mobile robot navigation," in *the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4970-4975, Taipei, Taiwan, 18-22 October, 2010.

[64] J. Yong-Gun, L. Ja-Yong, and K. Hoon, "Segmentation tracking and recognition based on foreground-background absolute features, simplified SIFT, and particle filters," in *the IEEE Congress conference on Evolutionary Computation (CEC)*, pp. 1279-1284, Vancouver, BC, Canada, 16-21 July , 2006.

[65] G. Medioni, A. R. J. Francois, M. Siddiqui, K. Kim, and H. Yoon, "Robust real-time vision for a personal service robot," *Comput. Vision and Image Understanding*, vol. 108, pp. 196-203, 2007.

[66] L. Hao and S. X. Yang, "An evolutionary visual landmark recognition system," in *the IEEE International Conference on Systems, Man and Cybernetics*, vol. 7, 6-9 October, 2002.

[67] C. Zhichao and S. T. Birchfield, "Visual detection of lintel-occluded doors from a single image," in *the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1-8, Anchorage, Alaska, 23-28 June, 2008.

[68] V. N. Murali and S. T. Birchfield, "Autonomous navigation and mapping using monocular low-resolution grayscale vision," in *the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW '08)*, pp. 1-8, Anchorage, Alaska, 23-28 June, 2008.

[69] K. Shubina and J. K. Tsotsos, "Visual search for an object in a 3D environment using a mobile robot," *Comput. Vis. Image Underst.*, vol. 114, pp. 535-547, 2010.

[70] DMP-Electronics-Inc. (last visited in July, 2012). *RoBoard RB-100 manual V2.00*. Available: http://www.roboard.com/Files/RB-100/RoBoard_RB-100_v2r0B.pdf

[71] Z.-y. Wang, X.-l. Ding, and A. Rovetta, "Analysis of typical locomotion of a symmetric hexapod robot," *Robotica*, vol. 28, pp. 893-907, 2010.

[72] J. Iovine. Hexapod walker robot. *Poptronics*. vol. 2, pp. 42-46, 2001. Available: http://ezproxy.ecu.edu.au/login?url=http://search.proquest.com/docview/204773656?accountid=10675

[73] Y. Go, Y. Xiaolei, and A. Bowling, "Navigability of multi-legged robots," *IEEE/ASME Transactions on Mechatronics*, vol. 11, pp. 1-8, 2006.

[74] M. S. Erden and K. Leblebicioglu, "Torque Distribution in a Six-Legged Robot," *IEEE Transactions on Robotics*, vol. 23, pp. 179-186, 2007.

[75] Robotgear. (last visited in July, 2012). *4 channel motor control unit*. Available: http://www.robotgear.com.au/Cache/Files/Files/158_4ChannelDCMotorControllerManual.pdf

[76] T. Z. Jr. (last visited in July, 2012). *"Robotic gripper sizing: the science, technology and lore,"* Available: http://www.grippers.com/size.htm

[77] A. P. Nanda, "Design and development of a two-jaw parallel pneumatic gripper for robotic manipulation," Bachelor's project report, Mechanical Engineering, National institute of Technology Rourkela, 2010.

[78] Txcess-Surplus. (last visited in 2012, July 30). *IBM Lenovo Thinkcentre M57 type 6087 CB5 motherboard*. Available: http://www.txcesssurplus.com/servlet/the-9249/IBM-Lenovo-Thinkcentre-M57/Detail

[79] S. Baskiyar and N. Meghanathan, "A survey of contemporary real-time operating systems," *Informatica - An International Journal of Computing and Informatics*, vol. 29, pp. 233-240, 2005.

[80] Ubuntu. (last visited in December, 2011). *Ubuntu desktop 12.04 LTS*. Available: http://www.ubuntu.com/download/ubuntu/download

[81] S. Montabone. (last visited in December, 2011). *Installing OpenCV 2.1 in Ubuntu*. Available: http://www.samontab.com/web/2010/04/installing-opencv-2-1-in-ubuntu

[82] C. Evans. (last visited in July, 2012). *Computer vision and image processing, the OpenSURF computer vision library*. Available: http://www.chrisevansdev.com/computer-vision-opensurf.html

[83] Arduino. (last visited in July, 2012). Available: http://www.arduino.cc/

[84] Electrosoft. (last visited in April, 2012). Available: http://electrosofts.com/parallel/

[85] Robot-Electronics. (last visited in July, 2012). Available: http://robot-electronics.co.uk/acatalog/examples.html

[86] Logitech. (last visited in January, 2013). Available: http://www.logitech.com/en-au/search?q=c200

[87] D. Cagigas and J. Abascaly, "An Efficient Robot Path Planning System for Large Environments Using Pre-Calculated Paths " in *proceedings of the the 10th Mediterranean Conference on Control and Automation*, Lisbon, Portugal, 9-12 July, 2002.

[88] B. I. Kazem, A. H. Hamad, and M. M. Mozael, " Modified vector field histogram with a neural network learning model for mobile robot path planning and obstacle avoidance " *International Journal of Advancements in Computing Technology* vol. 2 pp. 166-173, 2010.

[89] V. Lumelsky and A. Stepanov, "Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," *Algorithmica*, vol. 2, pp. 403-430, 1987.

[90] A. Sankaranarayanan and M. Vidyasagar, "A new path planning algorithm for moving a point object amidst unknown obstacles in a plane," in *the Proceedings of the IEEE International Conference on Robotics and Automation* vol. 3, pp. 1930-1936, Cincinnati, OH, 13-18 May, 1990.

[91] J. Ng, "An analysis of mobile robot navigation algorithms in unknown environments," PhD's thesis, School of Electrical, Electronic and Computer Engineering, The University of Western Australia ,WA, 2010.

[92] H. Carey. (last visited in July, 2012). *Reducing sidelobes of SRF10*. Available: http://robot-electronics.co.uk/htm/reducing_sidelobes_of_srf10.htm

[93] SRF08. (last visited in 2012, July 30). *SRF08 Ultra sonic range finder* Available: http://robot-electronics.co.uk/htm/srf08tech.html

[94] Phidgets. (last visited in July, 2012). *Interlink electronics 0.2" circular FSR*. Available: http://www.phidgets.com/products.php?product_id=3103

[95] N. Gupta, R. Gupta, A. Singh, and M. Wytock, "Object recognition using template matching," (2008). Available in: https://tmatch.googlecode.com/svn-history/r38/trunk/report/report.pdf.

[96] Black-ice. (last visited in 2012, July 30). *HSI colour space - colour space conversion*. Available: http://www.blackice.com/colorspaceHSI.htm

[97] A. S. Khusheef, G. Kothapalli, and M. Tolouei-Rad, "An approach for integration of industrial robot with vision system and simulation software," *Journal of World Academy of Science, Engineering and Technology*, vol. 58, pp. 18-23, 2011.

[98] B. F. Wu, W. C. Lu, and C. L. Jen, "Monocular Vision-Based Robot Localization and Target Tracking," *Journal of Robotics*, vol. 2011, pp. 1-12, 2011.

[99] R. Fisher, S. Perkins, A. Walker, and E. Wolfart. (last visited in July, 2012). *Morphology*. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/morops.htm

[100] G. Bradski and A. Kaehler, *Learning OpenCV: computer vision with the OpenCV library*. USA: O'Reilly, 2008.

Appendices

A.1 The Wheeled Robot

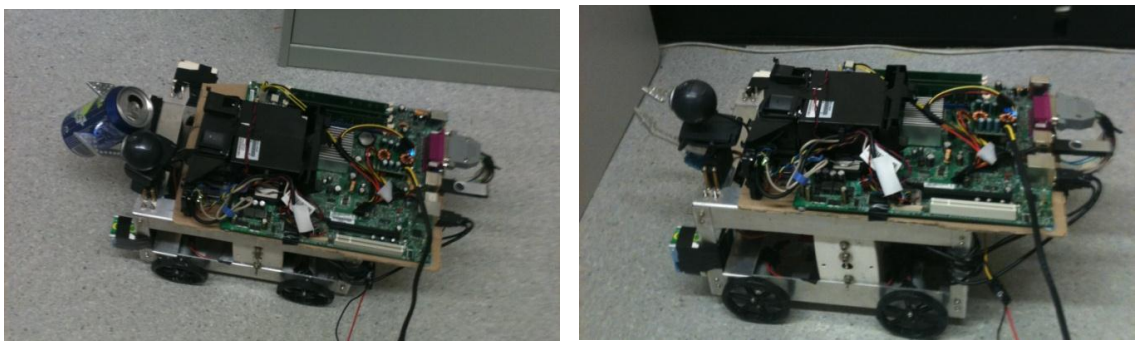


Figure A.1. The wheeled robot

A.1.1 Wheel Configuration

Wheels are the most popular locomotion mechanism in mobile robotics because wheeled mobile robots can be precisely controlled, are easy to program and require relatively simple mechanical implementation. Wheels are also very well suited for locomotion on flat structured surfaces where they are more efficient than a legged locomotion. Wheeled robots use different types of wheel and motor drive configurations. The wheel configurations are important because they influence the controllability and stability of the mobile robot.

There are many types of wheel configuration, in which the number of wheels on the robot rises from two to four or more. Let us explain the configurations that have up to four wheels. To start with, some robots have a two-wheeled configuration. These robots typically use a two-wheel differential drive mechanism for moving and changing direction. In a differential drive mechanism, the mobile robot's movement is based on two independently driven wheels placed on each side of the robot body. The robot can change its direction by altering the relative angular velocity of the wheels and therefore, does not require an additional steering mechanism. The disadvantages of two-wheel robots are that they lack stability and require an accurate method of distribution of the robot's weight.

Some robotics designers insert a third wheel that rotates freely to add more balance for their robots. The third wheel could be driven by a servo or a stepper motor to steer the

robot. The main benefit of this design, which is called tricycle drive, is that it is easy to control. However, the robot needs a long turning curve to change its direction. Some robots are designed to combine drive and steer mechanisms (tricycle drive) in the third wheel. In such a case, this wheel is driven by two motors (usually a DC motor for motion and a servo or stepper motor for steering), whereas the other two wheels rotate freely and are positioned for balance. In addition to the same disadvantages of the previous tricycle drive, this mechanism of steering and driving at the same time is difficult to build. Another three-wheeled configuration that employs a two-motor drive configuration is called the synchronic drive. In this, one motor drives all three wheels while the second motor steers all them. A robot with synchronic configuration guarantees straight-line motion but it has complex mechanisms yet to be developed.

Typically, increasing the number of wheels will improve the robot's stability and accordingly, many designers use the four-wheeled configuration. Some wheeled mobile robots are designed using the car steering configuration. In this case, the robot uses two driving motors attached to the rear or the front wheels to provide motion and two or one motor to steer. The main problem in this configuration is that robot needs to travel a long arc to change its direction. Another configuration also uses four wheels but employs a differential drive mechanism. The differential drive configuration is popular in robotics because it is easy to develop. However, the steering of the robot is more difficult to control than the robot using the car steering configuration. Lastly, some robots have four motor drives within a zero-turn radius configuration. The robot using this configuration reorients itself by rotating each pair of wheels that are mounted on one side in the same direction, whilst rotating another pair in the opposite direction. If the robot wants to drive in a straight line, it will rotate all the wheels with the same speed and direction.

A.1.2 Motors

There are three types of motors that can be considered for driving mobile robots: DC motors, stepper motors and servo motors. Let us give a brief argument for each class:

DC motors

DC motors are widely used to drive wheeled robots because of their small size and high output energy. They are defined by the their operating DC voltage that might be rated as few as 1.5 Volts up to or more than 100 Volts. The ideal DC motor generates high torque whilst requiring low current. However, the input power (current × voltage) indicates the mechanical power output. Consequently, if the DC motor were required to create more output mechanical torque, it would draw a higher current whilst the DC voltage remains constant. The maximum input current and maximum output torque will be at the stalled state when the motor starts rotating or during maximum load. The DC motors are also specified by their rotation speed, which is defined as rotations per minute (RPM) when the motor is running freely. Characteristically, DC motors rotate at speeds approaching the thousands of RPM. Therefore, gearboxes are added to reduce the speed and increase the output mechanical torque. The motors RPM and output torque specify the motors output power. The greatest output power is achieved mid-way between the maximum speed (no torque) and the maximum torque (no speed).

Typically, the speed of a DC motor is controlled using a pulse width modulation (PWM) technique, which controls the amount of input power to the motor. This is performed by digital circuitry that creates the square waves that rapidly switch between "on" and "off". The on-off model simulates input power between full "on" and "off" by changing the percentage of time "on" versus the percentage of time "off". For instance, if it is desired that the motor rotate at half-speed, the time of the switch "on" is equal to the time of the switch "off" in the square PWM waveforms.

Stepper motors

A stepper motor is a permanent magnetic motor that has shaft moves (steps) between discrete rotary positions each time the controller gives one pulse. They could be rotated at a specific number of steps with a high accuracy. Accordingly, they are used for applications that need high positioning accuracy, such as robot steering. Typically, stepper motors are controlled by series of pulses until their shafts reach the desired location.

Servo motor

A servo motor has different components in one package: a small DC motor, gear reduction to increase torque and electronic shaft to sense position and control circuit. The main difference between the servo and stepper motors is that the former have a feedback control loop. This loop drives the servo motor's shaft to the desired position assigned by the user. Consequently, if the servo motor encounters an obstacle in the rotation path, it will continue trying until it either reaches the target rotation or harms itself. However, if the stepper motor meets the same obstacle, it can bounce steps without damaging itself. Servo motors are generally used for those applications that need specific alternating movements between two positions, such as the joints of a legged robot. Typically, they have three wires: two for powering and one for control. The servo motors are controlled by a series of pulses that indicate the desired position of the shaft.

The PC motherboard and the microcontroller need to communicate and control each other. Normally, in the communication process the PC behaves as a master (host) and each of other peripherals are slaves (devices). Typically, the PC motherboard has ports that can be used mainly to control and communicate with external devices. Let us evaluate the three main types of PC ports: USB, Serial and parallel ports.

A.1.3 Communication ports

Universal serial bus (USB)

USB has been widely used to interface computers to other peripheral devices, such as memory sticks, computer mice and keyboards. It has been also introduced for many other industrial applications including measurement and automation. Typically, the USB connector can be up to 5 metres long and has four main connections: Power (normally 5v), Ground and a twisted pair of differential +/- data lines. The data and acknowledgement transaction will consist of a number of packets (specific numbers of Bytes for each packet) that the Master is sending or requiring. The main benefit of USB is that it supports the high data rates required within the computer and peripherals.

Serial port

A serial port is mostly used to transmit communication data, such as transferring files between the computers. Generally, the serial port has 9 pins: pins 2, 3 and 5 are used to transmit data, receive data and signal ground, respectively, whilst the other pins are used to control the communication process. The '1' and '0' are the data, which define a voltage level of 3V to 25V and -3V to -25V, respectively. The data are sent and received one bit after another with some extra bits, such as start bit, stop bit and parity bit to detect errors.

Parallel port

The parallel port, which is sometimes called the printer port, has 25 pins. These pins are classified into four categories: data registers (pins 2-9), control registers (pins 1, 14, 16 and 17), status registers (pins 10-13 and 15), and grounds (pins 18-25). The data registers can be used to store a byte of data that is sent to the port data register pins. The control registers are mainly used to send control data to the printer port while the status registers could read the states of the status pins. The data are sent as 8 bits of byte to the data pins at a time while the port can receive 5 bits of data from the status pins. The parallel port pins can be used individually to send and receive data. The port is easier to program and faster compared with serial ports.

The microcontroller board has a USB connection that works as a serial connection. Typically, this port is used to download the control program and to power the microcontroller. This port could be used as a communication interface between both the microcontroller and the motherboard. In this case, two programs (one program for each board to control another board) have to be written. As mentioned above, the serial port is more difficult to program and control than the parallel port. It also requires additional CPU time to process and assess the communication messages between the two boards. Consequently, the decision taken is to use some parallel port pins to send 2 bits of data to control the microcontroller and to receive 2 bits of data from the microcontroller to control the motherboard.

- More information about PC ports is given by [http://electrosofts.com/parallel/].

A.1.4 Part List

Table A.1: *Part list*

Name	Unit price	Qty	Total price ($)	Supplier
DC motor	24.95	4	99.80	http://www.jaycar.com.au/
Microcontroller	88.55	1	88.55	http://www.robotgear.com.au/
Motor drive	27.95	1	27.95	http://www.robotgear.com.au/
Servo motor	39.95	3	119.85	http://www.jaycar.com.au/
Servo motor drive	39.95	1	39.95	http://www.jaycar.com.au/
SRF08 Range Finder	53.5	5	267.5	http://www.robotgear.com.au/
USB flash memory	13.99	1	13.99	
Nuts and Screws	18.45	1	18.45	http://www.jaycar.com.au/
DC power supply	98.28	1	98.28	e-bay
Wheels	11.66	2 pairs	23.32	http://www.robotgear.com.au/
Pololu universal mounting	8.75	2 pairs	17.5	http://www.robotgear.com.au/

A.2 The hexapod Mobile Robot

A.2.1 Kinematic Modelling

Table A.2:

DH parameters of the robot leg on the left side

Link	a_i	α_i	d_i	θ_i
1	a_1	$\frac{\pi}{2}$	0	θ_1
2	a_2	0	0	θ_2
3	a_3	0	0	θ_3

$$A_i = \begin{bmatrix} C_{\theta_i} & -S_{\theta_i}C_{\alpha_i} & S_{\theta_i}S_{\alpha_i} & a_i C_{\theta_i} \\ S_{\theta_i} & C_{\theta_i}C_{\alpha_i} & -C_{\theta_i}S_{\alpha_i} & a_i S_{\theta_i} \\ 0 & S_{\alpha_i} & C_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(A.1)}$$

$$A_1 = \begin{bmatrix} C_{\theta_1} & 0 & S_{\theta_1} & a_1 C_{\theta_1} \\ S_{\theta_1} & 0 & -C_{\theta_1} & a_1 S_{\theta_1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} C_{\theta_2} & -S_{\theta_2} & 0 & a_2 C_{\theta_2} \\ S_{\theta_2} & C_{\theta_2} & 0 & a_2 S_{\theta_2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} C_{\theta_3} & S_{\theta_3} & 0 & a_3 C_{\theta_3} \\ S_{\theta_3} & -C_{\theta_3} & 0 & a_3 S_{\theta_3} \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(A.2)}$$

$$A_2^0 = A_1 A_2 = \begin{bmatrix} C_{\theta_1} C_{\theta_2} & -S_{\theta_2} C_{\theta_1} & S_{\theta_1} & a_2 C_{\theta_1} C_{\theta_2} + a_1 C_{\theta_1} \\ S_{\theta_1} C_{\theta_2} & -S_{\theta_1} S_{\theta_2} & -C_{\theta_1} & a_2 S_{\theta_1} C_{\theta_2} + a_1 S_{\theta_1} \\ S_{\theta_2} & C_{\theta_2} & 1 & a_2 S_{\theta_2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(A.3)}$$

$$T_{3(left)}^0 = \begin{bmatrix} R_3^0 & O_3^0 \\ 0 & 0 \end{bmatrix} = A_1 A_2 A_3 \quad \text{(A.3)}$$

Performing the required calculations yields the forward kinematic modelling:

$$T_{3(left)}^0 = \begin{bmatrix} C_{\theta_1} C_{(\theta_2+\theta_3)} & C_{\theta_1} S_{(\theta_2+\theta_3)} & -S_{\theta_1} & C_{\theta_1}(a_1 + a_2 C_{\theta_2} + a_3 C_{(\theta_2+\theta_3)}) \\ S_{\theta_1} C_{(\theta_2+\theta_3)} & S_{\theta_1} S_{(\theta_2+\theta_3)} & C_{\theta_1} & S_{\theta_1}(a_1 + a_2 C_{\theta_2} + a_3 C_{(\theta_2+\theta_3)}) \\ S_{(\theta_2+\theta_3)} & -C_{(\theta_2+\theta_3)} & 0 & a_2 S_{\theta_2} + a_3 S_{(\theta_2+\theta_3)} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(A.4)}$$

A.2.2 Jacobian

The origins of the DH frames of the left legs are given by

$$O_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, O_1 = \begin{bmatrix} a_1 C_{\theta_1} \\ a_1 S_{\theta_1} \\ 0 \end{bmatrix}, \quad O_2 = \begin{bmatrix} C_{\theta_1}(a_1 + a_2 C_{\theta_2}) \\ S_{\theta_1}(a_1 + a_2 C_{\theta_2}) \\ a_2 S_{\theta_2} \end{bmatrix}$$

$$O_3 = \begin{bmatrix} C_{\theta_1}(a_1 + a_2 C_{\theta_2} + a_3 C_{(\theta_2+\theta_3)}) \\ S_{\theta_1}(a_1 + a_2 C_{\theta_2} + a_3 C_{(\theta_2+\theta_3)}) \\ a_2 S_{\theta_2} + a_3 S_{(\theta_2+\theta_3)} \end{bmatrix} \quad \text{(A.5)}$$

The Z_{i-1} of the DH frames are given by:

$$Z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \qquad Z_1 = Z_2 = \begin{bmatrix} S_{\theta_1} \\ -C_{\theta_1} \\ 0 \end{bmatrix} \quad \text{(A.6)}$$

Performing the required calculations yields:

$$J = \begin{bmatrix} -S_{\theta_1}(a_1 + a_2C_{\theta_2} + a_3C_{(\theta_2+\theta_3)}) & -a_2C_{\theta_1}S_{\theta_2} - a_3C_{\theta_1}S_{(\theta_2+\theta_3)} & -a_3C_{\theta_1}S_{(\theta_2+\theta_3)} \\ C_{\theta_1}(a_1 + a_2C_{\theta_2} + a_3C_{(\theta_2+\theta_3)}) & -a_2S_{\theta_1}S_{\theta_2} - a_3S_{\theta_1}S_{(\theta_2+\theta_3)} & -a_3S_{\theta_1}S_{(\theta_2+\theta_3)} \\ 0 & a_2C_{\theta_2} + a_3C_{(\theta_2+\theta_3)} & a_3C_{(\theta_2+\theta_3)} \\ 0 & S_{\theta_1} & S_{\theta_1} \\ 0 & -C_{\theta_1} & -C_{\theta_1} \\ 1 & 0 & 0 \end{bmatrix} \quad \text{(A.7)}$$

A.2.3 Torque Distribution

The joint static torques are calculated by

$$\tau = J^T(q)\, F \quad \text{(A.8)}$$

The resulting joint static torques of the legs on the left side of the robot's body are then given as:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} = \begin{bmatrix} -S_{\theta_1}(a_1 + a_2C_{\theta_2} + a_3C_{(\theta_2+\theta_3)}) & C_{\theta_1}(a_1 + a_2C_{\theta_2} + a_3C_{(\theta_2+\theta_3)}) & 0 & 0 & 0 & 1 \\ -a_2C_{\theta_1}S_{\theta_2} - a_3C_{\theta_1}S_{(\theta_2+\theta_3)} & -a_2S_{\theta_1}S_{\theta_2} - a_3S_{\theta_1}S_{(\theta_2+\theta_3)} & a_2C_{\theta_2} + a_3C_{(\theta_2+\theta_3)} & S_{\theta_1} & -C_{\theta_1} & 0 \\ -a_3C_{\theta_1}S_{(\theta_2+\theta_3)} & -a_3S_{\theta_1}S_{(\theta_2+\theta_3)} & a_3C_{(\theta_2+\theta_3)} & S_{\theta_1} & -C_{\theta_1} & 0 \end{bmatrix} \begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix} \quad \text{(A.9)}$$

A.2.4 Part List

Table A.2: *The main parts of the hexapod*

Name	Link
Hitec HS-645MG Servo	http://www.robotshop.ca/hitec-hs645mg-servo-motor.html
Hitec HSR-5990TG Servo	http://www.servodatabase.com/servo/hitec/hsr-5990tg
RoBoard Starter Kit	http://www.robotshop.ca/roboard-starter-kit-3.html
1600mAh battery pack	www.joondaluphobbies.com.au

A.3 Related publications

The following papers are published or in preparation in support of this thesis:

1. S. Khusheef, G. Kothapalli, and M. Tolouei-Rad, "Simulation of a mobile robot navigation system," presented at the 19th International Congress on Modelling and Simulation, p.p 318-323, Perth, Australia, December 2011.
2. S. Khusheef, G. Kothapalli, and M. Tolouei-Rad, "An approach for integration of industrial robot with vision system and simulation software," *Journal of World Academy of Science, Engineering and Technology,* vol. 58, pp. 18-23, 2011.
3. S. Khusheef, G. Kothapalli, and M. Tolouei-Rad, "Efficient utilization of sensors and processors to enhance performance of mobile robots," Postgraduate Electrical Engineering and Computing Symposium (PEECS 2012), Perth, Australia, 9 November 2012.

Submitted for publication:

4. S. Khusheef, G. Kothapalli, and M. Tolouei-Rad, "Vision-based relocation of object using a mobile robot," the Emirates Journal for Engineering Research (EJER).