Development of an efficient complete area coverage strategy for low cost mobile agricultural robots employing navigation sensors with limited angular resolution and having applications in covering large areas with small obstacles

Masters by Research Thesis

submitted by

Namrata Nath

Principal supervisor: Dr. Sang Heon Lee

Co-supervisor: Dr. Chris Saunders

Submitted in Partial Fulfilment
of the Requirements
for the Degree of Masters by Research

Division of Information Technology, Engineering and the Environment

School of Engineering

University of South Australia

Abstract

Rapid advancements in technology enable robots to have advanced on-board multi-core processing systems, sensors and vision systems that allow for a greater degree of autonomous activity. Such robots can be used in agricultural applications to reduce intensive and monotonous human labour. Examples of activities typically carried out on agricultural areas dedicated to animal husbandry are monitoring levels of food and water, checking for toxicity or contamination of food, water and in the case of indoor environments – air, maintaining cleanliness and hygiene, ensuring frequent animal mobility and herding. One of the basic prerequisites for a robot that can automate some of these tasks would be the ability to intelligently cover the entire area of the farm while avoiding obstacles.

This research aims to develop a maximal area coverage strategy to be used by a low cost autonomous robot having applications in agriculture and livestock monitoring such as the free range poultry farming industry. The motivation for keeping the robot low cost is to enable farmers with smaller farms to invest in them while allowing scalability on larger farms. The navigation algorithm works with the knowledge of the boundaries of the farm, but without any knowledge of the obstacles therein. A new obstacle avoidance algorithm has been developed to aid maximal area coverage while keeping the obstacle circumvention path optimal. This research is based on the use of low cost ultrasonic sensors for navigation since they provide reliable ranging indoors and outdoors and can be easily sourced. However, these sensors have poor angular resolution. Methods to cope with the poor angular resolution have also been developed and included into a new obstacle avoidance algorithm and built into the area coverage strategy. The hardware on the autonomous robotic platform such as sensors has an impact on the obstacle avoidance algorithm and so the strategy used for this research was to initially investigate the possible hardware that went into making the robot, followed by a review of existing literature in the area coverage space. This review helped identify the gaps and inefficiencies of applying existing methods to the target agricultural applications, which in turn helped shape the area coverage strategy developed herein. Specifically, as the amount of free space available in agricultural areas is large and the obstacles are small and do not partition the available space (such as walls in an office building), algorithms designed to work in office buildings add significant overhead to a more simplistic area coverage approach proposed here – the Boustrophedon area coverage using the Hybrid Bug algorithm approach. The effectiveness of the Boustrophedon area coverage using the Hybrid Bug algorithm approach was tested using simulations and comparison studies with another efficient online

coverage algorithm, the BA* algorithm [65], which showed a 33.3% reduction in path length and 9% increase in area coverage on an average, while covering areas populated by smaller obstacles.

In summary, the outcome of this research is an algorithm which is ideally suited for maximal area coverage of agricultural areas and areas where navigable space is large when compared to the quantity, size and density of obstacles. Additionally, it is algorithmically more simplistic and therefore less computationally intensive than area coverage algorithms designed to work indoors that have to deal with large obstacles such as walls and partitions.

Declaration

I, Namrata Nath, declare that this thesis, to the best of my knowledge and belief, does not contain material authored and/or published by another person except where due references or acknowledgements are made. I also certify that no part of the material in this thesis has been accepted for the award of any other degree or diploma in my name by any university or tertiary institution.

_____ _____

(Signed) (Dated)

Acknowledgements

I would like to express my immense gratitude to the people without whom this research would not have been possible.

Firstly, I would like to thank my principal supervisor, Dr. Sang-Heon Lee, who, from the inception of this project to the conclusion, has been the driving force that has kept this research going against all odds. It is not often that one finds mentors like Dr. Sang-Heon Lee who are truly vested in the academic careers of students. His infinite patience, attention to detail, analytical abilities and natural flair for teaching has helped me immensely with conducting research and authoring my thesis and will continue to inspire many.

I would like to thank my co-supervisor Dr. Chris Saunders for his astute observations regarding my work, help during the mechanical design of the robot and unique perspectives on the algorithms developed in this research. In addition to being very patient with my demanding personal circumstances, Dr. Chris Saunders has always offered insights into my research and thesis that have had profound impact.

I would like to thank Abdullah Zawawi Mohamed for initiating me into the practical aspects of robotics during the initial days of this research and enabling my first journal publication. I would like to thank PeiLin Li for sharing his awareness on image recognition. I would also like to thank my fellow students in the Barbra Hardy institute and members of the mechanical lab for always being approachable and helpful.

I am very grateful to Peter Bain, Hugh Considine and Dr. Russell Brinkworth of the electronics department for sharing their knowledge on the electronic components and sensors which was indispensable when the prototype of the mobile robot was constructed. I would like to thank Zack and Utku for ideas on the mathematical aspect of the new admissibility criterion and technical aspects of the TangentBug algorithm respectively.

I am grateful to our babysitter Irina for taking such good care of our son.

Finally, I would like to thank my family – My husband, Dr. Raghunandan Parthasarathy, who has shown a great deal of patience, for having supported me and our son through this demanding phase; My mother Gayatri Nath, who has cheerfully made many sacrifices to help me succeed, for the time and help she offered during the simulation phase of this research; My father Madhur Nath for always being there and helping out with proofreading my thesis;

My son Narayan for giving me a reason to wake up every morning after late nights spent on research; My son Navneet for being the happiest baby I have ever known. They mean the world to me.

TABLE OF CONTENTS

TITLE PAGE

ABSTRACT i

DECLARATION iii

ACKNOWLEDGEMENTS iv

TABLE OF CONTENTS vi

LIST OF FIGURES viii

LIST OF TABLES xi

LIST OF ABBREVIATIONS xii

NOMENCLATURE xiii

1 INTRODUCTION 1

1.1 RESEARCH BACKGROUND AND MOTIVATION 1
1.2 GAPS IN EXISTING RESEARCH 5
1.3 RESEARCH PROBLEM STATEMENT 10
1.4 AIMS AND OBJECTIVES OF THE STUDY 10
1.5 RESEARCH SCOPE 11
1.6 THESIS ORGANIZATION 12

2 LITERATURE REVIEW 14

2.1 ROBOT APPLICATIONS IN AGRICULTURAL AREAS 14
2.2 ROBOT LOCOMOTION 15
2.3 ROBOT NAVIGATION 17
2.3.1 Complete Area Coverage Algorithms 18
2.3.2 Other algorithms for navigation 25
2.3.3 Obstacle Avoidance 27
2.3.4 Localisation 35
2.4 ROBOT SENSORS FOR OBSTACLE AVOIDANCE AND DETECTION 38
2.5 CONCLUSION 43

3 DEVELOPMENT OF A NEW HYBRID BUG ALGORITHM FOR OBSTACLE AVOIDANCE DURING AREA COVERAGE 45

3.1 INTRODUCTION 45
3.2 PRELIMINARIES 45
3.2.1 Bug2 algorithm 46
3.2.2 TangentBug Algorithm 49
3.3 DESCRIPTION OF THE NEW HYBRID BUG ALGORITHM 53
3.4 ANALYSIS OF THE NEW HYBRID BUG ALGORITHM 63

3.5 CONCLUSION 77
4 ROBOT SENSORS AND NAVIGATION 79
4.1 INTRODUCTION 79
4.1.1 Constraints posed by the ultrasonic sensor, SRF05 81
4.1.2 Analysis of existing methods to improve angular resolution of the SRF05 sensor 82
4.1.3 Examining the applicability of the existing techniques to improve angular resolution in the context of the Hybrid Bug algorithm 84
4.2 BOUSTROPHEDON AREA COVERAGE 85
4.3 DEVELOPMENT OF A NEW METHOD FOR THE DETECTION OF OBSTACLE BOUNDARIES USING LOW RESOLUTION SENSORS 88
4.4 AN INTEGRATED APPROACH TO BOUSTROPHEDON AREA COVERAGE INCORPORATING THE HYBRID BUG ALGORITHM WITH THE ENHANCED OBSTACLE BOUNDARY DETECTION METHOD TO ENABLE BOUNDARY FOLLOWING 92
4.5 CONCLUSION 98
5 SIMULATION 100
5.1 INTRODUCTION 100
5.1.1 Modelling for the simulation study 100
5.2 RESULTS AND DISCUSSIONS 102
5.2.1 Simple Hybrid Bug algorithm 102
5.2.2 Hybrid Bug Algorithm with simulated ultrasonic sensors and boustrophedon motion 105
5.3 CONCLUSION 119
6 CONCLUSION 121
6.1 MAIN CONTRIBUTION OF THIS RESEARCH 121
6.2 FUTURE WORK 122
APPENDIX I. BOUSTROPHEDON TRACK WIDTH FOR SIMULATIONS 125
APPENDIX II. TYPICAL OBSTACLES ON A FARM 127
APPENDIX III. DEVELOPMENT OF A LOW COST ROBOTIC PLATFORM 128
APPENDIX IV. ADMISSIBILITY CRITERION FOR A MODIFIED LTG EDGE 132
APPENDIX V. SUBSET OF THE SIMULATION RESULTS OBTAINED 134
REFERENCES 138

LIST OF FIGURES

Figure 1.1 A coverage of an unknown area as proposed in [27] 7
Figure 1.2 An example of complete area coverage of a farm 11
Figure 2.1 Power consumption of several locomotion mechanisms [56] 16
Figure 2.2 Effect of inlets on Boustrophedon coverage [27] 19
Figure 2.3 An example of the offline STC algorithm [30] 20
Figure 2.4 An example of the BA* algorithm using the A* algorithm to backtrack 21
Figure 2.5 Area coverage using wavefront propagation [65]. 22
Figure 2.6 Creating the diameter function by rolling the polygon. The dotted line represents the chord used to determine the diameter or altitude of the polygon [31] 24
Figure 2.7 Scenario where bug0 algorithm fails [83] 28
Figure 2.8 Bug1 path planning [83] 28
Figure 2.9 Bug2 path planning [85] 29
Figure 2.10 Comparison of the paths generated by the VisBug-21 and the VisBug-22 algorithms [87] 31
Figure 2.11 Paths generated by the VisBug-21 and the VisBug-22 algorithms are similar to the Bug2 algorithm when sensor range is smaller than in Figure 2.10 [87] 31
Figure 2.12 The Visibility Graph (a) and Tangent Graph (b) [86] 33
Figure 2.13 Obstacles modelled as thin walls in the LTG and the special node T_{node} [86] 33
Figure 2.14 A demonstration of the TangentBug algorithm with the sensor range represented by a dotted circle [56]. 34
Figure 2.15 Idealized Radio Model [72] 36
Figure 2.16 Comparison of the Gyro-Encoder Hybrid model with the encoder only model [79] 37
Figure 2.17 Beam pattern of the SRF05 [89] 38
Figure 2.18 The centreline model can make a narrow passage seem narrower [42] 39
Figure 2.19 The McKerrow method of determining a common tangent through two sonar arcs [42] 40
Figure 2.20 Single cone intersections can give misleading results [42] 41
Figure 2.21 Transversal and non-transversal intersections [42] 41
Figure 2.22 Performance of the SIFT algorithm [96] 42
Figure 2.23 Results of the EDISON system [95] 43
Figure 3.1 Robot's path (dashed line) under Bug2 [85] 47
Figure 3.2 An example of a local cycle in a maze like obstacle [85] 47
Figure 3.3 Examples of traps [85] 49
Figure 3.4 A path generated by the TangentBug algorithm, image modified from [86] 51
Figure 3.5 The robot at *x* has no straight line approach to the m-line. 56
Figure 3.6 An illustration of the subgraph G1 57
Figure 3.7 Image illustrating the concept of an occlusion node [86] 59
Figure 3.8 A flowchart summarising the Hybrid Bug algorithm 61
Figure 3.9 Path generated by the Hybrid Bug algorithm 62

Figure 3.10 Illustration of Type 1 and Type 2 events (Illustration created based on examples in [86]) 66
Figure 3.11 Illustration of a Type 3 event (Illustration created based on examples in[86]) 67
Figure 3.12 NoType 2 events occur when (a) LTG nodes remain fixed (b) occlusion nodes slide along the obstacle boundary [86] 71
Figure 3.13 Extended edges partitioning the free space into cells 73
Figure 3.14 Robot moving (a) within a cell (b) along blocking extended edges [86] 74
Figure 4.1 Shadow regions with and without overlapping coverage sections 80
Figure 4.2 Effect of the narrow beam pattern in the SRF235 [101] 81
Figure 4.3 Beam pattern of SRF05 [89] 82
Figure 4.4 Beam splitting [40] 82
Figure 4.5 A situation in which McKerrow's method can give erroneous results 83
Figure 4.6 Optimised obstacle circumvention in a boustrophedon scenario versus a depth first scenario 86
Figure 4.7 Large Obstacle in the course of a boustrophedon coverage 87
Figure 4.8 Effect of algorithms based on backtracking when obstacles are small in comparison to the navigable area 87
Figure 4.9 Sensor arcs of 3 adjacent sensors in different colours 89
Figure 4.10 Tangents to a circle from an external point P 89
Figure 4.11 Flowchart representation of the boustrophedon area coverage algorithm 96
Figure 4.12 Flowchart representation of the boustrophedon area coverage algorithm (continued) 97
Figure 4.13 Flowchart representation of the boundary following module 98
Figure 5.1 The detection zone along the XZ plane at y = -10 [88] 101
Figure 5.2 The Bug2 (Left) follows a longer path to the m-line than the Hybrid Bug (Right) 103
Figure 5.3 The Bug2 (Left) follows a longer path to the m-line than the Hybrid Bug (Right) due to a fixed direction of circumvention 103
Figure 5.4 The TangentBug (Left) covers less of the m-line than the Hybrid Bug (Right) 104
Figure 5.5 Illustration of the three bug algorithms on an obstacle with a maze like cavity 104
Figure 5.6: Typical obstacle shapes used in the simulations 107
Figure 5.7 : Area coverage by the BA* algorithm 108
Figure 5.8 The same area as in Figure 5.6 covered by the boustrophedon area coverage using the Hybrid Bug algorithm algorithm 109
Figure 5.9 Area coverage of a second obstacle course by the BA* algorithm 109
Figure 5.10 The same area as in Figure 5.6 covered by the boustrophedon area coverage using the Hybrid Bug algorithm algorithm 110
Figure 5.11 An example of BA* coverage of areas with large obstacles 111
Figure 5.12 The same area as in Figure 5.10 covered by the boustrophedon area coverage using the Hybrid Bug algorithm algorithm 111
Figure 5.13 A narrow passage between two obstacles 112
Figure 5.14 Navigation of corners using collinear points on the new face A 114
Figure 5.15 A section of the m-line is missed in between the 2 obstacles since the larger distance to the obstacles makes them appear large enough to block the passage. 115

Figure 5.16 m-line coverage is maximised by the robot greedily approaching the m-line in shorter segments along the normal. 116
Figure 5.17 Boustrophedon path generated by the boustrophedon area coverage using the Hybrid Bug algorithm when the target of the first track is blocked by an obstacle 117
Figure 5.18 Boustrophedon path generated by the boustrophedon area coverage using the Hybrid Bug algorithm when the start of the second track is blocked by an obstacle..... 117
Figure 5.19 Performance of the boustrophedon area coverage using the Hybrid Bug algorithm algorithm in U-trap situations 118
Figure 6.1 Clustered obstacles perceived as one – encircled in red..... 123

LIST OF TABLES

Table 5.1 A table indicating average path lengths and m-line coverage values by different bug algorithms 105
Table 5.2 Comparison of the path length generated by the Boustrophedon area coverage using the Hybrid Bug algorithm and the BA* algorithm 107
Table 5.3 Comparison of the m-line coverage of the Boustrophedon area coverage using the Hybrid Bug algorithm and the BA* algorithm 108
Table 5.4 Improvement of turn radius during boundary following behaviour by detecting the new obstacle face using collinear points 114

LIST OF ABBREVIATIONS

UAV	Unmanned Arial Vehicles
AUV	Autonomous Underwater Vehicle
ROV	Remotely Operated Vehicles
SET	Smart Encoder Trailer
MDOF	Multi Degree of Freedom
SLAM	Simultaneous Localisation And Mapping
MVS	Machine Vision System
LIDAR	Light Detecting and Ranging
RADAR	Radio Detection and Ranging
SONAR	Sound Detection and Ranging
PSO	Particle swarm optimization
SVM	Support Vector Machines
ANN	Artificial Neural Network
PE	Processing Element
PID	Proportion Integral Device
MA	Mobile Automaton
STC	Spanning tree coverage
DFS	Depth first search
LTG	Local Tangent Graph
IMU	Inertial Measurement Unit
NAN	Not a number

NOMENCLATURE

Navigable area	Area to be covered by the robot
C Traps	Deeply concave obstacles. Autonomous robots navigating with the help of several conventional obstacle avoidance algorithms often find it hard to navigate out of the cavity once inside.
Start	The start position of the robot during area coverage or obstacle avoidance.
Target	The start position of the robot during area coverage or obstacle avoidance.
Localisation	Deducing the position and orientation (pose) of a robot within its environment
O	The big O notation for time complexity used in the design and analysis of algorithms.
Offline algorithms	A set of path planning algorithms that are aware of the obstacles on a course and plan an optimal path before the robot navigation begins.
Online algorithms	A set of path planning algorithms that are not aware of the obstacles on a course and plan a sub optimal path as the robot navigates based on sensor information.
Azimuth resolution	Angular resolution of a range sensor.

1 Introduction

Over the years robots have been used in various conditions and to serve many purposes, mainly to automate activities that demanded rigorous effort. Agriculture being a very labour intensive industry took keen interest in employing robotics right from the early 1960s. Early applications of robotics in agriculture mainly saw them being used to harvest crops using brute force such as shaking the trunks of trees [8]. These would cause significant damage to the products. Today, sophisticated technology allows fruits and vegetables to be harvested with minimal damage [103]. Robots are also becoming popular in the livestock industry. However, even in the present day, many robots in the agriculture industry need to be controlled by human operators. Additionally their work needs to be supplemented by people who perform the various actions that the robots cannot, such as sorting through the harvested fruits to remove spoilt/damaged produce. Autonomous robots work independently to perform entire tasks without human effort. Despite the first autonomous robots making their early appearances in the late 1940s [7], presently they are more widely used in controlled environments rather than in agricultural environments where the number of variable factors are many and lead to extremely complex decision making processes.

This chapter briefly explores the existing autonomous robots employed in the agricultural sector and serves as an introduction to the thesis by discussing the factors that necessitate this research. The research problem and scope are clearly specified, followed by an outline of the thesis.

1.1 Research Background and Motivation

Mobile robots are used in a wide range of agricultural applications. Snippy, the Grapevine Pruner [9] designed by Vision Robotics Vineyard Products can prune 400 acres of grape vines every season. There are robots to milk cows and clean barns. The fact that robots today can work successfully with minimal human intervention (such as Google's autonomous car [10]) coupled with increased lobbying for humane treatment of animals has created the need for an autonomous field robot that can monitor the welfare of livestock. The efforts to reduce the number of battery cages in poultry farming practice since 1999 saw more than 250 million hens benefited by free range or barn raised farming [12]. The European Union imposed a complete ban on battery cages as of January 2012 [11]. This sudden surge of free range and barn raised poultry farms due to the new policies has increased the work load on

farmers and created the need for autonomous monitoring of the welfare of chicken on such farms by robots.

The rapid decline in the cost of sophisticated technology and the extensive research involving application of technology in agriculture make a compelling argument to develop robotic farm hands to supplement the farmers' tasks. Most agricultural machinery such as tractors and vine pruners are manually operated. On account of major advances in microcontroller technology in recent times, better control systems can be built for agricultural robots. Research done in image processing also suggests that machines are able to detect objects of interest more reliably.

Currently, autonomous mobile robots are commercially unavailable for monitoring free range poultry farms. Since autonomous robots can emulate the functioning of a human counterpart on the poultry farm and the technological advancement allows for the development of such a system, this is a ripe opportunity for research. From the literature review it was found that there were extremely few researchers catering to the development of autonomous robots in the poultry farming agricultural sector. However, with a strong push towards free range poultry farms in Europe since 1999 and the subsequent ban in battery cages [11], the research towards this sector saw an impetus, particularly in the Netherlands and Australia. Free range poultry farms allow hens to roam freely. However, farms often have nesting boxes for the hens to enter and lay eggs in private - in partially covered cubicles. The eggs are not always laid in the boxes, as the hens normally take about 14 -15 days to learn to use the nesting boxes. Until such time, the eggs are laid at random locations, which are prone to trampling and impact damage by the hens, humans and equipment. This causes an avoidable loss and necessitates frequent cleaning, resulting in revenue loss for the farmer. An autonomous field robot, can effectively address this problem, by plugging the revenue loss, thus making it an extremely farm friendly piece of equipment.

Water is involved in every aspect of poultry metabolism. It plays important roles in regulating body temperature, digesting food, and eliminating body wastes. At normal temperatures, poultry consume at least twice as much water as feed. When heat stress occurs, water consumption will double or quadruple. A safe and adequate supply of water is therefore essential for efficient poultry production [13]. The characteristics of safe, good-quality drinking water for poultry can be evaluated in a number of ways. It can be tested for the presence of bacteria and other microbes, for the levels of minerals and for other chemical and

physical factors. If an autonomous mobile robot were available, it could be designed to test the quality and quantity of water at various preconfigured stations and send a wireless alert to the owner if water levels are found to be inadequate or contamination is detected using on board sensor technology such as the SPIAS sensor which can detect pathogens and costs less than $20 [14]. This can increase the longevity of the livestock.

Feeding makes up the major cost of production of poultry and good and nutritious feed is reflected in the birds' produce almost immediately. The most convenient way of feeding chickens is with a balanced pelleted ration, whether the birds are kept indoors or allowed to range outdoors. Most diets contain corn for energy, soybean meal for protein, and vitamin and mineral supplements. Commercial rations often contain antibiotics and arsenicals to promote health and improve growth, of the birds. In the industry, the feed is pelleted so the bird can eat more at one time. Chickens are nibblers and make frequent trips to the feed for small meals, which requires energy. Pelleting reduces the amount of energy required for a bird to feed [15]. An autonomous robot can be equipped with sensors to check the quality of feed and ensure adequate supply at all times and report contamination if detected. This ensures lowest mortality rates for the birds.

Noxious gases in poultry houses are generated as product of animal metabolism and as animal wastes break down. These gases can produce oxygen-deficient, toxic or dangerous atmospheres. The most common noxious gas, which is produced in poultry houses, is ammonia. The effect of this gas can even increase the mortality rate in broiler and laying hen houses where the litter bedding system is used under insufficient ventilation in hot climates. Ammonia gas in poultry houses seriously affects the health condition of the birds. Good litter management and ventilation will minimise the level of ammonia which can improve productivity, reduce the likelihood of respiratory diseases, improve the birds' welfare and provide a pleasant, safe environment for workers. In addition, carbon dioxide (CO_2), nitrous oxide (N_2O) and methane (CH_4) gases are released [16]. An autonomous robot through forward integration can be made to sense these toxic and dangerous gases through sensors and more advanced technology such as electronic noses [17]. A suitable alert can be sent if levels of toxic gasses cross allowable limits thus ensuring good health of the birds and of the workers in the farm.

When chickens in a poultry farm encounter a roving machine, they also tend to move away, which enables an autonomous robot to provide a gentle herding function as well. This keeps the chickens well exercised and healthy.

Additionally, despite added welfare mechanisms, in the event of death of a hen in the poultry farm, an on board vision system trained to identify dead birds can be employed and the robot can then gently remove them so that any possible contamination is prevented.

Note that all of this monitoring will be most effective if the robot performing these tasks is capable of covering the area of the poultry farm as close to its entirety as possible. In general, an autonomous field robot for such applications should be equipped with the following modules factored into its design:

- Hardware design - To develop a low cost autonomous robot and enable picking up of eggs and dead chicken.
- Area Coverage – To cover the entire area of the farm providing a gentle herding function to keep the chicken active and also scout for eggs.
- Obstacle avoidance – To avoid obstacles that the robot does not know of a priori, such as trees, feeding and watering stations, chicken and humans in open and under cover facilities.
- Image analysis and pattern recognition – To recognize the eggs and chicken and possibly unexpected obstacles in a dynamic environment.

While the development of the entire system would be beyond the scope of this research, algorithms for area coverage and obstacle avoidance modules have been developed. The complete coverage of an area is essential to make certain that dead chickens on the farm are not missed, thus controlling contamination of the farm. Additionally, complete coverage ensures that the livestock is continuously herded, all the feeding and watering stations are serviced and no areas of the farm have a build-up of unwanted gasses or waste.

The challenge of complete area coverage of unknown environments has been a keenly researched subject. The focus of most complete area coverage research has been for use indoors to allow robots such as autonomous vacuum cleaners to cover indoor areas. However, obstacle shapes, density and distribution is vastly different in indoor environments such as homes and office spaces when compared to agricultural farms. For example, algorithms for

area coverage on indoor robots need to be able to deal with maze like obstacles such as walls, which are unlikely to occur outdoors. The computational effort and resources that focus on such features can be utilized elsewhere. Additionally factors such as interference from direct sunlight and unavailability of GPS come into play outdoors affecting the choice of sensors. Popular algorithms that work on covering areas where the shape, positioning and density of obstacles are unknown operate by decomposing the area to be covered into cells and applying graph traversal algorithms on the collection of cells. These suffer from the problems that come with approximating the area to be covered and are discussed in detail in section 2.3. Other algorithms use intensive approaches to identify and map obstacle boundaries and are not resilient to changing obstacle positions. These are some of the factors that necessitated the development of a bespoke efficient area coverage algorithm.

The algorithms for area coverage and obstacle avoidance developed in this thesis cater to free range and barn raised poultry farms, using low cost sensors which are easily available. Developed without loss of generalisation, these algorithms can be applied to other farms with similar requirements as well.

1.2 Gaps in Existing Research

The field of area coverage has been extensively researched with various algorithms addressing *online* and *offline* versions of the problem. An online area coverage algorithm has no prior knowledge of the obstacles in its environment and will have to rely on local sensor information. An offline area coverage algorithm is completely aware of the obstacles in its environment and therefore can make path planning decisions based on globally available data. As this research aims at exploring areas where the dimensions, placement, size and shape of the obstacles are unknown, the online navigation and coverage algorithms are of greater interest. Further, it can be argued that after one pass, obstacles can be mapped and offline algorithms can be applied. However, if a mapping process is implemented, chicken and human movement can cause sensor interference and *ghost obstacles* to be mapped onto the list of permanent obstacles, to avoid which a wrong path may be planned by the offline algorithms. Therefore, online algorithms are considered within the scope of this research.

Several approaches have been developed to handle the task of area coverage. Howie Choset discusses recent developments in the field of area coverage [21]. Additionally, a few algorithms are based on heuristics and randomized components such as those developed by Balch and Arkin in [22, 23], where a team of robots comprising from 1 to 5 are made to

search a space and bring objects back to the home-base and extended this to include multi-agent vacuuming tasks. Heuristics such as obstacle avoidance and repulsion from other robots define overall robot behaviour which can prove to be efficient in multi-agent systems. The arguments presented for randomised search based coverage algorithms are:

1. They do not require accurate localisation which could be computationally expensive or require additional sensors such as GPS [24]

2. In contexts where the probability of having a job done is less than 1 even with complete coverage due to sensor limitations, and multiple robotic agents are available to be used cooperatively, a case has been formed that with diminishing probability of systematic area coverage being effective, using random search techniques with multiple robotic agents can be more advantageous than methodical searches [25].

The key disadvantages of using randomised heuristic algorithms in the context of this research are:

1. These algorithms do not guarantee complete coverage

2. They can be slow due to the randomness involved and can be inefficient when the number of agents decreases to 1(Assumption 7, section 1.5).

3. The probability of accomplishing the task in one pass of the robot is 1 (Assumption 6, section 1.5). Therefore, the case of efficacy of a random search cannot be established.

A template based heuristic area coverage method is outlined in [26], predominantly an offline area coverage algorithm which uses predefined paths to cover areas between obstacles and also deals with unexpected obstacles. While results are satisfactory, the paper does not address how

1. The algorithm will cope when obstacles are removed.

2. Obstacle contours are followed using low cost ultrasonic sensors.

Hert, Tiwari and Lumelski [27] propose a completely online approach to perform area coverage of a terrain by a robot moving in 3 dimensional space such as a submarine mapping the ocean floor. This algorithm is an attractive option for area coverage when the area to be covered is not bounded. However, in a case where the bounds of the area to be covered are known, the attempts of this algorithm to identify and segregate *islands*(obstacles) from *inlets*

to the area to be covered can unnecessarily increase path length. This is because identifying these contours leads to multiple skirting of the obstacle boundaries, as shown in Figure 1.1 which can add wasteful cycles to the robot's run length and therefore decrease efficiency. Additionally, if used on a farm where obstacles such as tractors can move, the *inlet* and *island* identification process can fail.

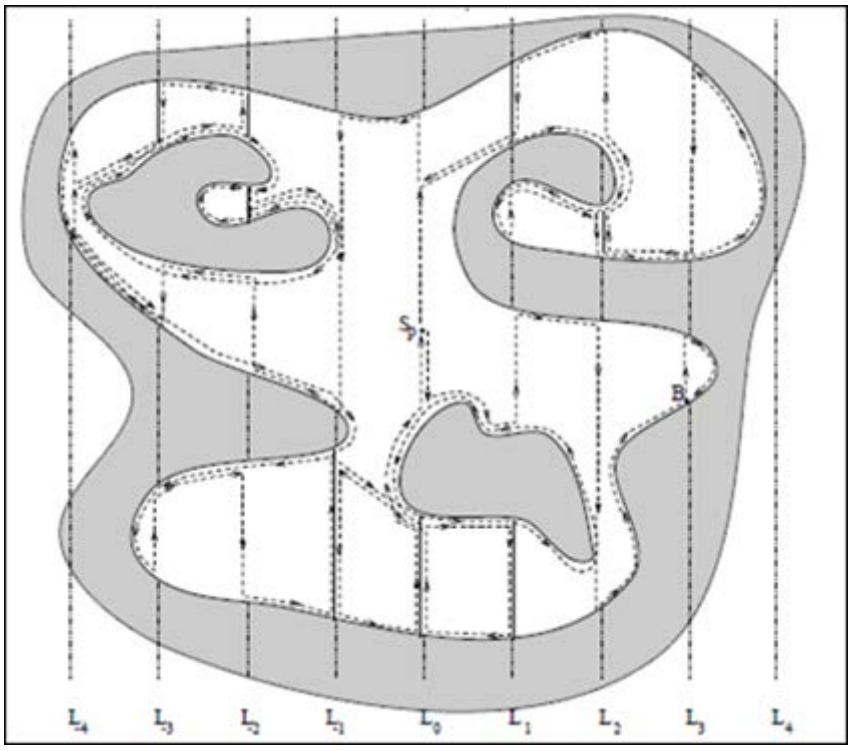


Figure 1.1 A coverage of an unknown area as proposed in [27]

Other online coverage algorithms mostly work on breaking up the area into sub-areas. The Boustropheon decomposition [28] uses a vertical line sweeping across the area to be covered, splitting it into sub-sections when there is a loss of continuity in the line due to the presence of an obstacle. The point at which continuity is lost is called a *critical point*. The subsections are covered in a boustrophedon pattern (boustrophedon means "the way of the ox" and involves parallel tracks separated by a constant width). The next subsection to be covered is decided in a depth first manner, where neighbouring cells are evaluated in an anticlockwise manner. If none of the neighbouring cells are available to cover, the algorithm calls for backtracking to uncovered cells. Primarily offline, this algorithm was made online by the work done in [29]. The procedure that identifies critical points using sensor information makes loops each time an expected closing critical point is missed. This behaviour along with the backtracking behaviour of the coverage algorithm could prove expensive.

Approximate cellular decomposition algorithms such as the Spanning Tree Coverage algorithm (STC)[30] and Boustrophedon A* (BA*) [33] algorithms divide the area into cells, typically based on the robot's sensor footprint. They then proceed to visit each cell, with the aim that visiting every cell will give a complete coverage pattern. The optimisation of the path taken to visit the cells is a version of the travelling salesman problem termed the *covering salesman problem*. The STC algorithm polls each neighbouring cell in an anticlockwise order to see if a cell is unoccupied by obstacles and has not been previously visited. It moves to the first *free cell*, creating a depth first approach to area coverage. The BA* algorithm results in a boustrophedon area coverage by polling the neighbours in a North - South - East –West order. In both these algorithms, when a cell without free neighbours is encountered, the algorithm calls for backtracking to a cell with free neighbours. These algorithms are approximate as the aggregation of unoccupied cells is expected to represent the whole free space to be covered. They provide effective online area coverage but suffer from the overhead of backtracking and depending on the size of the cells, incomplete area coverage.

In essence, while several options are available to perform coverage operations, there are no algorithms that are specifically tailored to optimise coverage of agricultural areas such as free range poultry farms, where the area to be covered is large and the obstacles are small in comparison and spread out. The problem with using algorithms that require backtracking to cover vast areas with small obstacles is that often one run of the algorithm can take it to great distances from the point at which backtracking needs to be performed. This means that the backtracking operation involves significant time and effort. Other online coverage algorithms often make multiple passes around the boundaries of obstacles and this leads to cumulative effort overrun.

This research prioritises the use of low cost sensors for navigation. This is because sensors can have an effect on the cost of the entire system, especially if multiple sensors are used to get a 360° coverage. It is desirable to have a low cost autonomous agricultural robot that could be easily afforded by small businesses and on smaller farms. Additionally, if used on larger farms, imposing a cost constraint enables applications in scalable robotic systems with multiple co-operative robots working on the same area. Few algorithms use ultrasonic sensors to navigate and those that do, do not discuss how effectively the ultrasonic sensors can follow obstacles and at what distances the obstacle following behaviour is implemented. Low cost ultrasonic sensors suffer from poor angular resolution due to the inherent beam pattern. For

example, the SRF05 sensor used in this research has an approximate angular resolution of 55° [89]. Reducing receiver gain can enable the focussing of the perceived sonar beam and therefore increase angular resolution [40]. This will necessitate the requirement for more sensors. However, low cost ultrasonic sensors such as the SRF05 do not expose their receiver circuitry. Techniques of beam splitting where sensor beams are overlapped [40] are effective but insufficient and will be considered to partially fulfil the need for improved angular resolution. Existing methods to cope with low resolution include models such as the centreline model [42], McKerrow's common tangent method [41] and Arc Transversal Median (ATM) [42] methods. The centreline model is quite useful when sonar arcs are small. Since the arc length is dependent on the distance from the obstacle, this model can be used when following obstacles closely; however, if obstacles are followed very closely and their contours are followed in a bug like fashion, the use of having range sensing is not leveraged. At larger distances, the centreline model can cause passages between obstacles to appear narrower than expected. McKerrow's common tangent method [41] is based on finding the orientation of the face of an obstacle, by passing a tangent through two sonar arcs from the same sensor at different points along the robots movement along the face of the obstacle. The orientation of the tangent gives an idea of the orientation of the face of the obstacle. This method does not work when there is a corner or change in the orientation of the obstacle face. Also, since it builds upon historic sensor data, it does not give an indication of what is to come in terms of the obstacle's contours. The ATM method [42] on the other hand is more adept at locating corners as it primarily works on intersections of sonar arcs. Due to this very nature, it works better on specular surfaces rather than smooth surfaces. This method also works off historic data and is not predictive in nature. It would be advantageous if, in addition to an understanding of the current orientation of the obstacle face, the low angular resolution ultrasonic sensors could have an idea of the changes in the obstacle face being followed, especially when turning corners and do so at a decent distance from the obstacle. If the robot follows the face of the obstacle very closely, it would result in the following of unnecessary concave contours of obstacles. This gap is also addressed in the formulated navigation strategy.

In summary, among the many gaps that need significant research for applications in autonomous agricultural robots that navigate using low cost sensors, the two gaps addressed by this research lay the foundation for navigation. They are:

- Development of a whole area coverage algorithm with path planning and obstacle avoidance modules that enable maximal coverage at shorter path lengths.
- Development of methods to overcome poor angular resolution of navigation sensors to achieve maximal area coverage with the aid of low cost sensors.

1.3 Research Problem Statement

The complete coverage of areas is the fundamental task that an autonomous robot would need to perform in order to assist with agricultural activities. If the landscape of the area to be covered is likely to change, which is the case in agricultural areas, especially those involving livestock, an online algorithm that has no idea of the position, shape, size or orientation of the obstacle must guide the process of area coverage. In such a scenario, existing algorithms that involve closely following obstacle contours in order to map them will not only waste effort in the mapping process but also be inflexible to changes in obstacle positioning. Other area coverage algorithms tend to form approximate cellular decompositions of work spaces compromising on the completeness of area coverage. Additionally, there is insufficient literature on area coverage using low cost sensors for outdoor navigation. These factors prompt the research into the development of a custom algorithm designed to maximise coverage of outdoor areas comprising of unknown obstacles whose positioning can change over time, with the use of low cost navigation sensors.

Therefore, the main research problem of this project is “Finding a suitable strategy for the autonomous navigation of a robot in agricultural applications requiring complete coverage of large areas with small obstacles whose distribution, position and boundaries are unknown”.

1.4 Aims and Objectives of the Study

The main aim of the study is to analyse and develop a strategy for the complete area coverage of an unknown environment using a low cost mobile platform, typically for use in free range poultry farms where navigable area is large and obstacles are relatively small. The robot must be capable of operating indoors in barns as well as outdoors. Figure 1.2 shows an example of a complete coverage of a simplified free range poultry farm where the rectangular obstacle represents a feeding station and the pentagonal obstacle represents a watering station. The pattern of area coverage shown in the figure is just an illustration and the aim of this study is also to evaluate many of the existing area coverage and obstacle avoidance algorithms.

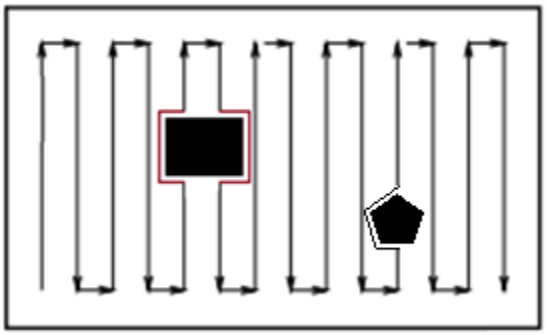


Figure 1.2 An example of complete area coverage of a farm

The constraints imposed on the mobile robotic platform are that it should be cost effective and work indoors and outdoors. These constraints influence the sensors chosen for navigation and subsequently have ramifications in the navigation strategies. It would also be highly advantageous to employ a commonly used low cost, easily sourced sensor for navigation if they need replacement during the life of the robot. Such sensors often suffer from poor angular resolution. Overcoming the drawbacks of low cost sensors in order to provide effective area coverage is another important issue in this research.

1.5 Research Scope

The following assumptions define the scope of the research:

1. The boundary of the area to be covered is a regular convex polygon whose dimensions are known.

2. The dimensions, placement, size and shape of the obstacles are unknown.

3. The stationary obstacles are polygonal, modelled after obstacles on a poultry farm and are typically less than 3m wide. (Appendix II). The obstacles are typically not clustered together. The area coverage algorithm must deliver optimal area coverage for such obstacle dimensions and distributions.

4. Moving obstacles such as chickens are expected to make way for the robot and modelling them in simulations is out of the scope of this research. Also, in the scope of this research, any moving obstacles are treated as stationary when detected by the sensors. When it is detected that the obstacle has moved away the planned path the algorithm is expected to use this movement of the obstacles opportunistically to maximise area coverage.

5. The analysis of the problem focusses on avoiding static obstacles. It is assumed that mobile obstacles will move out of the robots path as the robot approaches them.

6. The current research assumes ideal localisation conditions. It is proposed that dead reckoning errors and wheel slippage issues will be addressed and offset using inertial navigation systems as part of future research.

7. The complete area coverage will result in the complete probability of the intended task being performed. For example, if the intended purpose of the robot is to inspect an area for toxic gasses, the probability that the entire area will be inspected in one pass of the robot is 1 or close to it.

8. The area coverage operation is to be carried out by a single robot at the level of this research. However, it can be easily scaled up to include multiple robots.

The proposed navigation and obstacle avoidance systems described in the thesis idealize the robot to be a holonomic point mobile automaton.

1.6 Thesis Organization

The first chapter of this thesis presents an introduction to the entire thesis, outlining the factors that prompted this research on complete coverage of agricultural areas. The description of the problem statement and a succinct specification of the scope allow for clarity on the expectation from this thesis. While the first chapter also discusses the possible target application of the algorithm developed in this thesis, the second chapter on literature review delves deeper into the existing literature on the technical aspects of mobile robots, sensors and area coverage algorithms and discusses some of the reasons why the existing literature cannot provide a solution to the research problem statement. The third chapter proposes and analyses the new obstacle avoidance algorithm used by the area coverage strategy developed in this research. The preliminary algorithms influencing this obstacle avoidance algorithm are discussed and the robustness and efficiency of the new algorithm is verified analytically. This chapter presents and analyses the algorithm for the ideal case of unlimited sensor range and infinite sensor angular resolution. Chapter 4 discusses the complete area coverage algorithm and the efficacy of low cost sensors in the context of obstacle avoidance using the newly developed algorithm. The limitations on the range and angular resolution of the sensors in the practical implementation of the algorithm are analysed. A method to overcome the issues arising from the use of low cost sensors is

proposed and an integrated approach that incorporates the new obstacle avoidance algorithm and sensor improvement technique within the complete area coverage algorithm is proposed. Chapter 5 presents the results of simulating this integrated algorithm. A comparison study featuring a contemporary online algorithm is also presented and verifies the superior performance of the newly developed algorithm. Chapter 6 serves as a conclusion and discusses future work.

The Appendix provides the supporting details such as the specifications of the prototype built to gain an understanding of the robotic platform and on-board sensors as well as proofs to criteria that support the analytical verification of the robustness of the new obstacle avoidance algorithm.

2 Literature Review

This chapter gives an overview of existing literature covering the various aspects of this research. Starting with an overview of the application of robotics in agricultural areas, the review then explores the various means of robot locomotion. The choice of wheeled locomotion follows from literature review which then helps direct the review on area coverage. Area coverage forms the main focus of this research and existing research is covered in depth. Coverage by multi-robot systems is also presented as an extension to the scope. Research gaps pertaining to the field of area coverage, which has been discussed in section 1.2, is elaborated in this chapter. Finally, the sensors that aid navigation are reviewed and literature on robot vision systems is presented as possible extensions to the work done in the current research.

2.1 Robot applications in agricultural areas

Today, technological advancements in robotics are beginning to be applied in agricultural areas such as animal husbandry, soil management, fruit picking and harvesting and dairy and poultry farming. Robot chicken feeders, robotic milking devices and automatic sheep shearers are some examples of robots used in animal husbandry. The automatic milking system has been in use since 1907 [43] while sheep shearing robots [44] were experimented in Western Australia between 1985 and 1993 and often proved more humane as they inflicted fewer cuts and nicks when shearing than the human counterparts due to faster reaction times. These are immobile robots that are used to perform one function only. Barn cleaning robots are beginning to be used in cattle sheds [45], although their performances warrant improvement [45]. In addition to cutting down on human labour, the use of robots in animal husbandry proves to increase levels of hygiene.

The use of robots in the harvesting industry is rapidly picking up pace. In countries like Australia, a majority of fruit harvesting is still done by human labour, often involving complicated processes such as climbing ladders/trees. Since labour is in short supply and expensive, the general climate of the agricultural industry here encourages automated fruit harvesting. Sanders K.F. [46] suggested that efficient mechanical harvesting systems reduce harvesting costs by about 35%-40%. Schertz and Brown initially considered automatic individual harvesters for oranges [48], which were operated with a vacuum twist mechanism or a rotating blade mechanism. Fruits were identified using photometric comparisons since the light reflected off oranges that were around the 680nm range was 10 times more than the

light reflected off leaves. They used a primitive form of thresholding. Montpellier France [49] devised the earliest form of the modern day automatic fruit harvester. This consisted of a black and white camera mounted on the machine with a telescopic arm extending horizontally to grab the fruit detected by the camera. This design was improved in [50, 51] with the use of a colour camera and a proximity sensor in the grasper to sense the touch of the fruit. Additionally, an arm with 3 degrees of freedom and suction cup with end effectors replaced the telescopic arm. About 50% of the fruit were collected and harvested. Spanish and French teams collaborated on the Eureka project [52, 53] to develop a colour image processing system based on a Bayesian Classifier as a discriminating function and achieved 90% of global detection rates.

Additionally, autonomous sprayers have been researched by Tillett and Hague to selectively spray weedicides on individual weeds [54], while Vision Robotics has been working on a range of robots and has successfully developed autonomous pruning systems that can use their vision systems to prune up to 400 acres of grape vines every season [55].

2.2 Robot Locomotion

While aerial or ground locomotion can be considered for the target application of monitoring chicken welfare in free range farms, the nature of the activities such as picking up dead chickens or eggs, cleaning, providing gentle herding and monitoring food and water calls for robots working closer to the ground. Additionally, since the requirement is for the robot to be able to carry a significant payload of about 20 kilograms (Appendix III), using aerial locomotion can be ruled out, given that hauling the payload would be very energy intensive.

The locomotion mechanism of mobile on-ground robots is largely based on the terrain they are designed to traverse. Several factors such as power expenditure, stability, inclination of the terrain can contribute towards choosing a locomotion mechanism. Choset et al. [56] explained that the choice of a locomotion mechanism determines the amount of energy expended. As expected, on a terrain where the amount of friction is high, locomotion mechanisms with fewer points of contact offer most efficiency in terms of conserving energy (see Figure 2.1).

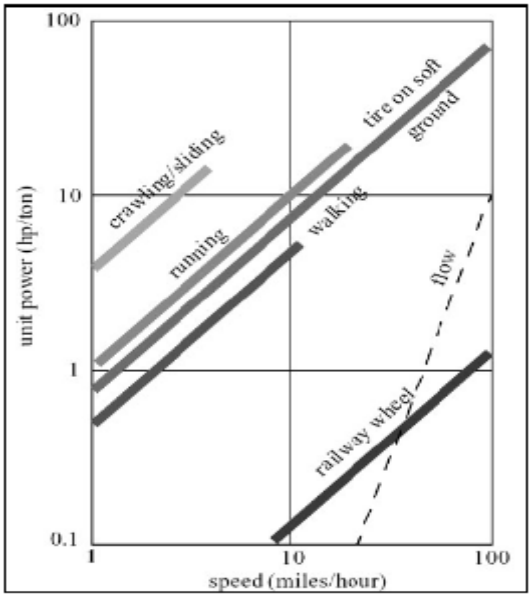


Figure 2.1 Power consumption of several locomotion mechanisms [56]

This leaves the option of using legged locomotion or wheeled locomotion. Though legged locomotion seems more energy effective, it has a few challenges when compared to wheeled locomotion:

- Stability: Legged systems are usually modelled after biological systems with 2, 4, 6 or 8 legs as in the case of humans, mammals, ants, spiders etc. However, when these systems attempt to move, a few of the legs will be lifted off the ground and the centre of gravity of the whole system will be displaced. In order to achieve balance at this point when the legs are off the ground, the robot should have at-least three legs on the ground and ensure that the centre of gravity is within the tripod of ground contact [1]. This method of "static walking" requires a minimum of 6 legs. To achieve equilibrium with fewer legs, the robot will also have a large area of contact on the ground for each leg.

- Complexity: Since legs afford greater degrees of freedom, they are inherently more complex to design. A minimum of 2 degrees of freedom is generally required to move a leg forward by lifting it and swinging it forward. However, commonly, a third degree of freedom is added to this for more complex manoeuvres. Every degree of freedom can be likened to a joint of its animal counterpart and is usually implemented using stepper motors.

In light of the stability and complexity benefits of wheeled locomotion, it was decided that the navigation aspects of this research as well as the concept prototype would adopt wheeled

locomotion. A decision was made to have a two wheel differentially driven system. The two wheels were powered independently by two motors, with a proportional integral derivative controller (PID controller) [110] keeping the wheels synchronised. Details of the constructed prototype are available in Appendix II.

2.3 Robot Navigation

One of the primary requisites for an autonomous mobile robot is that it is able to plan its path in order to achieve its basic function. Complete area coverage robots aim to completely cover or inspect the whole area with the help of their sensors. Complete coverage robots have the advantage of removing any doubt that the robot has covered the target region. A factor that determines the efficiency of operation of such robots is the amount of time to completion [21]. In cases where the time taken for coverage of all aspects of the path is uniform, time to completion translates to the path length taken by the robot. Therefore, the ratio of the area covered to the path length is a measure of efficiency for these robots. Some robots such as autonomous cars on the other hand may purely prioritise generating shortest paths to destinations over area coverage. Other robots such as search robots that do not have to guarantee completeness may not seek to cover areas or have a destination to reach but use the incidence of the object to be found per unit path length as an efficiency metric [21]. The literature on robot navigation is vast and there are various algorithms catering to each of these navigation paradigms. While monitoring areas such as chicken farms, for a typical target application that is considered in this research, it is imperative that the robot inspects all regions that chickens have access to, as having dead chickens or eggs in an uninspected part of the farm for extended periods of time can cause contamination of the surroundings. Due to this reason, most of the literature reviewed in this section focuses on complete area coverage.

Navigation algorithms are also grouped based on their awareness of their environments. Offline navigation algorithms have a prior knowledge of their environment and can use information about the positions and shapes of obstacles and the demarcations of the area to be covered to plan an optimal path based on the application. Online navigation algorithms do not have a global understanding of the environment and have to rely on their sensors to get an understanding of their immediate environment. The sensors' readings yield local information based on which navigation decisions will have to be made. Offline algorithms can in some cases produce more optimal paths as they possess prior knowledge of the environment and local information in such cases typically does not reflect the overall shape of an obstacle [36].

2.3.1 Complete Area Coverage Algorithms

Complete area coverage algorithms typically work by using a heuristic approach to cover the entire area or divide the area to be covered smaller sections that are individually covered and linked together to provide a complete area coverage.

A template based heuristic area coverage method is outlined in [26], incorporating in a natural way, the kinematic and the geometric model of the mobile robot on the path planning procedure. The path planner is capable of dealing with a priori mapped or unexpected obstacles in the middle of the working space. If unmapped obstacles permanently block the planned trajectory, the path tracking control avoids these obstacles. A heuristic formula is used for choosing the best template to apply at each stage based on the robot's current understanding of the environment, aiming at maximizing the total covered area. Unexpected obstacles are circumvented by following the boundary at a pre-set distance. In experiments with a labmate robot, the total area covered was found to be 85.5% [26]. This algorithm does not deal with obstacles that are moved out of their usual positions or out of the work area and gets into a deadlock situation when confronted by deeply concave obstacles termed as *C traps*.

Hert, Tiwari and Lumelski propose a completely online approach to perform area coverage of a terrain by an Autonomous Underwater Vehicle (AUV) moving in 3 dimensional space such as a submarine mapping the ocean floor [27]. The algorithm performs a boustrophedon motion on an area whose boundaries and obstacles are not known beforehand. Since the algorithm focusses on underwater mapping, the obstacles are referred to as islands. The coverage area is described to have *inlets* – irregular concave sections of the coverage area lying to one side of a boustrophedon track that divides the coverage area into more than one section and can impact the efficiency and accuracy of boustrophedon procedure as shown in Figure 2.2. Figure 2.2 (a) shows how boustrophedon coverage of areas within convex boundaries is more straightforward than those enclosed by concave boundaries that contain inlets which cause multiple traversals of the parallel tracks to come out of the inlet I_2 while not covering the inlet I_1 at all Figure 2.2 (b) [27].

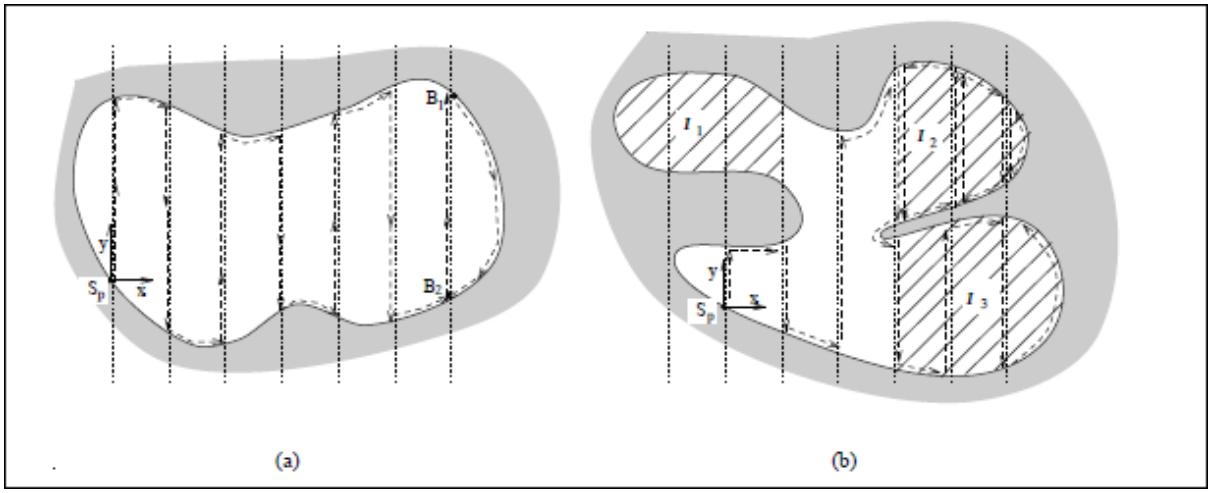


Figure 2.2 Effect of inlets on Boustrophedon coverage [27]

Capes are said to be the entrances of inlets and are the deeply concave parts of the boundary (looking from the outside) or the extremely concave or convex periphery of an island that divide the boustrophedon coverage track. The algorithm optimises area coverage by detecting capes within its sensor range and pre-empting the boustrophedon motion to follow the boundary of the cape through an inlet, recursively covering any cascading inlets that may occur. The robot is required to remember every point of intersection of the boustrophedon track with the boundary area. Once the boundaries of the inlet have been sensed, the robot covers the inlet using a boustrophedon motion and returns to the point of pre-emption to cover the rest of the path. The last track of the boustrophedon motion in every inlet is also remembered and called a *locked doorway*. This helps the algorithm navigate back to the starting point after covering an inlet encountered while following the boundary or shunting via the locked doorway. When coverage areas are not simply connected and are blocked by islands, the boundary of the islands are followed closely till either the starting point of the area coverage or the point at which the cape of the island or the entry point was encountered. Inlets within islands are covered in a similar way as described above. This algorithm is complete and provides excellent online coverage. However, for scenarios where obstacles and capes caused by temporary obstacles can possibly move, the algorithm will not function as expected. Further, if the area to be covered is fairly regular, unnecessary complete circumvention of the obstacles so as to distinguish them from inlets can be possibly redundant operations.

Other online coverage algorithms mostly work on breaking up the area into sub-areas. Approximate cellular decomposition algorithms [20] divide the area to be covered into cells,

dimensions of which typically depend on the robot's sensor range. The aim of decomposing the work space is to be able to apply the wide array of existing graph traversal algorithms such as the A* algorithm [108], the Depth First Search (DFS) [108] and the Breadth First Search (BFS) [108] on the cellular grid. Spanning Tree Coverage algorithm (STC) [30] and Boustrophedon A* (BA*) [34] algorithms are examples of algorithms based on approximate cellular decomposition. The optimisation of the path taken to visit the cells is called the *covering salesman problem*, an NP hard problem. The STC algorithm has both online and offline versions and aims at covering continuous planar area by a square shaped tool attached to a mobile robot. The side of the tool is said to have a dimension *D*. The offline STC algorithm computes an optimal covering path in linear time in the order of O(*N*) where *N* is the number of cells comprising the area. The online version executes in time in the order of O(*N*) and requires memory in the order of O(*N*). A second version of the online algorithm leaves pheromone like markers and is shown to have a time complexity of O(*N*) and requires only O(1) memory. The offline version of the algorithm first subdivides the work area into cells of dimension $2D \times 2D$ and then discards cells partially blocked by obstacles. A graph, G(*V*,*E*) is defined where the nodes *V* are the centre points of the new cells and *E* is the set of edges joining the centre points of adjoining cells. The algorithm then constructs a spanning tree using a Depth First Search (DFS) or PRIM [32] algorithm which is then used for area coverage.

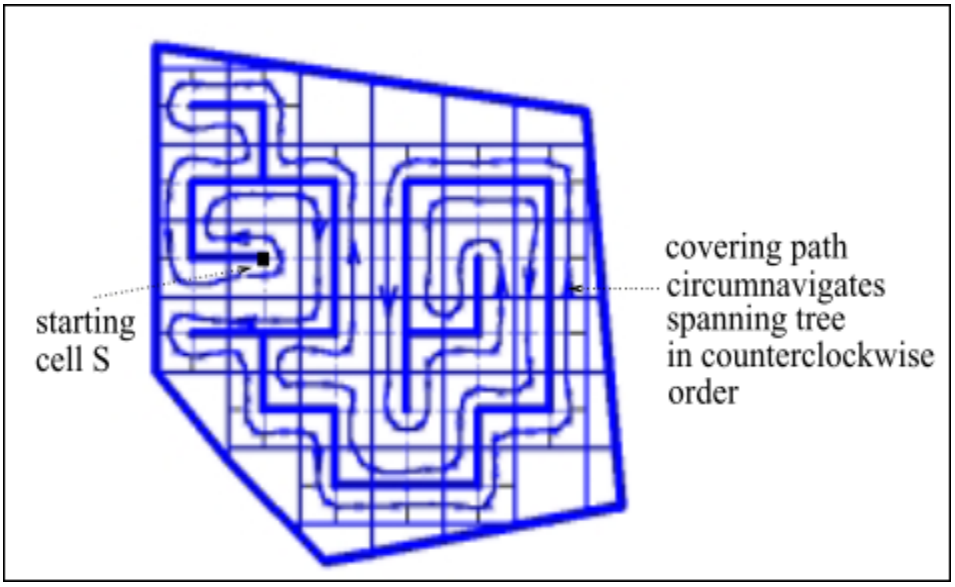


Figure 2.3 An example of the offline STC algorithm [30]

The online version of the algorithm polls each neighbouring cell in an anticlockwise order to see if a cell is unoccupied by obstacles and has not been previously visited. It moves to the first free cell, building spanning tree edges on the fly and forming a depth first approach to area coverage. As the algorithm requires the robot to memorise which cells have been covered, the space complexity of the algorithm is O(*N*). However, it is proposed that if the robot were able to leave markers in the cells, the memory map could be offloaded to the physical world requiring only O(1) memory.

The BA* algorithm [31] operates very similarly. However, it results in boustrophedon area coverage by polling the neighbours in a North - South - East –West order. When critical points, i.e. cells whose neighbours are either obstacles or have been covered previously, are encountered, the algorithm calls for backtracking to a cell with free neighbours. A list of possible backtrack points is maintained and the backtracking is optimised over the STC algorithm by backtracking to the physically closest cell. The A* algorithm [33] is used for the purpose of generating a backtracking path. Figure 2.4 shows an example of the BA* algorithm. As can be seen, the algorithm starts with the first track of the run marked 1, backtracks from the end of the area to the run marked 2and so on. The backtracking paths are shown in red.

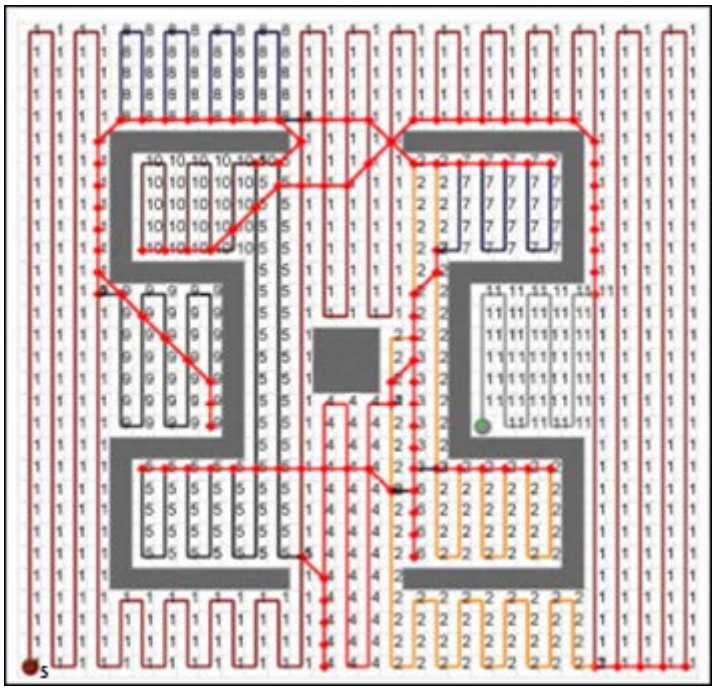


Figure 2.4 An example of the BA* algorithm using the A* algorithm to backtrack

Choset et al. [36] classify these algorithms as approximate since the union of unoccupied cells is expected to only approximate the whole free space to be covered. The size of the cells can be influenced by factors such as range of the on-board sensors. The accuracy of approximation depends on the how fine the grid is and therefore, a trade-off exists between efficiency and completeness of area coverage. Additionally, they provide effective online area coverage but require the overhead of backtracking.

Offline algorithms have the advantage of possessing complete knowledge of their surroundings. This means that they can further optimise path planning. An *exact cellular decomposition* is the set of non-intersecting regions each termed a cell, whose union fills the target environment [21].

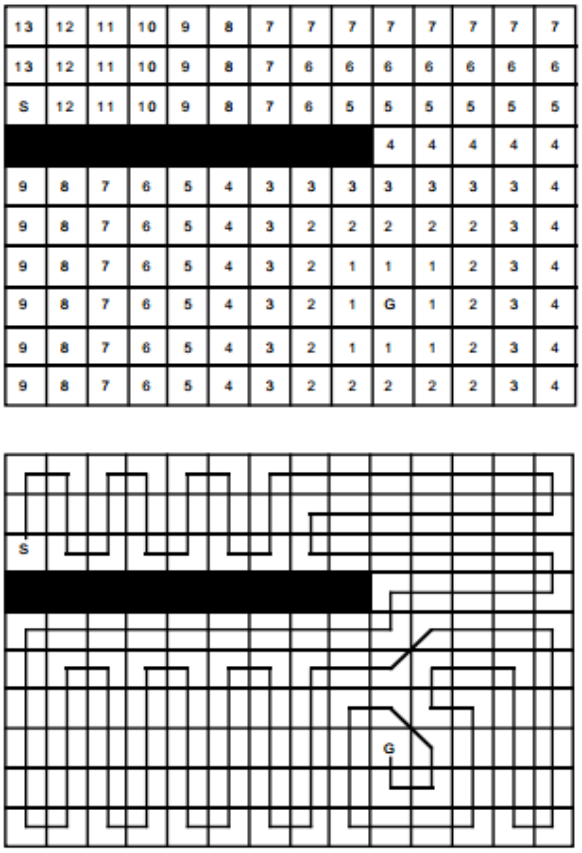


Figure 2.5 Area coverage using wavefront propagation [65].

An offline algorithm that works on cellular decomposition is the wavefront method [65]. In this method, the target cell is assigned a value of 0 and all neighbouring cells that are not occupied by obstacles are then assigned a value of 1. Then, all unmarked and unblocked cells surrounding these cells with value 1 are assigned a value of 2 and so on, till this *wavefront* crosses the start cell. To achieve the complete coverage path planning behaviour the robot

follows the path of steepest ascent. In other words the robot moves away from the target keeping track of the cells it has visited. The robot only moves into a grid cell which is closer to the target if it has visited all the neighbouring cells which lie further away from the target.

Figure 2.5 shows a complete coverage path example. This figure shows an environment with one obstacle, start (S) and target (G) locations, values of the distance transform, and a complete coverage path from S to G.

An offline strategy for polygonal worlds with polygonal obstacles is proposed in [31] and is called the *Minimum Sum of Altitudes* algorithm. This is an adaptation of the boustrophedon decomposition. It is an exact cellular decomposition that is primarily offline and is based on a planar line sweep segmentation of the area to be covered and can be made online. This algorithm focusses on minimising the number of turns the robot has to make to cover a subregion in a boustrophedon motion, as turns necessitate deceleration. Additionally, robots with nonholonomic constraints (robots that have fewer controllable degrees of freedom than total degrees of freedom) require greater efforts at turning. The number of turns to cover a boustrophedon cell is proportional to the width of the region perpendicular to the boustrophedon motion. The problem then becomes that of choosing a slice direction that minimises the sum of the altitudes of all cells in the decomposition. A diameter function is defined to describe the altitude of the polygon along the sweep direction. For a given angle θ, the diameter of a polygon is determined by rotating the polygon by θ and measuring the height difference between its highest and lowest point as shown in Figure 2.6. As can be seen, the diameter function is piecewise sinusoidal with breakpoints occurring when an edge of the rotated polygon is parallel to the horizontal. To minimize this sum of diameter functions, the second derivatives at critical points need to be checked. However, the second derivatives of critical points between breakpoints are negative which indicate local maxima and therefore, the minimum must lie at a breakpoint of one of the component diameter functions. Since these breakpoints correspond to when the boustrophedon sweep direction is perpendicular to an edge of a hole or the perimeter, the minimum can be determined by testing each of these sweep directions. The concept of a *nonuniform line sweep decomposition* is also introduced in [31] where the coverage area is divided into subregions with different boustrophedon sweep orientation, and is shown to provide more optimal area coverage.

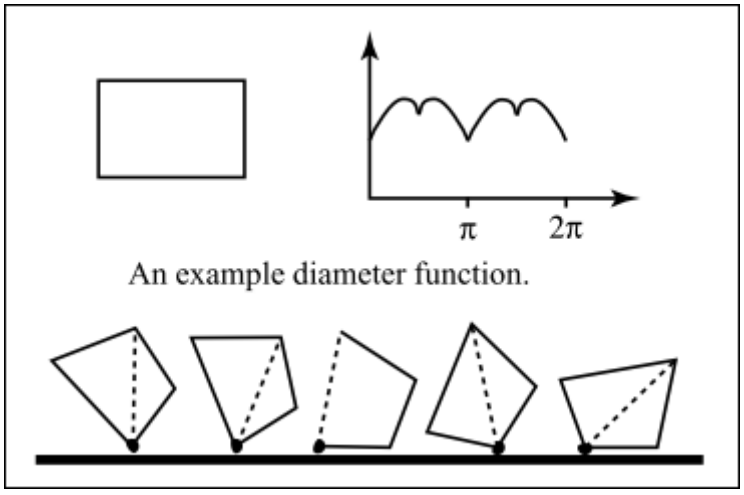


Figure 2.6 Creating the diameter function by rolling the polygon. The dotted line represents the chord used to determine the diameter or altitude of the polygon [31].

Recent studies have examined the use of multiple robots for the purpose of area coverage [57] . The advantage of using multi-robot systems is that they can minimise the time taken to cover areas, are more fault tolerant and not susceptible to single point failures and use each other as beacons to minimise dead reckoning errors. Batalin and Sukhatme [57] consider three techniques relating to multi-robot coverage:

1. *Informative,* where local identities are assigned to robots when they are within sensor range of each other. This approach relies on ephemeral identification where temporary local identities are assigned and mutual relative location information is exchanged between the interacting robots, allowing them to spread out in a coordinated manner. The simulations resulted in a maximum area coverage of 39.8% with 3 robots and 92.7% with 9 robots.

2. *Molecular,* where robots do not perform any directed communication, and no local identification is made. Instead each robot selects a direction 'away' from all its immediate sensed neighbours and moves in that direction without communicating with its neighbours. The simulations resulted in a maximum area coverage of 47.7% with 3 robots and 97.3% with 9 robots.

3. *Basic,* where there is no inter-robot interaction (other than obstacle avoidance). In this approach robots make no distinction between robots and other objects in the environment. The simulations resulted in a maximum area coverage of 48.4% with 3 robots and 86.1% with 9 robots.

As can be observed, the Informative and Molecular approaches outperform the Basic approach as the number of robots increased. Wagner et al. [58] discuss the problem of many simple robots cooperating to clean the dirty floor of a non-convex region represented by a grid. Wagner, who has also done related work on using pheromone like markers such as evaporating chemical odour traces to model the behaviour of social insects in communicating ant robots [59, 60], uses the dirt on the floor as the main means of inter-robot communication between the floor cleaner robots. Continuity of the uncleaned area is preserved by disallowing the robots from cleaning critical points which are graph points that lead to disconnecting dirty grid areas. This ensures that the robots only stop when the entire area has been cleaned. Additionally, this approach is fault tolerant, meaning that if a few robots fail, others will be able to complete the remaining tasks. Wagner et al. [61] show that the ant can cover the graph within time of O(*ND*) where *N* is the number of vertices and *D* the diameter of the graph. In [21], Choset also describes multi-robot approaches to exact cellular decomposition based algorithms. An offline method for multi-robot coverage based on Voronoi diagram and contour parallel coverage has been developed by Kurabayashi and others [62], while Rekleitis and team use visibility graphs like decomposition to enable area coverage [63]. Butler et al. [64] developed an online coverage algorithm where the cooperation and coverage are algorithmically decoupled. A single robot coverage algorithm that incrementally performs cellular decomposition has been extended to multi-robot systems by the inclusion of an *overseer* whose responsibility is to take incoming data from various robots and integrate it into a master map of the environment's cellular decomposition such that overlaps with new cells are reduced. In this method, robots accomplish complete coverage while being unaware of the communication with other agents.

2.3.2 Other algorithms for navigation

This section discusses navigation algorithms that do not target complete area coverage, but are specialised for certain applications and help understand the relevance of complete coverage algorithms to the target application. Often, when the objective of the robot's navigation is exploration or to search for objects, such as in the context of search and rescue operations, complete area coverage may prove less efficient than random approaches. One algorithm based on randomised components was developed by Balch and Arkin in [22, 23], where a team of robots comprising from 1 to 5 are made to search a space and bring objects back to the home-base and extended this to include multi-agent vacuuming tasks. The robots

have built in mechanisms of repelling each other to ensure better spread of search and obstacle avoidance. Since there is no path to be followed and search is random, the robots do not require accurate localisation which can be computationally expensive or require additional sensors such as GPS [24]. In some simulated coverage tasks, Balch found performance ratios of 5 to 1 between multi-robot teams using methodical search strategies and teams using randomised strategies. Therefore, if robots without localisation capabilities can be constructed at one-fifth the cost, it may be effective to use a randomised search. Gage [25] analysed contexts where the probability of doing a job, such as demining, is less than 1 even with complete coverage due to sensor limitations. A case was then formed that with diminishing probability of systematic area coverage being effective, using random search techniques with multiple robotic agents could be more advantageous than methodical searches.

Some of the earliest path planning algorithms were based on graph search algorithms [66] and aimed at finding paths from a start point to a target. The D* algorithm and its variants have been used in a prototype system tested on the Mars rovers Opportunity and Spirit [68]. These algorithms are based on Dijkstra's Algorithm [67] to find the shortest path in a weighted edge graph. It is a heuristic which examines its unvisited neighbours from a node and jumps to the node which costs the least. Whilst effective, these algorithms become very expensive while catering to obstacle avoidance.

The particle swarm optimisation or PSO algorithm [69] and its variants are optimisation algorithms which simulate the effect of birds finding a path to their target by introducing a factor of randomness. The PSO and accelerated PSO work with the belief that a swarm of particles exist. Each particle can be a robot or any virtual particle in the field of vision of a single robot. A fitness function determines the distance of each particle to the target. The position of the particle deemed most fit is used as the new best position and the positions of all the particles are incremented from this new position. This algorithm works well if path planning can be considered to be a unimodal optimisation problem, where the aim is to minimise the distance to the target. However, when there is a single robot, the particles in the swarm (field of vision of the robot) tend to be in the same area and if it is decided that the path is not to have large grain random increments in position (i.e. not jump over obstacles), this algorithm boils down to simply following the fitness functions. Also, this algorithm was found to get trapped in local minima when faced with concave obstacles [70].

2.3.3 Obstacle Avoidance

Obstacle avoidance is one of the most basic functions of autonomous mobile robots. In a typical agricultural application the robot would need to avoid obstacles such as trees or clusters of bushes, feeding and watering stations and some obstacles that can frequently change positions such as wheelbarrows, tractors etc. Additionally, it is highly advantageous if obstacle avoidance algorithms are resilient at coping with the presence of obstacles that constantly move and react to the motion of the robot such as chickens and other animals on the farm.

The aspects to obstacle avoidance are obstacle detection and planning a path to avoid them. There are many ways to detect obstacles. One of the simplest and earliest forms of obstacle detection used in the first known autonomous mobile robots, Elmer and Elsie, [84] was the use of IR sensors to echo-locate obstacles in their path. Several indoor mobile robots use IR sensors today. Other forms of echo location include LIDAR (Light Detection and Ranging), SONAR(Sound Navigation and Ranging) and RADAR(Radio Detection and Ranging). Additionally, it is intuitive that when detecting obstacles, checking for their persistence at a given position using the above ranging sensors intermittently is important to avoiding them. Several algorithms exist to circumnavigate obstacles. Some algorithms have prior knowledge of the shape and positioning of the obstacles around them, in which case they are called offline algorithms and can plan an optimal path around the obstacles. Other algorithms first survey an area and build a map of the obstacles around them and use offline methods to plan paths. The online algorithms that do not have knowledge of the obstacles surrounding them use local sensor data to navigate. Online obstacle avoidance algorithms are more responsive to changes in obstacle positions. Some of the most reliable path planning algorithms that guarantee to find a path if one exists are the Bug1 algorithm and some of its variants [83].

The most intuitive of bug algorithms, often dubbed the Bug0 algorithm [83], simply states that given a start position *S* and a target *T*, the autonomous robot starts from *S* and heads towards the target and if an obstacle is detected, follows the obstacle around its edge till the path towards the target is clear again. This algorithm works in most cases except when the target is surrounded by a spiral obstacle as shown in Figure 2.7.

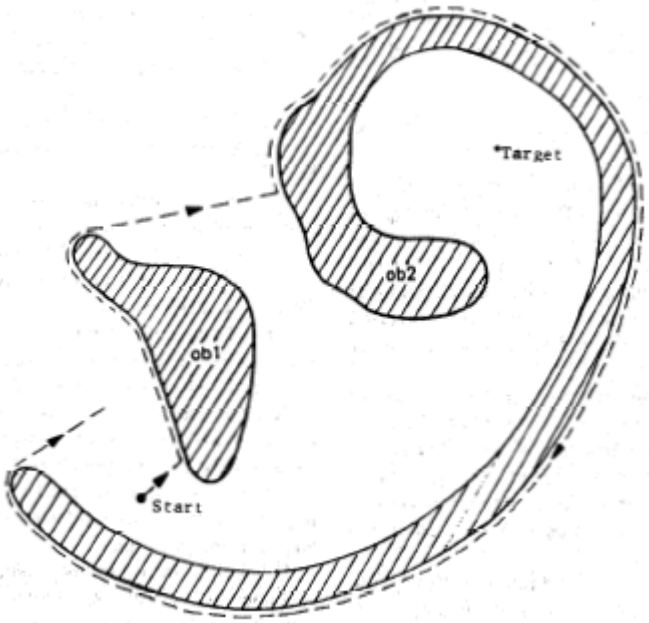


Figure 2.7 Scenario where bug0 algorithm fails [83]

This problem was solved by the Bug1 algorithm where, when an obstacle was reached, the robot follows the obstacle along its perimeter till it reaches the point where the object was encountered. While circumventing the obstacle, the robot constantly records its distance to the target and navigates to the point that minimises the distance to the target along the periphery of the obstacle by retracing its previous path. This guaranteed finding a path to any reachable target at the cost of multiple trips around the obstacles' edge (see Figure 2.8).

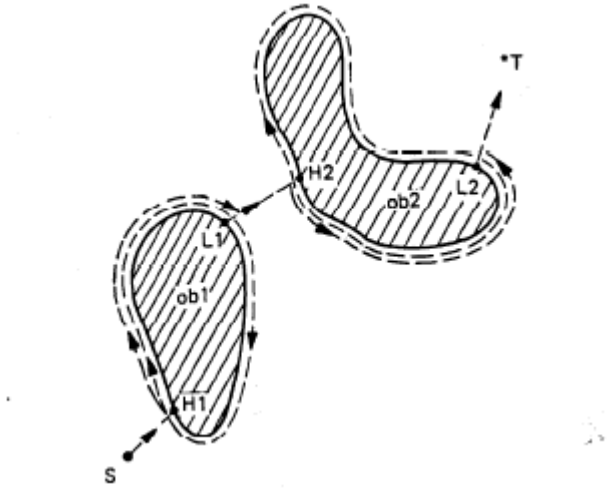


Figure 2.8 Bug1 path planning [83]

The Bug2 algorithm analysed by Lumelski and Stepanov in [85] further improved on Bug1 by drawing a straight line called the m-line (representing the shortest path) between the start

S and target *T* and having the robot travel along the m-line until either the target or an obstacle is encountered. If an obstacle is encountered, the point at which it is encountered, called the *hit point* H_i (the index subscript *i* denotes multiple occurrences of hit points) is noted and the obstacle is circumvented by closely following the periphery of the obstacle in a pre-configured direction using tactile sensors until

1. the m-line is encountered again at a point *Q* such that the length of *QT* is less than the length of H_iT in which case, a *leave point* L_i is identified and the loop of travelling along the m-line and circumventing obstacles till the target is reached or deemed unreachable is resumed.

   OR

2. the hit point H_i is encountered again. In this case the target is assumed to be embedded within the obstacle and unreachable and the algorithm terminates (see Figure 2.9).

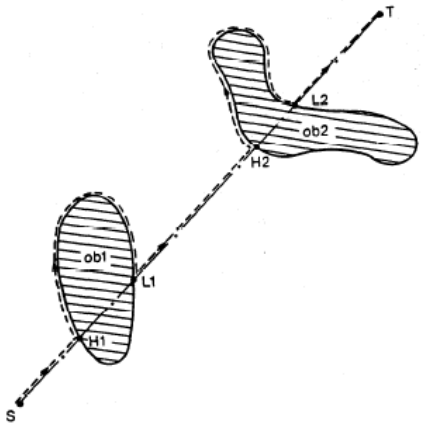


Figure 2.9 Bug2 path planning [85]

This algorithm is still not the most efficient way to avoid obstacles, but guarantees that any reachable target will be reached.

The VisBug-21 and VisBug-22 algorithms proposed in [87] incorporate range sensing into the Bug2 algorithm. The VisBug-21 follows the Bug2 algorithm more closely making local shortcuts detected by the sensors and limited by the sensor range. i.e., in a case where the range sensor can clearly sense the obstacle boundary, the algorithm directs the robot at every point on the path to look ahead to the farthest point along the path taken by the Bug2 algorithm as permitted by the sensor range and approaching it along the shortest path. These intermediate extreme points are called *intermediate targets*. However, if the target is visible

at any point, it is directly approached. The VisBug-22 algorithm is almost similar to the VisBug-21 except for the clause on detecting every point in the path of the Bug2 algorithm. The VisBug-22 detects points on the m-line that are within sensor range and closest to the target than the last encountered hit point and Intermediate target, should it lie on the m-line. These points are approached directly along the shortest path despite the algorithm not locating a leave point corresponding to the previously detected hit point. Unlike VisBug-21, not all intermediate targets have to lie on the Bug2 path and can lie on what is termed as a *quasi-Bug2* path[87], which is the path the Bug2 algorithm would take with the detected points on the m-line taken to be the starting point. The author demonstrates that neither approach is necessarily more optimal using illustrations in Figure 2.10. Figure 2.10(a) shows the path taken by VisBug-21, where it can be seen that points along the Bug2 path that can be traced by a sensor with the indicated sensor range r_v are directly approached. When it reaches the point *c*, the line of sight extends beyond A to a point within the semi-circular cavity and is directly approached. However, when the point A is reached, the robot is now able to see the target and approaches it directly. Figure 2.10(b) shows the path taken by the VisBug-22, where the initial path generated is identical to VisBug-21. However, when the point B is reached along a small distance from the outer periphery of the semi-circular hull, the VisBug-22 estimates the point A along the m-line to be the closest point to the target, closer than the previous hit point, traces the path taken by the quasi-Bug2 algorithm from this point to point D, after which the continuity of the obstacle boundary is obscured. A is defined as the new hit point and D is approached directly, causing the robot to make a second loop around the convex hull, when it finally takes the same course as VisBug-21 to the target. Figure 2.10(c) shows an instance where the path taken by the VisBug-21 algorithm is very long as the algorithm is restricted to only making local shortcuts along the path generated by the Bug2 algorithm. VisBug-22 however, performs optimally in this case by approaching the m-line directly at the first opportunity and then making a shortcut to the target *T* when it comes into view, as shown in Figure 2.10(d).

As the sensor range is reduced, the algorithms become more similar to Bug2 and miss out on some advantageous shortcuts. Figure 2.11 illustrates this phenomenon, where the target *T* is not directly approached from the right semi-plane, as is the case when sensor ranges are larger.

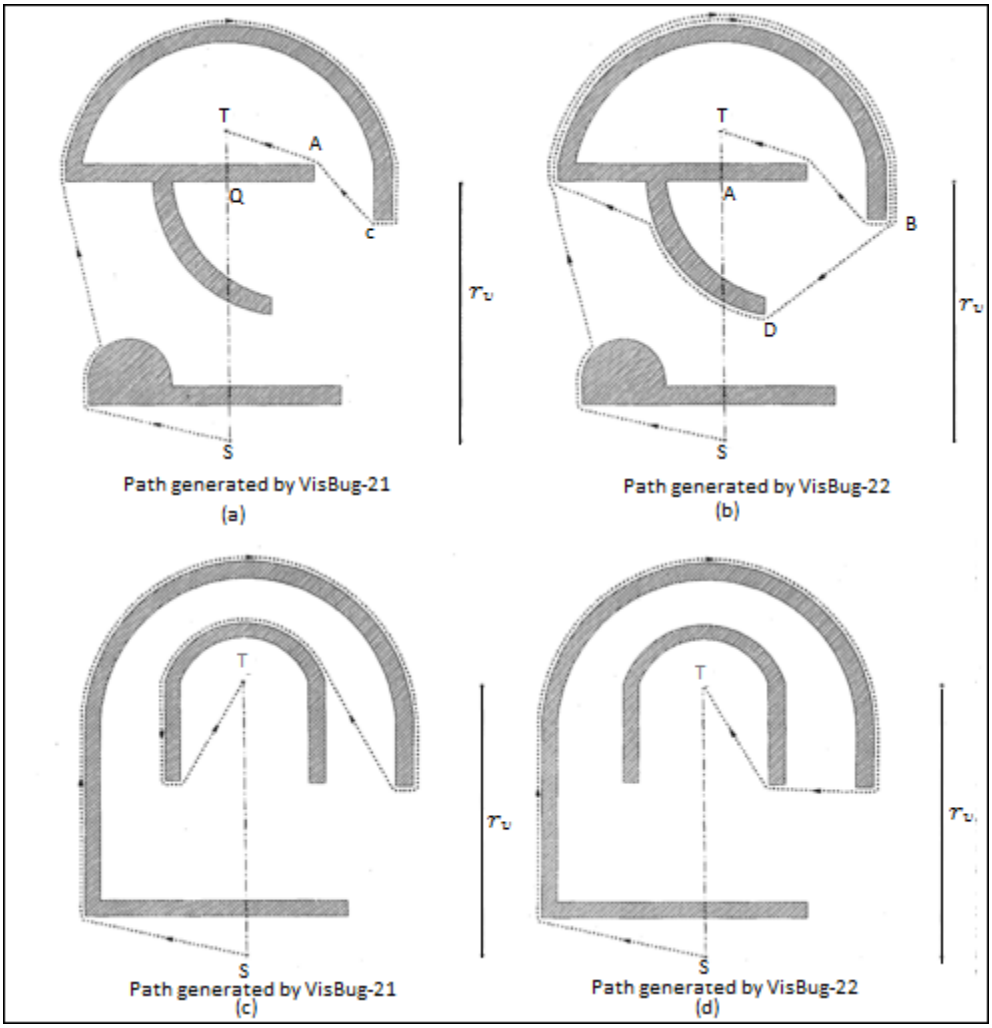


Figure 2.10 Comparison of the paths generated by the VisBug-21 and the VisBug-22 algorithms [87]

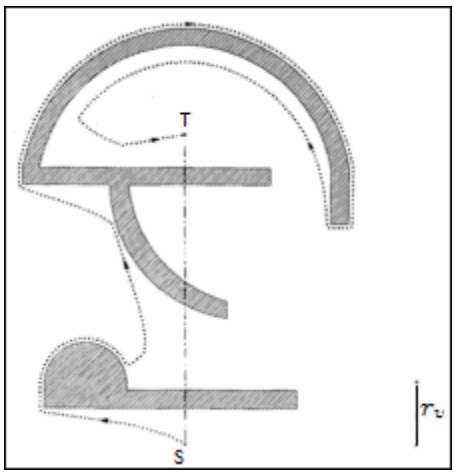


Figure 2.11 Paths generated by the VisBug-21 and the VisBug-22 algorithms are similar to the Bug2 algorithm when sensor range is smaller than in Figure 2.10 [87]

Rao and Iyengar deviated from the bug algorithms while developing a terrain acquisition algorithm that builds a map of an obstacle terrain populated by an unknown (but, finite)

number of polygonal obstacles of varied sizes and locations in the plane leveraging on the use of range sensors [80]. The algorithm which is to be used on robots of finite sizes is an extension of a previous algorithm based on terrain acquisition by a point robot [81]. The basic terrain acquisition algorithm was developed for an ideal point-sized robot equipped with a computing device and an ideal sensor is capable of detecting all vertices visible from the present location of the robot. This algorithm constructs a visibility graph based on local sensor information from time to time. The visibility graph of an obstacle terrain O, VG(O) is defined as a undirected graph (V,E), where

1. V is a set of all vertices of all obstacles
2. The line joining two vertices u and v , u ,$v \in V$, forms an edge $(u ,v) \in E$ if it is not obstructed by an obstacle or it is an edge of an obstacle.

The point robot initially starts in the obstacle terrain O at a start point and moves to the closest obstacle vertex. From this starting vertex the algorithm to acquire the terrain is invoked. The adjacency list of v (in VG (O)) is built by detecting all the vertices visible from v using the scan operation. The vertex v is marked 'visited' and then pushed onto a stack. There are two cases:

1. If v has unvisited adjacent nodes, then the autonomous point robot moves to a node, say w, which is nearest (to v) among the unvisited adjacent nodes. From w the algorithm to acquire terrain is recursively invoked.
2. If all adjacent nodes of v are visited, then the nodes on the stack are repeatedly popped till a node x with at least one unvisited adjacent node is obtained. Then shortest paths to each of the unvisited adjacent nodes of x are computed using the Dijkstra's shortest path algorithm [67]. The point robot chooses the shortest path (among the computed ones) and moves to the corresponding unvisited node w. From the node w the algorithm to acquire terrain is recursively invoked. The algorithm (and hence the terrain acquisition process) terminates when the point robot is located at a vertex u such that (a) all nodes adjacent to u are visited, and (b) the adjacent nodes of each node on the stack (at this time) are all visited. At this point, the point robot moves back to the starting vertex along the shortest path.

It is argued in [81] that by building a comprehensive map of the obstacles in an environment, a robot can use existing computational strategies to avoid obstacles in a path from the start to

the target without having to rely on sensor information. Such rigid imposition of stationary constraints on the environment is a disadvantage when it comes to the practical implementation of the algorithm. However, the visibility graph sets the stage for more advanced online obstacle avoidance algorithms that optimally circumvented obstacles, such as the TangentBug algorithm [86].

The TangentBug algorithm proposed by Kamon, Rimon and Rivlin [86] creates a local tangent graph (LTG) using range sensor based information – This is a subset of the tangent graph within the sensor range and the tangent graph in turn is a subset of the visibility graph. The visibility graph is a graph whose vertices are the obstacle vertices and the start and target points, and edges are the collision free segments that connect graph vertices. The tangent graph retains only the convex vertices of the visibility graph and edges that are bi-tangential to these vertices as shown in Figure 2.12.

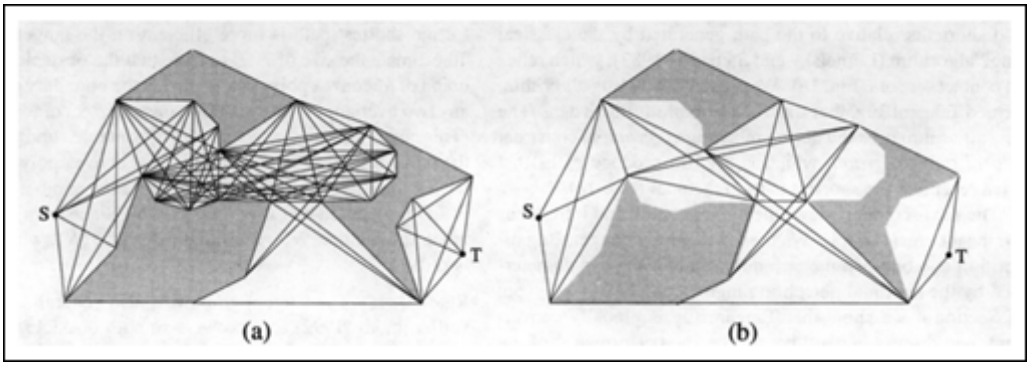


Figure 2.12 The Visibility Graph (a) and Tangent Graph (b) [86]

The LTG can be constructed as if the local range information represents all of the obstacles and each sensed object is modelled as a thin wall (see Figure 2.13).

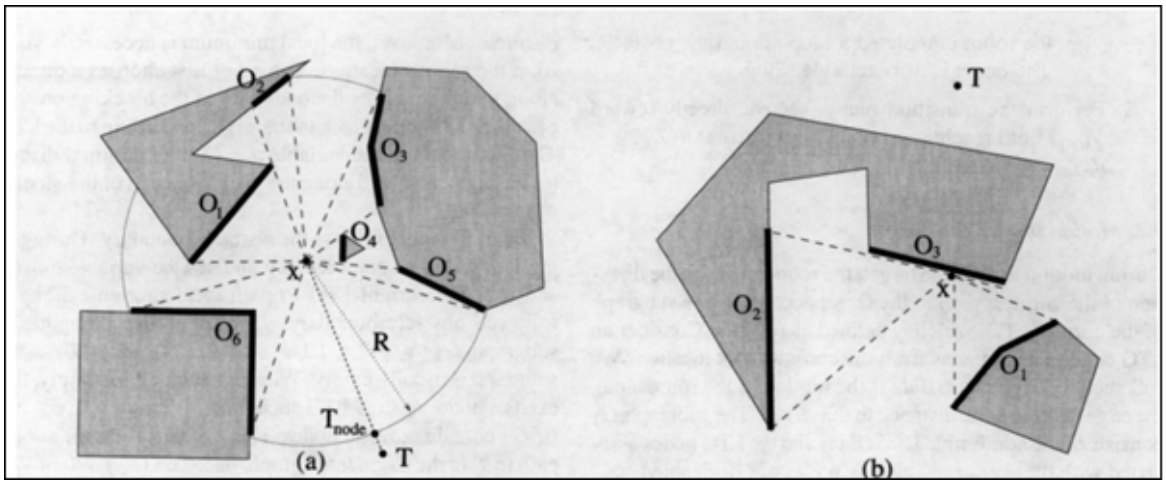


Figure 2.13 Obstacles modelled as thin walls in the LTG and the special node T_{node} [86]

The nodes of the LTG are the end points of the sensed obstacles, the current robot location x and the additional optional node T_{node} in the direction of the target. If the direction to the target is not blocked by an obstacle, T_{node} is the furthest point on this line, unless the target is in the visible set, in which case T_{node} is the target. The edges of the LTG connect the robot location with the other vertices of the LTG. The other edges that are bi-tangent to the sensed obstacles are never used, so they are not constructed.

Informally, the algorithm directs the robot to move along the locally optimal edge of the LTG constructed at every step, until either the target is reached or a local minimum has been detected, in which case the obstacle boundary is followed using the LTG to make local shortcuts until either the local minimum has been overcome by using a stipulated exit condition or the target has been reached. Once the local minimum has been circumvented, the motion toward the target is resumed. Figure 2.14 shows the path taken by a point mobile robot using the TangentBug algorithm to avoid obstacles. The start point is *S* and target is *T*. The points H_x depict points where obstacles were first encountered. The points D_x represent *depart points* where the robot leaves the current obstacle boundary. The points M_x represent local minima detected where boundary following is initiated and L_x represent the corresponding *leave points* where the robot switches to motion to the target behaviour. Dashed lines show motion to the target behaviour and dotted lines show boundary following behaviour. The circles represent the sensor range of the point mobile robot.

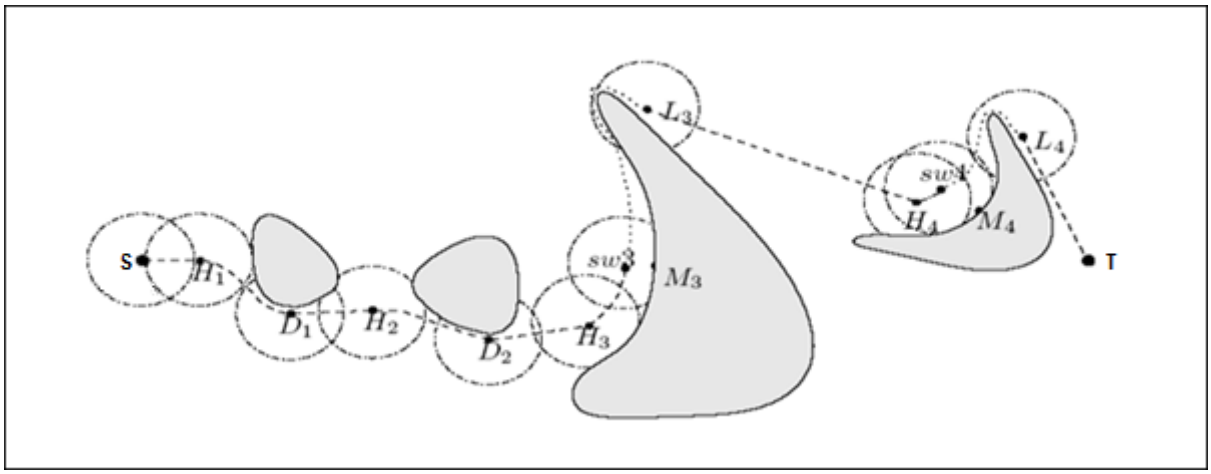


Figure 2.14 A demonstration of the TangentBug algorithm with the sensor range represented by a dotted circle [56].

Simulation based comparison studies show a performance improvement of upto 72% of the TangentBug over VisBug algorithms in office like environments, 80% in an environment cluttered with convex obstacles and 90% in a maze-like environment [86].

2.3.4 Localisation

Localisation is the method of determining the robot's position in the workspace using sensor data. Localisation mechanisms include *idiothetic* mechanisms such as maintaining track of the robots position based on its initial location and parameters such as the speed and direction of the robot. This method is called dead reckoning and is a very popular choice for indoor robot localisation as it depends on typically low cost encoders. The feedback from motor encoders is used to get the actual speed of the robot and calculate its position. Although dead reckoning is fairly suitable on even ground, it suffers from cumulative errors which become intensified on uneven terrain due to wheel slippage [71]. Kalman filters can be considered to provide better location estimates while using odometry [74]. Idiothetic mechanisms can also be used to determine the orientation of the robot in systems such as the differentially driven systems, where one of the two wheels will turn along a smaller radius while the other will do so along a larger radius, i.e., the wheels will be made to rotate at different speeds. The angle by which the turn has been made will be recorded and used to update the orientation of the robot.

Allothetic mechanisms for localisation involve the use of localising systems such as GPS and active or passive beacons made of RF sensors. GPS based techniques, although easier due to their ready availability tend to be expensive and have accuracy varying from 1 – 5 meters [73]. Since in most cases the robot is expected to make vertical traces of the field that are spaced closely when compared to the accuracy of the GPS, this method as a stand-alone means of localisation will not suffice. N. Bulusu and others [72] propose using active RF beacons and two methods of interpreting the received signal strength. In the first method, they check the strength of the received signal. They discovered that the received signal strength at short ranges was subject to variation due to fading, multipath and interferences in short range (around 10m). The second approach dubbed the idealized radio mode, involved setting up an RF grid and the mobile node (or robot) would listen to various beacon signals and would infer its proximity to the reference beacons based on the received signal strength being above a threshold and would then localise itself to be in the region of overlap between the detected beacons. As can be inferred from Figure 2.15, the localisation becomes more

effective as the ratio of the transmission range of the reference point to the distance of separation between adjacent reference points increases. This low cost alternative is very attractive. However, it suffers from some minor drawbacks:

- Set up time to place beacons at standard distances between each other.
- The approach has low precision and experimental results suggest it is useful for applications with less stringent accuracy requirements.

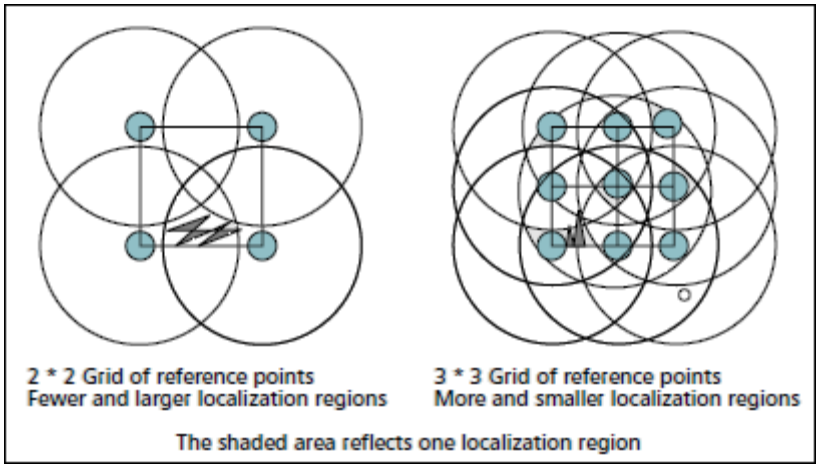


Figure 2.15 Idealized Radio Model [72]

Optical encoders are a popular choice for measuring angular speed and position. It is basically a mechanical light chopper consisting of a stationary light source and light sensor on either side of a fixed grating that masks light and an optical grid that is attached to the shaft and therefore rotates with it [77]. As the rotor moves, the fixed and moving gratings allow different amounts of light to fall onto the optical detectors which is then converted to a sine wave, which can be further converted to a square wave by thresholding to choose between light and dark states. Although optical encoders allow the measurement of angular speed with great accuracy, in order to measure the direction of rotation it is better to use a quadrature optical wheel encoder, which improves over its predecessor by introducing a second light emitter-detector pair shifted 90° from the original in terms of the disc.

Once the angular speed and direction are available, the robot can calculate its position using the initial position and the radius of its wheels. However, as stated before, when implemented practically, the method of relying on odometry suffers from cumulative errors built up due to irregularity of the ground and wheel slippage. Also, once the robot makes a small deviation to avoid obstacles, it will be necessary for the robot to come back to its course and realign itself to the direction it was travelling. One way of reducing dead reckoning errors was proposed by

Borenstein [71, 74]. This system was implemented on a Multi-Degree-of-Freedom (MDOF) mobile platform, a vehicle in which two differential-drive mobile robots (called *trucks*) are physically connected through a compliant linkage. In addition to its four wheel encoders, the MDOF platform has one linear and two rotary internal encoders, which allow measurement of the relative distance and bearing between the two trucks. During operation, both trucks perform conventional dead-reckoning with their wheel encoders. In addition, this method uses information from the internal encoders to detect and correct dead-reckoning errors as soon as they occur [71]. This is a relatively expensive solution and adds about $1000 to the cost of the robot. A simpler solution was proposed by David. C. Conner [75], who recommended the addition of a third encoder on the caster wheel of the robot. In situations where wheel slippage may occur, such as when the robot is stuck in slushy mud, the castor wheel would not rotate, thereby giving more accurate measurements of actual distance travelled by the robot.

C. Smart and others [76, 79] propose a white paper on using a combination of a gyroscope for orientation and an encoder for measuring distance travelled. This system, called the Gyro-encoder, can be useful when the estimated orientation differs greatly from the actual orientation of the system. As can be seen in Figure 2.16 which represents the results of their experiments, the gyroscope adds substantial accuracy to the robot's path perception. They concluded that on close loop circuits, the addition of the gyroscope increased accuracy by several orders of magnitude.

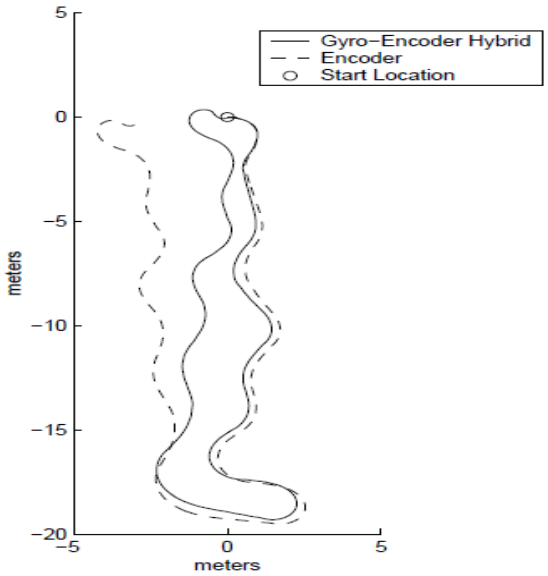


Figure 2.16 Comparison of the Gyro-Encoder Hybrid model with the encoder only model [79].

Inertial Measurement Units (IMUs) such as those listed in [97], offer a convenient way to supplement the feedback from encoders with that from gyroscopes, accelerometers and magnetometers, to further improve the reliability of the robot's localisation.

2.4 Robot Sensors for obstacle avoidance and detection

Autonomous robots rely on several sensors to make navigation decisions. As discussed in the previous section, sensors such as encoders, IMUs and GPSs provide mechanisms to help the robot to perceive its actual position in an environment more accurately. In addition to localisation, sensors also help robots navigate around obstacles when obstacle positioning and shape are not known accurately. Infrared, LIDAR (Light Detection and Ranging) and Ultrasonic sensors are a few commonly used mechanisms that are used to navigate around obstacles. The method outlined in [98] is popular and discusses how a vision system can be used in conjunction with ultrasonic sensors to navigate around stationary obstacles with some limitations. However, as in [98], obstacle avoidance mechanisms based on vision systems are limited by computing power required for image processing and therefore not as reliable as those based on other sensor mechanisms.

Infrared sensors have good ranging abilities but face interference from the IR spectrum of direct sunlight [18] and are therefore unsuitable for use on robots operating outdoors. Laser ranging systems provide excellent ranging and angular resolution. However, they are very expensive [19] for use on a low cost robot. Ultrasonic sensors are low cost sensors with reasonable performance both indoors and out. The SRF05 [89] is a very popular choice of an ultrasonic sensor due to its cost effectiveness, easy availability and reliability. The technical specifications show the beam pattern to be as in Figure 2.17.

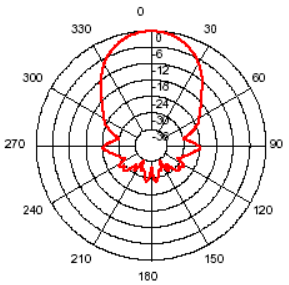


Figure 2.17 Beam pattern of the SRF05 [89]

Wickramasooriya et al. [88] have demonstrated that the detection zone of the SRF05 sensor can be roughly approximated to a leaf pattern where it spreads out at an angle of 55° reaching

a maximum breadth of 80cm to a meter before tapering off to 0 at a length of 350m. Within this detection zone, the sensor gives the distance of the closest obstacle. While the range resolution is accurate to the centimetre, the azimuth resolution or the angular resolution is very poor owing to the large spread of the detection zone. Several methods exist to cope with low sensor resolution such as the centreline model [42], McKerrow's common tangent method [41], the Arc Transversal Mean method [42], etc. The centreline model is an intuitive and popular model which assumes that the obstacle is at the centre of the sonar arc[42]. It provides reasonable estimates of an object's location if the object is close to the sensor. The error produced by this model can approximately vary between 1.5cm and 40cm, derived from the range specification of the sensor. The problem with this approach as discussed by Choset, Nagatani and Lazar [42] is that due to the approximation of the position of the obstacle to the centre of the sonar arc, narrow spaces between obstacles can appear narrower. For example, as shown in Figure 2.18, when a robot equipped with ultrasonic sensors is faced with the decision of whether to enter a narrow passageway, it may choose not to since the actual width of the passageway is *c* which is incorrectly estimated using the centreline method as *a,* as the robot passes by the passageway which is less than the robots dimensions, i.e., $c<b<a$

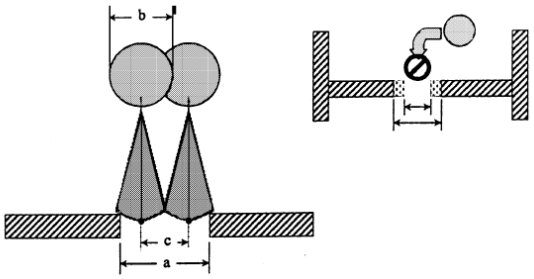


Figure 2.18 The centreline model can make a narrow passage seem narrower [42]

McKerrow proposes a new model of combining sequential sonar readings from a fixed sensor on a robot with the motion of the robot to estimate the position of the edges of polygonal obstacles, known as McKerrow's common tangent method [41]. McKerrow suggests that when an obstacle is first encountered, it is tangential to the sonar arc. However, as there are infinitely many points on the arc, there can be infinitely many tangents to it and therefore infinite orientations and positions of the object. This uncertainty can be resolved by taking multiple readings of the same sensor as the robot moves and attempting to fit a common tangent to the different sonar arcs. Let r_{prev} and r_{curr} be readings from the same senor on a robot before and after travelling a distance *d* as shown in Figure 2.19. Let the centreline of the sonar arcs make an angle of *a* with the direction of motion of the robot. If *b* is the half

width of the sonar arc, then McKerrow suggests that a common tangent exists if $| r_{prev} - r_{cur} |$ lies between *d cos(a-b)* and *d cos(a+b).* Once this condition is satisfied, the slope of the common tangent, the range vectors within the two sonar arcs that are perpendicular to this tangent and the centres of the sonar arcs are available. Equations of the lines representing the range vectors can therefore be found. Since the lengths of the range vectors are known, the points at which the obstacle is tangential to sonar arcs can be estimated.

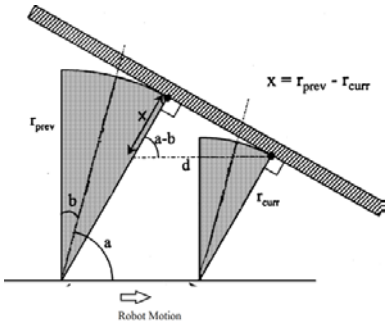


Figure 2.19 The McKerrow method of determining a common tangent through two sonar arcs [42]

The disadvantage of the McKerrow method is that if the face of the obstacle is not strictly planar or if a corner is encountered on a polygonal obstacle, it poses a challenge when the condition for a common tangent is not met. Following the boundary of such obstacles will be difficult if this technique is used to improve angular resolution.

It is clear that sensed obstacles can lie anywhere along the sonar arc. When arcs intersect, intuitively the point of reflection is likely to lie on the intersection. The arc transversal mean or ATM method [42] relies on finding intersections between sonar arcs as the robot goes past an obstacle. Considering single intersections can often give misleading information as shown in Figure 2.20. If many arcs intersect in the close vicinity of a region, the probability of the point of reflection being in that region is high. Ideally these arcs should coincide. However, they may cluster in an area due to dead reckoning errors or sensor errors and the median of these intersections is considered in such a case. In situations where no intersections are found along an arc, the centreline is used. This method also considers only stable intersections, i.e., arcs which intersect transversally. This means that the location of the intersection does not change greatly if the sonar arcs are slightly perturbed due to dead reckoning errors. This angle was found to be 30° in experiments involving sonar sensors [42]. Figure 2.21 shows transversal and non-transversal intersections. This method is best suited for non-specular walls and to detect corners. If the surface of the obstacle is specular and flat, the ATM

approach gets reduced to the centreline approach due to the dearth of arcs that intersect transversally.

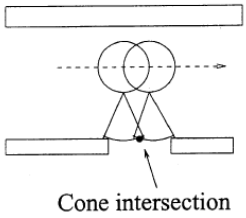


Figure 2.20 Single cone intersections can give misleading results [42]

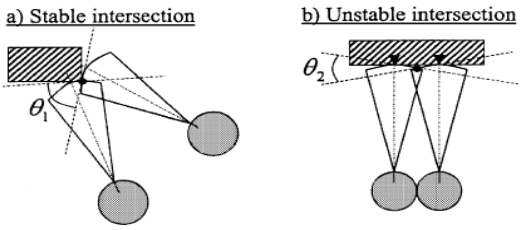


Figure 2.21 Transversal and non-transversal intersections [42]

In addition to sensors necessary for navigation, it would be advantageous for an autonomous robot that monitors agricultural areas to have a vision system. Although the development of the vision system is out of this project's scope, some literature was reviewed as part of the feasibility study towards the development of the robot. Machine Vision System (MVS) has been increasingly employed in the agricultural sector in the recent years. While autonomous harvesters need it to identify a wide variety of fruit in order to harvest them, many autonomous robots are employed in de-weeding farms since the use of chemical weed killers can be harmful both to the plant and the consumer of the produce. Yang. C. and others [92] discuss the use of Artificial Neural Networks *(*ANN) in the detection of weeds on a corn farm. They concluded that the ANN after training with several samples and using different number of Processing Elements (PE) and output layers yielded a success rate of 80 to 100% for the identification of corn, while the success rate for weed classification was as high as 60 to 80%. Applications such as free range poultry farming will need vision systems to aid functions such as detecting eggs that need collection and detecting chicken to report sick/dead chicken.

In the field of image processing, features of the object to be detected are paramount. These features are mostly based on increased curvature (such as corners) [93]. As such, in the scenario of detecting eggs, the eggs tend to have fairly few identifiable features. Other

conventional methods of object recognition in image processing involve Support Vector Machines (SVM) which analyse data using classifiers and regression analysis [107]. Here too, the features of the objects come in handy. Artificial Neural Networks (ANN) are also extensively used in machine vision. Artificial Neural Networks consist of several processing elements (PEs) that mimic the neurons in the human brain. The PEs are organized as layers and for most ANNs, PEs in each layer are fully connected with PEs in the adjacent layer or layers, but are not connected to other PEs in the same layer [92]. Another method of object recognition, proposed by Mackworth and others [93], involves detecting the corners of the outline of all the objects and using this data to describe the features of various objects. This method is called Curvature Scale Spacing – CSS.

Classically popular algorithms such as the SIFT(Scale Invariant Feature Transofrm) [94] that rely heavily on feature based classification do not perform consistently when faced with detecting eggs on a noisy background such as dirt and hay. For example in Figure 2.22 (a) a simple implementation of the SIFT algorithm has correctly identified the box of Basmati rice by its feature rich inscription. Unfortunately, the background hay in Figure 2.22 (b) has more features than the object of interest – eggs. This confuses the SIFT algorithm which will not be able to identify eggs.

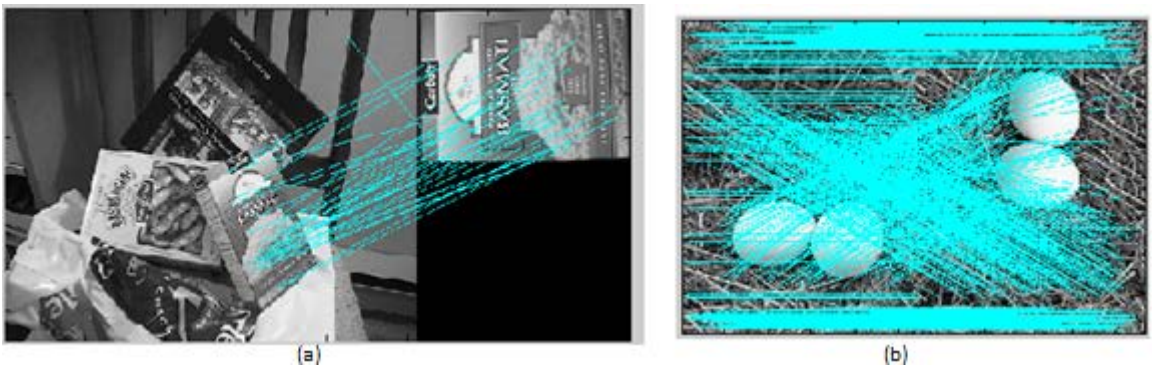


Figure 2.22 Performance of the SIFT algorithm [96]

One of the best ways of detecting a relatively featureless egg on a noisy background is to segment out the object on the noisy background and then perform analysis on the result. Segmentation is the process of classifying the pixels/subsets of an image based on some property – typically colour or intensity level in order to establish clear boundaries. Often, while segmenting textures, a subset representing a pattern from which the whole is made up is considered and segmented. Thresholding techniques which are based on choosing pixels falling within a range in their histograms are also often used to detect boundaries.



Figure 2.23 Results of the EDISON system [95]

Figure 2.23 is the result of a segmentation algorithm EDISON [95] which uses a confidence based edge detector and a mean shift based image segmentation algorithm. Once this is done as a pre-processing step, a variation of the CSS algorithm [93] can be used to detect which segments minimize the number of corners to the enclosed area.

2.5 Conclusion

This chapter presents an overview of the application of robotic technologies in agriculture, highlighting the evolution of agricultural robots from systems that used brute force and very little intelligence to fully autonomous systems using the latest technologies.

It was seen from experiments in existing research involving locomotion systems that wheeled locomotion offered stability and reduced design complexity without altering the power consumption greatly. This helped design the prototype which in turn helped decide on what aspects of area coverage would be relevant to this research.

The aim of the section on area coverage was to present the relevant literature that focussed on complete area coverage. The current algorithms would provide a potential answer to the problem of near complete area coverage of the specific applications targeted in this project. However, the literature review revealed that there was room for improvement. For instance, several graph traversal based coverage strategies that approximated the area to be covered, existing approaches based on boustrophedon coverage and randomised approaches did not offer complete coverage at the outset. The literature review prompted the need to experiment with a complete area coverage strategy based simply on boustrophedon coverage that effectively circumvented obstacles as they occurred without having to map them. It was seen from the literature review on obstacle avoidance that the TangentBug algorithm generated the shortest path when it came to avoiding obstacles of all shapes while the concept of the m-line

seen in the Bug2 algorithm would fit naturally into a boustrophedon path. Therefore, a best of both worlds approach to obstacle avoidance would be highly desirable.

It was discovered from initial experiments in this research with the prototype of the robot that it was possible to attain extremely small turn radius using differential steering without greatly changing the speed of the robot from what was required. Therefore, turning in headland areas of farms and around obstacle periphery is not likely to cause impediments to the area coverage process. Subsequently, literature focussing on reducing the number of headland turns has not been covered.

Finally, literature on ultrasonic sensors was reviewed as it was understood that they provided the best outcomes to assist with the navigation of an agricultural robot at a reduced cost. The issues arising out of using ultrasonic sensors and existing methods of dealing with these issues were detailed.

3 Development of a new Hybrid Bug Algorithm for obstacle avoidance during area coverage

3.1 Introduction

This chapter details the new algorithm called the Hybrid Bug algorithm, which was developed specially to be used in obstacle avoidance during area coverage operations in the animal husbandry sector considered in this research. Typical of the target applications, the navigable space is vast in comparison to the obstacles, which are usually smaller convex polygons, although this is not a strongly imposed restriction. The new algorithm is tailored to be used for obstacle avoidance in the context of boustrophedon area coverage and produces optimal results in the absence of large maze-like obstacles. Details regarding the choice of the boustrophedon pattern for area coverage are discussed in Chapter 4. The Bug2 algorithm works by travelling along the line that connects the start and target points and avoids obstacles by tracing their periphery. The concept of a straight line connecting the start and target positions, borrowed from Bug2 algorithm, translates directly to one track of the boustrophedon pattern, with the start and target points being the beginning and end of the boustrophedon track. As the boundaries of the area to be covered are known, this can be easily computed. However, the Bug2 algorithm does not circumvent obstacles very efficiently. The TangentBug algorithm, on the other hand, is considered to be much more efficient in circumventing obstacles of unknown shapes and dimensions but does not trace the m-line. The Hybrid Bug algorithm attempts to combine these two algorithms to maximise the efficiency of the resulting obstacle avoidance and area coverage operation, which is measured in this research in terms of the resulting path length and amount of area covered. Also, it is ensured that the algorithm terminates in the ideal case, as is the case with the component algorithms. Consequently, this chapter first discusses these two algorithms that provide the foundation to the new Hybrid Bug algorithm and then describes the Hybrid Bug Algorithm. In the subsequent section on the analysis of this algorithm, proof of termination of the Hybrid Bug algorithm is presented in addition to comparisons with existing algorithms. Finally, a summary is presented in the conclusion section.

3.2 Preliminaries

The main aim of the new Hybrid Bug Algorithm is to maximise the coverage of the line between the start and target positions while minimising the length of the path generated when

avoiding obstacles that may lie along this line. The Bug2 algorithm's notion of the m-line - a line connecting the start and target positions, is a useful concept, as is the TangentBug algorithm's ability to leverage range sensors to plan optimal paths around obstacles online. The new Hybrid Bug Algorithm is a combination of these two algorithms. Hence, an overview of these two algorithms is presented first in this section.

3.2.1 Bug2 algorithm

The Bug2 algorithm was developed by V.J. Lumelski and A. Stepanov[85]. They propose an online obstacle avoidance algorithm to avoid obstacles where a point mobile robot with no prior knowledge of its surroundings avoids obstacles using sensor readings. It is argued that since the robot does not have an idea about its environment, global optimality is ruled out. No approximation of the obstacle is done and computational complexity is measured in terms of the length of the generated path as a function of the obstacle perimeter.

Given a start position *S* and a target *T*, the goal of the Bug2 algorithm is to generate a path from *S* to *T*. The straight line (*S, T*) is sometime referred to as the *m-line*. The point at which a mobile robot (or *mobile automaton* as referred to in [85]), MA, first encounters an obstacle that is sensed by its tactile sensor, is called the *hit point* H_j. The point at which it leaves an obstacle is called the *leave point* L_j. The subscript *j* is the index of the encounter of either the same or a different obstacle by the robot. Initially, let L_0=*S*. *j* is then assigned the value 1.The distance to *T* from any point *x* is denoted as *d(x,T).* With these definitions in place, the procedure of the algorithm is described in [85] as follows:

Algorithm 1 – Bug2Algorithm

Note that the inputs to this algorithm are the start point S and the target T.

Step 1: From the point L_{j-1} move along the straight line (*S, T*) until one of the following occurs

- The target *T* is reached. The procedure has completed successfully.
- An obstacle is encountered and a hit point, H_j, is defined. Go to step 2.

Step 2: Using the accepted local direction, follow the obstacle boundary until one of the following occurs:

- The target is reached. The procedure has completed successfully.

- The line (S, T) is met at a point Q such that the distance $d(Q,T) < d(H_j, T)$, and the line (Q, T) does not cross the current obstacle at the point Q. Define the leave point $L_j = Q$. Set $j=j+1$. Go to step 1.
- The robot returns to H_j and thus completes a closed curve (the obstacle boundary) without having defined the next hit point, H_{j+1}. The target is trapped and cannot be reached. The procedure is terminated.

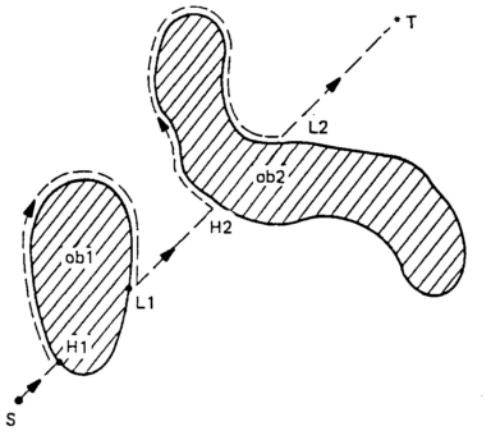


Figure 3.1 Robot's path (dashed line) under Bug2 [85]

This algorithm improves on the Bug1 algorithm in that the entire perimeter of the obstacle is not skirted every time the robot needs to circumvent it. However, this can be disadvantageous, as V.J. Lumelski. et al. show [85], if the start is embedded within a maze, the inner boundary of the maze is followed closely a number of times, as shown in Figure 3.2 where the inner boundary is followed up to three times.

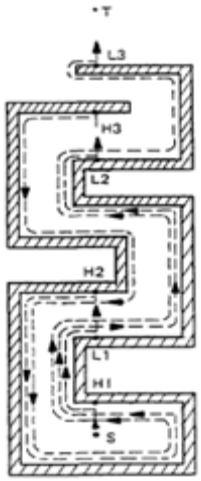


Figure 3.2 An example of a local cycle in a maze like obstacle [85]

However, in the general cases of agricultural applications considered in this research, the occurrence of patterns such as mazes is highly unlikely. Hence, such obstacles can be ignored for the purpose of this application.

The usability of a path planning algorithm lies in its ability to terminate and not waste precious resources in infinite cycles of execution. Additionally a guarantee that a target will be reached if it is accessible increases the confidence in an algorithm's efficiency. A proof of the Bug2 algorithm's termination and a test for target reachability given in [85] are summarised below:

The possibility that the Bug2 algorithm will execute infinitely occurs when it is trapped in an infinite loop while trying to circumvent an obstacle in step 2 of Algorithm 1. A local cycle is said to have been created when the robot has to pass some point of its path more than once. The following lemma and theorem, proofs of which are given in [85] guarantee that the procedure terminates by showing that there is a limit on the number of local cycles generated and therefore the path length.

Lemma 3.1 : Under Bug2 algorithm, the robot will pass any point of the i^{th} obstacle boundary at most $n_i/2$ times, where n_i is the number of times the i^{th} obstacle intersects the (*S*, *T*) line.

Theorem 3.1*:* The length of a path generated by the Bug2 algorithm never exceeds the limit

$$P = D + \sum_i \frac{n_i p_i}{2}$$

where *D* is the distance from *S* to *T* and p_i refer to the perimeters of the obstacles intersecting the (*S, T*) line and n_i is the number of times the i^{th} obstacle intersects the (*S, T*) line.

In addition to the Theorem 3.1, a test for target reachability is given in [85]. Every time after leaving the point H_j , the robot can expect one of the following to occur:

- The robot will never return to H_j – in the absence of a local cycle.
- The robot will define at least one pair of points (L_j, H_{j+1},...) before returning to H_j to start a local cycle .
- The robot returns to H_j without having defined on the previous cycle a point L_j. In other words, robot could find no other intersection point *Q* of the line (H_j, *T*) with the current obstacle such that *Q* would be closer to the target *T* than H_j, and the line (*Q*, *T*)

would not cross the current obstacle at *Q*. This can happen only if either the robot or *T* is trapped inside the current obstacle (Figure 3.3). This condition is both necessary and sufficient to test for target reachability.

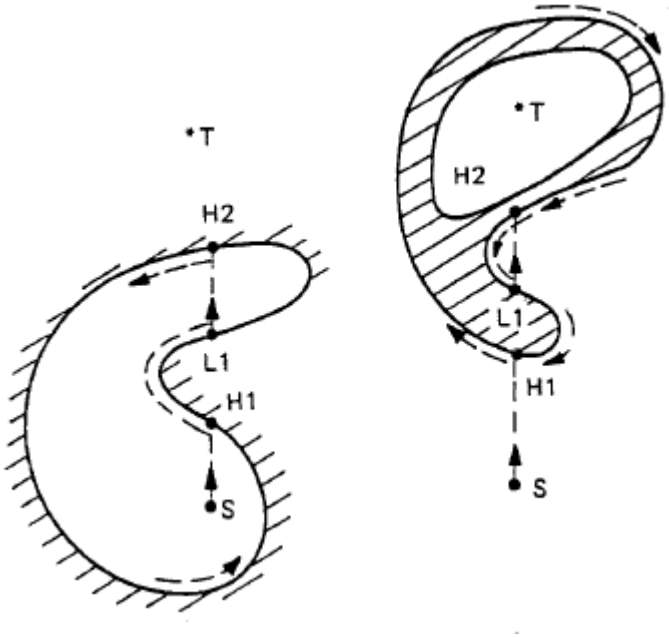


Figure 3.3 Examples of traps [85]

Formally, if, on the p^{th} local cycle, $p = 0, 1, \ldots,$ after having defined a point H_j, robot returns to this point before it defines at least the first two out of the possible set of points L_j, H_{j+1},..., . . . , H_k, it means that the robot has been trapped and hence the target is not reachable.

The test along with the theorem and lemma suffice to ensure that the Bug2 algorithm terminates.

Although the Bug2 algorithm has the advantage of tracing the m-line and is shown to terminate, the behaviour it displays in closely following the obstacle contours in a fixed direction can be quite disadvantageous as it can result in long obstacle circumvention paths. In order to overcome this disadvantage, the TangentBug algorithm is explored.

3.2.2 TangentBug Algorithm

The TangentBug algorithm [86] creates a local tangent graph (LTG) using range sensor based information. This is a subset of the tangent graph within the sensor range and the tangent graph in turn is a subset of the visibility graph. The visibility graph is a graph whose vertices are the obstacle vertices and the start and the target points and edges are the collision free segments that connect graph vertices. The tangent graph [99] retains only the convex

vertices of the visibility graph and edges that are bi-tangential to these vertices. The tangent graph, also referred to as the global tangent graph, is expected to contain the shortest path in the environment. The local tangent graph [86], is a tangent graph that includes only the portion of the obstacles that lie in the sensor range of the robot centred at its current position. The LTG is constructed as if the local range information represents all the obstacles in the environment. This assumption allows the robot to base its local decisions only on the currently observable obstacles, thereby greatly simplifying the data processing.

The nodes of the LTG are defined as the end points of the sensed obstacles, the current robot location x and the additional optional node T_{node} in the direction of the target. If the direction to the target is not blocked by an obstacle, T_{node} is the furthest point on this line, unless the target is in the visible set, in which case T_{node} is the target. The edges of the LTG connect the robot location with the other vertices of the LTG. Bi-tangential edges are not used. The Euclidean distance to the target T from any point x, while circumventing any part of the visible obstacle, is denoted as $d(x, T)$. Broadly, the algorithm has two behaviours: one is the motion toward the target behaviour and the other is the boundary following behaviour. The edges of the LTG will be used to plan the shortest distance to the target when the robot engages in the algorithm's motion toward the target behaviour and local shortcuts during the boundary following behaviour. $d_{min}(T)$ is the minimum distance to the target T observed along the visible portion of the blocking obstacle and is updated continuously while following the boundary. A summary of the TangentBug algorithm according to [86] is as follows:

Algorithm 2 - TangentBug algorithm

Note that the inputs to this algorithm are the start point S and the target T.

Step 1: Move toward T along the locally optimal direction on the current LTG subgraph, until one of the following events occurs:

- the target is reached. The procedure has completed successfully.
- a local minimum is detected. Go to step 2.

Step 2: Choose a boundary following direction. Move along the boundary using the LTG while recording $d_{min}(T)$, until one of the following events occurs:

- the target is reached. The procedure has completed successfully.

- the leaving condition holds : $\exists V_{leave} \in$ LTG such that $d(V_{leave}, T) < d_{min}(T)$. This means that a vertex has been identified in the LTG that produces a shorter path to the target from the current position than the minimum distance to the target, $d_{min}(T)$, recorded during the boundary following behaviour. Go to step 3.
- the robot completed a loop around the obstacle. Therefore the target is unreachable. The procedure is terminated.

Step 3: Perform the transition phase. Move directly toward V_{leave} until reaching a point Z that satisfies $d(Z, T) < d_{min}(T)$. Go to Step 1.

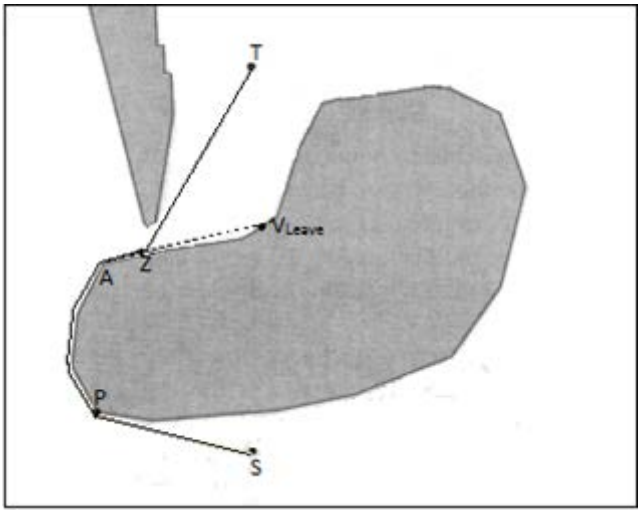


Figure 3.4 A path generated by the TangentBug algorithm, image modified from [86]

In Figure 3.4, the path generated by the TangentBug algorithm, implemented using an infinite range sensor is shown. *P* is the point at which the robot switches to the boundary following behaviour. At point *A*, the robot identifies a leave vertex V_{Leave} and approaches it directly. However, during this transition phase, at point *Z*, the target is no longer obscured by the second obstacle and $d(Z, T) < d_{min}(T)$. Therefore from this point the robot approaches the target *T* directly.

The following theorem and supporting lemmas and propositions prove that the TangentBug algorithm terminates in the ideal case of infinite sensor range. In the following statements, $d(w, T)$ represents the Euclidean distance of a point *w* from *T*, such that the points *w* belong to the free space and P_i is a point where the robot switches from the motion toward the target behaviour to the boundary following behaviour.

Theorem 3.2: Using unlimited sensor range, TangentBug algorithm always terminates after following a finite-length path.

Lemma 3.2: The distance of the robot from the target, $d(x, T)$, decreases monotonically during each motion toward the target segment, and between successive motion toward the target segment.

Proposition 3.1: Using unlimited sensor range, each motion toward the target segment has finite length.

Proposition 3.2: Using unlimited sensor range, each boundary following segment has finite length. Moreover, if the target T is reachable from the point where the robot switches to boundary following behaviour, the leaving condition will cause the robot to leave the obstacle.

Lemma 3.3: Every switch point P_i, where the robot switches from the motion toward the target behaviour to the boundary following behaviour, has a corresponding local minimum point M_i of the distance function $d(w, T)$, such that $d(M_i, T) < d(P_i, T)$.

Corollary 3.1: Let M_i be the local minimum of $d(w, T)$ associated with the switch point P_i. Then the distance to the target decreases between successive local-minimum points, that is, $d(M_{i+1}, T) < d(M_i, T)$.

The proof of Theorem 3.2 and the supporting lemmas, propositions and corollaries can be found in [86]. The following theorem and the proposition it is based on, assert the completeness of the TangentBug algorithm [86].

Theorem 3.3: Using unlimited sensor range, TangentBug algorithm always finds the target if it is reachable from the start point.

Proposition 3.3: Using unlimited sensor range, each boundary following segment has finite length. Moreover, if the target T is reachable from the point where the robot switches to boundary following behaviour, the leaving condition will cause the robot to leave the obstacle.

The proof of Theorem 3.3, the completeness of the TangentBug algorithm, is shown in [86]. Since the length of the path generated by the TangentBug algorithm is finite, an upper bound,

Len_{max} on the path generated by the TangentBug algorithm using an unlimited range sensor is shown in [86] to be

$$Len_{\max} = \left(1 + \frac{1}{\gamma}\right)\|S - T\| + \sum_{i \in \chi} (\Pi_i + \rho) \times \#Minima_i$$

where χ is the index set of the obstacles that intersect the disc D_T which denotes the disc with centre at T and radius $\|S\text{-}T\|$ - S is the start position and T is the position of the target. In addition, Π_i is the perimeter of the i^{th} obstacle, and $\#Minima_i$ is the number of local minima of $d(w, T)$ in D_T along the i^{th} obstacle boundary. The parameter γ is a small positive parameter controlling the admissibility of the candidate edges for the LTG and ρ is a positive and finite parameter representing the range w ithin which the robot is allowed to seek candidate nodes satisfying the leaving condition.

3.3 Description of the new Hybrid Bug algorithm

A hybrid algorithm combing the beneficial aspects of the Bug2 and the TangentBug algorithms has been developed to suit area coverage applications where the amount of Navigable space is vast and obstacles have small widths. As described in the previous sections, the TangentBug algorithm is inspired by the fact that the edges of a Global Tangent Graph contain the shortest path. While there is no guarantee that Local Tangent Graphs give rise to the very same path, it is more likely that decisions made based on local sensor information give better outcomes than having a fixed direction of circumvention as in the case of the Bug2 [85] and VisBug [87] algorithms. As sensor ranges become more comparable to obstacle dimensions, the only aspects of the Global Tangent Graph not reflected in the Local Tangent Graphs are the edges and vertices obscured by obstacles. Another advantage of the TangentBug algorithm is its ability to use the LTG to make local shortcuts, which is a contributing factor to the 80% improvement in the performance of the TangentBug algorithm over the VisBug algorithms in an environment cluttered with convex obstacles [86]. These features of the TangentBug algorithm are employed in the Hybrid Bug algorithm. The Hybrid Bug algorithm uses the concept of the m-line of the Bug2 algorithm, which is the straight line connecting the start and the target positions. In the new Hybrid Bug algorithm, the m-line represents a portion of the path that the robot needs to travel along in order to attain complete area coverage. Therefore, maximising the distance travelled along the m-line will result in maximising area coverage. Aspects of path planning for complete

area coverage are discussed in depth in Chapter 4. As the algorithm takes a greedy approach to cover the m-line, any moving obstacle such as a chicken, which happens to be detected along the m-line causing a diversion to circumvent it, will typically not have a profound impact on coverage since the robot will travel back to the m-line as soon as the approach to the m-line is free from obstacles. This is another advantage over algorithms such as the Bug2 or the VisBug-21 which try to locate the periphery of the obstacle boundary all the way to the m-line.

The Hybrid Bug algorithm has the following 3 basic behaviours:

1. Motion toward the target or toward the m-line behaviour
2. M-line coverage behaviour
3. Boundary following behaviour.

This is different from the TangentBug algorithm which only has the motion toward the target behaviour and the boundary following behaviour. Note that the Bug2 Algorithm also has two behaviours: the boundary following behaviour and m-line tracing behaviour.

Some basic definitions of constructs used in the Hybrid Bug algorithm are as follows:

The *Local Tangent Grapth(LTG)*: At every stage of the motion toward the target behaviour, a Local Tangent Graph (LTG) is constructed to aid navigation. The definition of the LTG is slightly different to the LTG defined in the TangentBug algorithm [86]. The LTG used by the Hybrid Bug algorithm comprises of the end points of the sensed obstacles, the current robot location x and a new version of the optional node, T_{node}. The edges of the LTG connect the robot location with the other vertices of the LTG. The other edges that are bi-tangent to the sensed obstacles are never used, so they are not constructed. The main difference between the LTGs of the TangentBug algorithm and Hybrid Bug algorithm is in the way in which the optional node, T_{node} is defined.

A new definition for T_{node}: The LTG described in [86] has a special optional node called T_{node} which is the furthest point along the direction to the target, within the visible sensor range, if the direction to the target is not blocked by an obstacle. If however, the target is in the visible set, T_{node} is the target. This node has been redefined in the Hybrid Bug algorithm, depending on the current position of the robot, as follows:

- **If the robot is on the m-line:** If the robot is moving along the m-line, the T_{node} of the LTG will now be the furthest point along the m-line towards the target T when no obstacles are sensed within the sensor range r along this direction. If however, obstacles are sensed, we define another configurable parameter t (a threshold distance from the obstacle), to increase the distance travelled on the m-line. In a case where there is an obstacle along the m-line within the sensor range, T_{node} will assume an alternate definition as the furthest point along the m-line towards the target T at a threshold distance t from the obstacle.t is allowed to take effect only when it is assigned a value less than r. if $t > r$, T_{node} will not be defined in this scenario.
- **If the robot is not on the m-line:** If the robot is not on the m-line, a point T_{proj} is defined, which is the projection of the current robot position onto the m-line. A temporary target, T_{temp} is defined. If no current value is assigned to T_{temp}, it is assigned a value equal to T. If a current value $T_{temp} \neq T$ is defined but an obstacle lies along the path to T_{temp}, T_{temp} is reset to T. Further, if $T_{temp}=T$ and T_{proj} lies between H and T, where H is the hit point where the robot last encountered an obstacle on the m-line, there exist two cases:
  - The approach to the m-line along the normal is free of obstacles: T_{temp} is set to T_{proj}.
  - The approach to the m-line along the normal is not free of obstacles: If D is any point on the m-line segment $\overline{HT}$ (see Figure 3.5), such that it is visible from the current position x and $d(H,T) > d(D,T)$, but an obstacle is detected along the segment $\overline{DT}$ and the line $\overline{Dx}$ is *admissible* (described as follows), where $d(x,T)$ denotes the distance to the target T from any point x, T_{temp} is set to the point D that minimises the length of the line segment $\overline{T_{proj}D}$. One such situation is shown in Figure 3.5, where the line segment $\overline{T_{proj}x}$ is blocked by an obstacle but a set of points are available such that for any point D in this set, segments $\overline{Dx}$ are unobstructed and admissible, and $d(H,T) > d(D,T)$. T_{temp} is now set to the point D that minimises the length of $\overline{T_{proj}D}$.

The special node T_{node} of the LTG will now be the furthest point along the direction to the new target T_{temp} and all distances are calculated from T_{temp} until it is either reached or is blocked by an obstacle. Upon reaching T_{temp}, the algorithm briefly suspends motion toward the target behaviour and switches to m-line coverage behaviour, inspects the m-line in the

direction of the start *S*, covers any part of the m-line that has not been traversed before returning to T_{temp} and resuming motion toward the target behaviour. In both cases, the target shifts back to *T*.

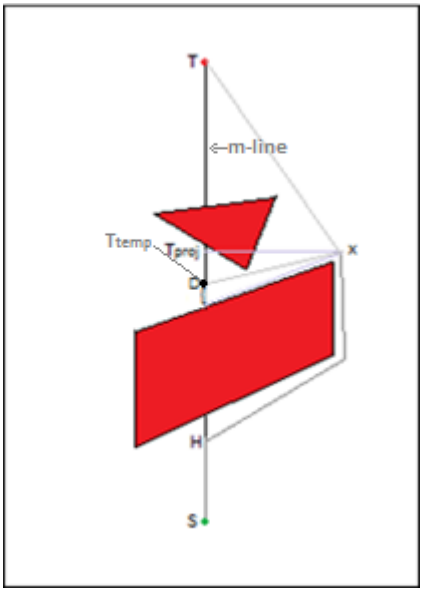


Figure 3.5 The robot at *x* has no straight line approach to the m-line.

Criterion for *admissibility*: An edge is called *admissible* if travelling along the edge decreases the distance to the target. More formally, if *x* is the current robot location, the set of vertices that gives rise to admissible edges emanating from *x*, V_{G1}, is defined as

$$V_{G1} = \{\{x\} \cup \{V \in LTG : (V - x) \bullet (T - x) > 0\}\}$$

This equation states that the dot product of the candidate edge and the line connecting the target to the current robot position must be positive for it to be considered admissible. Put simply, the enclosed angle between them must be acute.

Note that the criterion for admissibility is different when the robot approaches a $T_{temp} \neq$T, since the dot product $(V - X) \bullet (T - X)$ is not repeatedly evaluated when the robot travels along $\overline{xT_{temp}}$ as in the case of other vertices of the LTG. Therefore, in order to maintain the property of the TangentBug algorithm that the travel along this direction monotonically reduces distance to T, which in turn guarantees termination of the algorithm, the admissibility criterion for an edge $\overline{xD}$, with reference to Figure 3.5 is defined as

$$\angle TxD < \frac{\pi}{2} - \frac{\angle xTD}{2}$$

where, *T* is the target, *x* is the current robot position and D is a point on the line segment $\overline{HT}$, such that it is visible from the current position *x, d(H,T)>d(D,T). H* is the hit point where the robot last encountered an obstacle on the m-line.

It can be shown that travelling along any edge $\overline{xD}$ satisfying this condition monotonically decreases the distance to the target. The proof that this criterion gives a path with monotonically reducing path length is discussed in Appendix IV. The restrictions on all approaches to the m-line not along the normal serve another purpose apart from guaranteeing convergence of the algorithm: it reduces the path length to the m-line while increasing the time spent on the m-line. The fact that the m-line can be approached in other directions than the normal also allows the robot to explore some m-line segments lying between 2 obstacles that may not have a straight line approach along the normal as in Figure 3.5.

The *subgraph G1*: The concept of the subgraph *G1* of the LTG is also adopted from the TangentBug algorithm [86]. The subgraph *G1* consists of admissible LTG edges, an optional edge leading to the m-line and constituent nodes. In Figure 3.6, the nodes of the subgraph G1 are $\{x, T_{temp}, A, B\}$ and the edges are$\{\overline{xT_{temp}}, \overline{xA}, \overline{xB}\}$ where *x* is the current robot location. During the motion toward the target behaviour, the robot moves in the locally optimal direction, along the edges of a subgraph *G1* of the LTG. When the graph *G1* is augmented with virtual edges from nodes of *G1* to *T*, it is called the *augmented graph* [86].

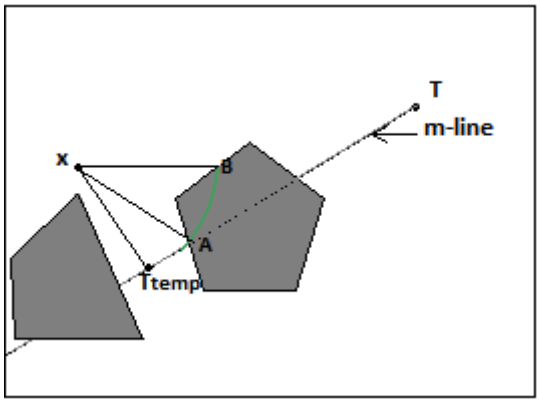


Figure 3.6 An illustration of the subgraph G1

Once the nodes of the sub-graph *G1*, V_{G1} have been identified, the absolute distance of each of the sub-graph vertices to the target (*T* or T_{temp}) is computed and added to the distance of each vertex from the current robot location. That is, the values of $d(x, V_{G1}) + d(V_{G1}, T_{temp})$ are

computed. The *locally optimal direction* is chosen as the direction to the vertex that minimizes this value.

Identifying a *local minimum* : The robot keeps moving toward the target along the locally optimal direction until it is trapped in the concave region of a *local minimum*. This is identified to have happened when all the vertices of V_{G1} produce longer paths to target than the current robot position – i.e. $d(V_{G1}, T) > d(x, T)$.

The *Leaving Condition*: The robot switches to the boundary following behaviour upon identifying a local minimum. The robot follows the boundary in the chosen direction, and moves along the boundary while using the LTG to make local shortcuts. However, the robot may not leave the boundary before the following leaving condition is met:

> While the robot is following the boundary, it records the minimal distance to the target, $d_{min}(T)$, observed so far along the boundary of the followed obstacle. The robot leaves the obstacle boundary when there is a node V_{leave} in the current LTG that satisfies $d(V_{leave}, T) < d_{min}(T)$.

After leaving the obstacle, the robot performs a transition phase before it resumes its motion toward the target behaviour. In the transition phase, the robot moves directly toward V_{leave} until it reaches a point Z that satisfies $d(Z, T) < d_{min}(T)$. At this point the robot resumes its motion toward the target via the m-line, when possible.

In addition to the optional node T_{node}, the LTG consists two types of nodes – *tangent nodes* and *occlusion nodes*. Tangent nodes are the sensed tangent obstacle vertices V with the property that the line segment from the current robot position x, $\overline{xV}$ is tangential to the obstacle at V. Occlusion nodes are points along obstale edges that arise out of occlusion by a convex obstacle vertext, as showin in Figure 3.7. At every step the robot moves in the direction of a particular tangent node, occlusion node or T_{node} on the LTG. This node is called the current *focus node*, and is denoted F. In Figure 3.7, B_L is the occlusion node and also the focus node, as the shortest path to T passes through it. V_L and V_R are tangent nodes.

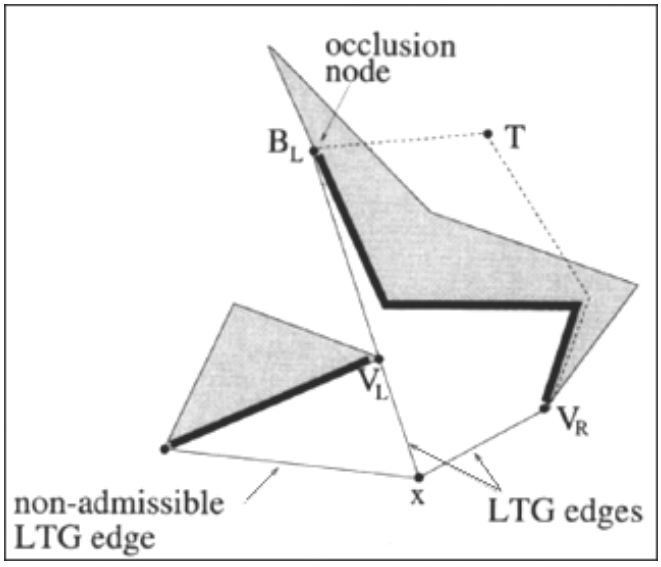


Figure 3.7 Image illustrating the concept of an occlusion node [86]

This new algorithm, adapted from the TangentBug algorithm [86] guides an autonomous robot from the start *S* to a target *T* while maximising traversal along the m-line, the line drawn from *S* to *T*. The sensor range is *r*. *t* denotes the threshold distance to an obstacle. It is an optional parameter with a default value equal to *r*. If $t < r$, the robot can travel along the m-line when it encounters an obstacle until it reaches a distance *t* from the obstacle, before using the LTG to circumvent the obstacle.

Algorithm 3 - Hybrid Bug algorithm

Note that the inputs to this algorithm are the start point S, the target T and the threshold nearest distance to obstacle t

Step 1: Move towards the target *T*, via the m-line when possible or along the locally optimal direction on the current LTG subgraph (as defined for the Hybrid Bug algorithm), until one of the following events occurs:

- The target is reached. The procedure has completed successfully.
- The robot reaches the m-line at a point T_{temp} from the left or right semi-plane. Go to Step 2.
- A local minimum is detected: Define a hit point *H* as the point on the junction of the m-line and the obstacle boundary.
    - If a threshold value *t* is defined that is less than the range *r* and the robot is on the m-line, proceed until the obstacle is at a distance *t* from the robot.

- Go to Step 3.

Step 2: Suspend motion toward the target behaviour and survey the direction of the m-line towards the start *S* for uncovered segments of the m-line. Travel down the m-line till the previous obstacle or the last hit point *H* is encountered and return to T_{temp}. The robot ensures keeping a distance *t* to the previous obstacle. Reset T_{temp} to *T*. Go to Step 1.

Step 3: Commence boundary following in the chosen direction. Move along the boundary using the LTG while recording d_{min} (*T*), until one of the following events occurs:

- The target is reached. The procedure has completed successfully.
- The leaving condition holds: $\exists V_{leave} \in$ LTG s.t. $d(V_{leave}, T) < d_{min}(T)$. Go to step 4.
- The robot completed a loop around the obstacle. The target is unreachable. The procedure is terminated.

Step 4: Perform the transition phase. Move directly toward V_{leave} until reaching a point *Z* that satisfies $d(Z, T) < d_{min}(T)$. Go to step 1.

Figure 3.8 gives a simplified flowchart representation of the Hybrid Bug algorithm.

It can be seen that Step 1 of the Hybrid Bug algorithm corresponds to the motion toward the target behaviour. Step 2 corresponds to the m-line following behaviour and Step 3 corresponds to the boundary following behaviour. Step 4 relates to the transition phase when the robot is switching from boundary following behaviour to motion toward the target behaviour.

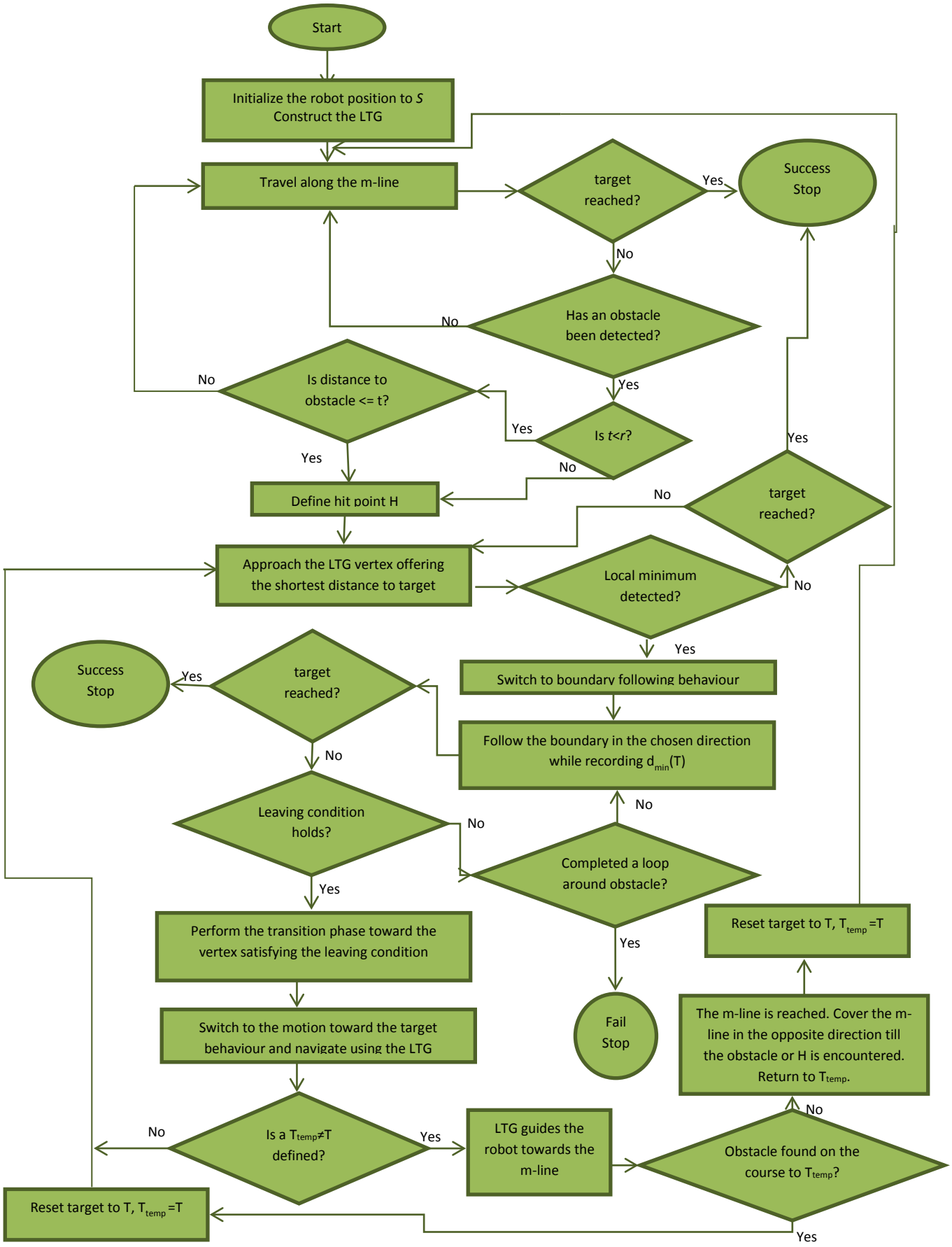


Figure 3.8 A flowchart summarising the Hybrid Bug algorithm

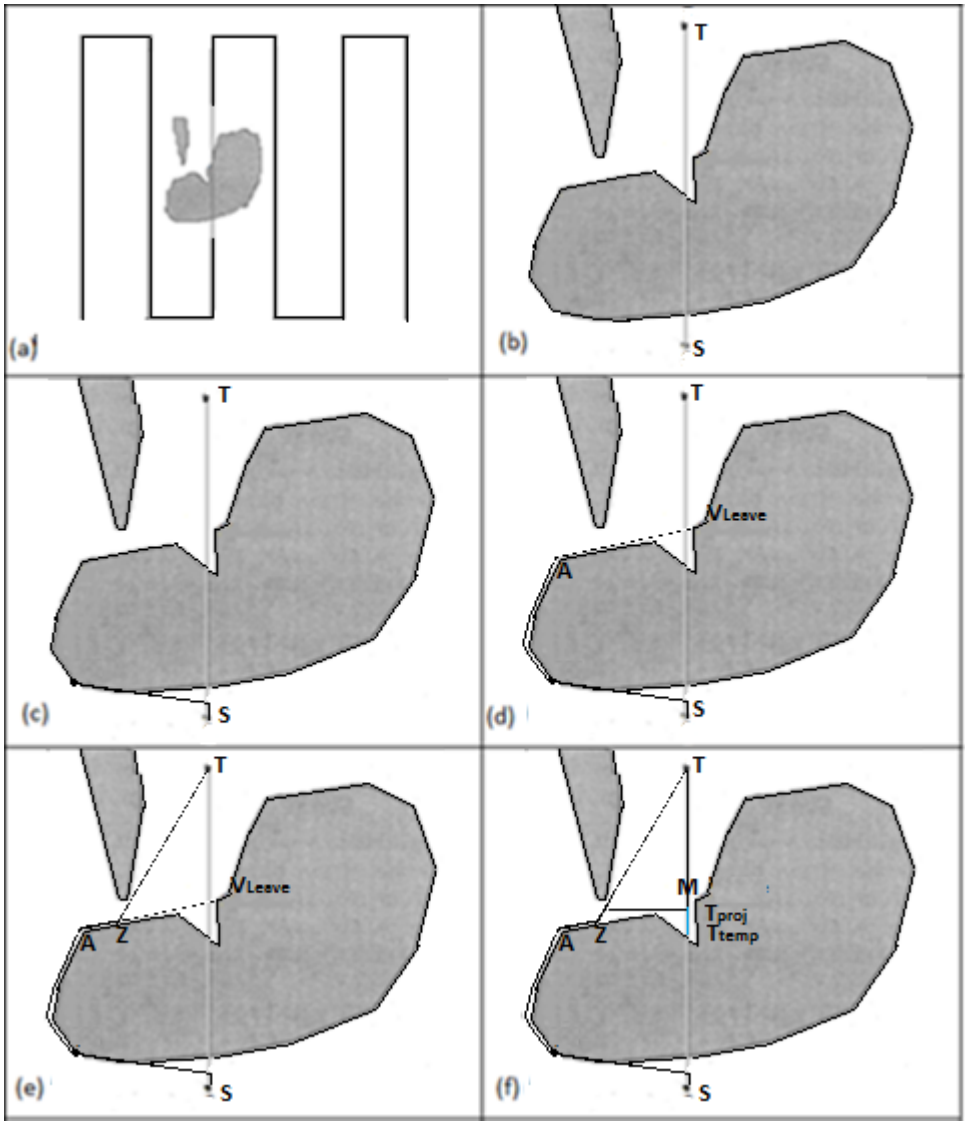


Figure 3.9 Path generated by the Hybrid Bug algorithm

A scenario showing the behaviour of path generated by the Hybrid Bug algorithm is shown in Figure 3.9. A sample path planned for area coverage is shown in Figure 3.9(a). The robot starts progressing towards the target *T* from the start *S* of one track of the planned path along the m-line in Figure 3.9(b). When it reaches a threshold distance *t* from the obstacle it proceeds towards *P*, which it perceives as the end point of the left shortest path (Figure 3.9(c)). The left shortest path is chosen as the distance to the target via the perceived end point on the left is shorter than that via the perceived end point on the right. At *P*, a local minimum is identified and so the robot switches to boundary following behaviour (Figure 3.9(c)). The boundary is followed until point *A*. At point *A*, the vertex V_{Leave} is identified that satisfies the leaving condition, i.e., a point V_{leave} is identified such that $d(V_{leave}, T) < d_{min}(T)$ and the robot proceeds towards it (Figure 3.9(d)). At point *Z*, the path to the target is no

longer obscured by the second obstacle and so the robot switches back to the motion toward the target behaviour, terminating the transition phase (Figure 3.9(e)). When an admissible edge leading to the m-line becomes available, the robot proceeds towards the m-line, meeting it at *M*. The blue section shows the length of the m-line covered during the m-line coverage behaviour. The robot then proceeds to the target *T* (Figure 3.9(f)).

As can be seen the key difference between this algorithm and the TangentBug algorithm is the concept of temporary targets along the m-line which alter the path during the motion toward the target behaviour. While the boundary following behaviour and the leaving condition remain identical, a minor difference is the coverage of the m-line in the opposite direction known as the m-line coverage behaviour. The introduction of the threshold *t* to allow the robot to go closer to an obstacle along the m-line, thereby increasing the distance travelled along the m-line is a second difference between the two algorithms. As a result, the Hybrid Bug algorithm attains a much higher coverage of the m-line which is desirable for complete area coverage.

3.4 Analysis of the new Hybrid Bug algorithm

This section analyses the new Hybrid Bug algorithm in greater detail and verifies its termination and completeness. Although the algorithm has potential applications in scenarios where obstacles are mobile due to its property of greedily approaching the m-line (If the robot veers off the m-line to avoid a mobile obstacle which moves away from the m-line, it will try to quickly resume m-line coverage), the analysis of the algorithm in the context of mobile obstacles is out of the scope of this research, as it is assumed that mobile obstacles such as humans and livestock will make way for the robot as it approaches them (research scope, Section 1.5). If not, they will be treated as stationary obstacles. Therefore, this analysis focusses on stationary obstacles.

The Hybrid Bug algorithm can be described as TangentBug algorithm having an affinity to approach the m-line along an admissible path whenever possible. Therefore, analysis is carried out along similar lines as in [86], with the robot is represented as a point travelling in the configuration space, which is the complement of the obstacles interiors. In addition to proposing new lemmas unique to the Hybrid Bug algorithm, the stepwise structure of the analysis takes the lemmas, propositions and theorems of the analysis of the TangentBug algorithm and examines their validity in the context of the Hybrid Bug algorithm.

Since the concept of T_{temp} is new, the analysis begins with this special point. From the summary of the algorithm, it follows that T_{temp} is used during the motion toward the target behaviour of Step 1 and not during boundary following behaviour to evaluate the leaving condition, which is in Step 3. The following lemmas, propositions and corollaries aim to prove that the Hybrid Bug algorithm always terminates.

The next lemma characterises the reachability of temporary targets, T_{temp}.

Lemma 3.4: Using an unlimited range sensor, if $T_{temp} \neq T$, the point T_{temp} is always reached.

Proof: T_{temp} is a temporary target unique to the Hybrid Bug algorithm. It is assigned a value as follows

$$\mathrm{T}_{\mathrm{temp}} = \begin{cases} \bullet\ T \text{ when the robot is on the m - line} \\ \bullet\ T \text{ when admissible points on the m - line segment } \overline{HT} \text{ are not visible from the current position of the robot} \\ \bullet\ T_{proj} \text{ when the approach to the m - line along the normal is clear of obstacles, } T_{temp} = T \text{ and } T_{proj} \in \overline{HT} \\ \bullet\ \text{Any point } D \in \overline{HT} \text{ such that } d(D,T) < d(H,T),\ D \text{ is in the line of sight and sensor range of the robot, } T_{temp} = T, \text{ there exists an obstacle along } \overline{DT} \text{ and the length of } \overline{\mathrm{DT}}_{\mathrm{proj}} \text{ is minimal} \end{cases} \tag{3.1}$$

From (3.1), T_{temp} has values other than T when it perceives that the approach to the m-line along the normal is obstacle-free or other points beyond the last encountered hit point H are perceived to be obstacle-free. In both of these cases, since the sensor has an unlimited range in the ideal case, the chosen points to represent T_{temp} on the m-line are actually visible from the current location of the robot and are therefore reachable along a straight line from the current location. As the algorithm prioritises reaching this point over finding a shortest path to T, the robot travels along the direction $\overline{xT_{temp}}$ where x is the current location of the robot. The definition of T_{temp} also guarantees that if the current value of $T_{temp} \neq T$, it is not overridden by newer values of T_{temp} until the robot reaches the m-line at the point T_{temp}.

The following corollary asserts that subsequently defined values of temporary targets are always closer to the target than those previously assigned.

Corollary 3.2: Using unlimited sensor range, new values of $T_{temp} \neq T$ are always closer to T than previously defined values of $T_{temp} \neq T$.

Proof: As per Lemma 3.4, any point $T_{temp} \neq T$ is always reached. Since T_{temp} always lies on the m-line, every definition of $T_{temp} \neq T$ must have a corresponding hit point H which is the point at which the next obstacle intersects the m-line, unless the path to the target T along the m-line is clear of obstacles. If the path to T is clear of obstacles, no new values will be assigned to T_{temp} as Step 1 ensures this. Otherwise, if T_{temp_i} and H_i denote the i^{th} instance of T_{temp} and the following hit point H, it is known that $d(T_{temp_i},T) > d(H_i,T)$ as distance to target decreases monotonically along the m-line. Any subsequent definitions of T_{temp}, T_{temp_i+1} will happen during motion toward the target behaviours, and will have to lie on the line segment $\overline{H_iT}$ as defined in (3.1), and therefore would be closer to T than H_i. Therefore, $d(T_{temp_i},T) > d(H_i,T) > d(T_{temp_i+1},T)$.

The following lemma verifies that the robot continually approaches the target during the motion towards the target behaviour.

Lemma 3.5*:* The distance of the robot from the target, $d(x, T)$, decreases monotonically during each motion toward the target segment and between successive motions toward the target.

Proof: By definition of the algorithm, during motion toward the target, the robot moves along admissible edges, along which $d(x, T_{temp})$ decreases. From (3.1), it is clear that since $T_{temp} = T$ when on the m-line or when the m-line segment $\overline{HT}$ is not visible to the robot, and so $d(x, T)$ decreases in these cases. In the other 2 cases listed in (3.1), when the robot approaches the m-line, it does so along a direction that is admissible and the condition for admissibility ensures that $d(x,T)$ decreases (Appendix IV). Therefore $d(x,T)$ decreases in all cases.

As in the case of the TangentBug [86], the distance $d(x, T)$ also decreases between successive motion toward the target segments, since the leaving condition guarantees that $d(P_i, T) \geq d_{min}(T) > d(Z_i, T)$, where P_i is the endpoint of the previous motion-toward-the-target segment (where the robot switched to boundary following) and Z_i, where the robot ends the transition phase, is the start point of the next segment. Figure 3.9 shows one instance of P_i and Z_i labelled P and Z respectively.

The following definition of event types is necessary for the systematic analysis of the path taken by the Hybrid Bug algorithm, in order to prove that it terminates. These events are seen to be associated with direction changes.

Definition 3.1: Three types of events are defined to occur when a robot approaches a focus node [86] :

- Type 1: The robot reaches the focus node *F* and chooses a new focus node.
- Type 2: The optimal path to *T* via the m-line on the augmented graph ceases to pass through F, and a new focus node is selected.
- Type 3: The edge of the current focus node ceases to be admissible, and the edge with its endpoint *F* is removed from the subgraph *G1*.

As is evident, these events occur due to changes in the robot's positioning and perspective. Illustrations for each of these event types are given below.

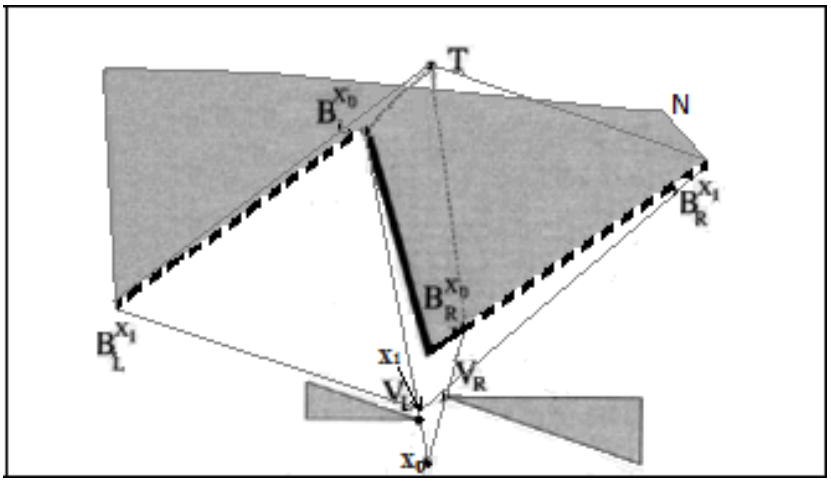


Figure 3.10 Illustration of Type 1 and Type 2 events (Illustration created based on examples in [86])

In Figure 3.10, the grey area represents the obstacle, the solid line represents what is visible to the robot from the point x_0 and x_1. The dashed line represents the portion of the obstacle boundary that is visible from x_1 only. When the robot is at x_0, $B_L^{X_0}$ is the end point of the blocking obstacle that is visible to the robot on the left semi-plane. Similarly, $B_R^{X_0}$ is the end point visible from x_0 on the right semi-plane. At this point it appears as if the path via $B_L^{X_0}$ would have the shortest distance to the target and the robot continues to x_1, with $B_L^{X_0}$ as the focus node F. However, here a greater part of the obstacle boundary is visible to the robot with $B_L^{X_1}$ being the left end point of the sensed obstacle boundary and $B_R^{X_1}$ being the right end point. When the robot re-evaluates distances to the target from its current position, the path via $B_R^{X_1}$ appears to be the shorter path and therefore the focus node changes to $B_R^{X_1}$. This

change is an example of a Type 2 event. The direction remains constant until the robot reaches $B_R^{X_1}$. Once the robot reaches the current focus node $B_R^{X_1}$, a new focus node, which is *N* in this case is chosen. This is a Type 1 event.

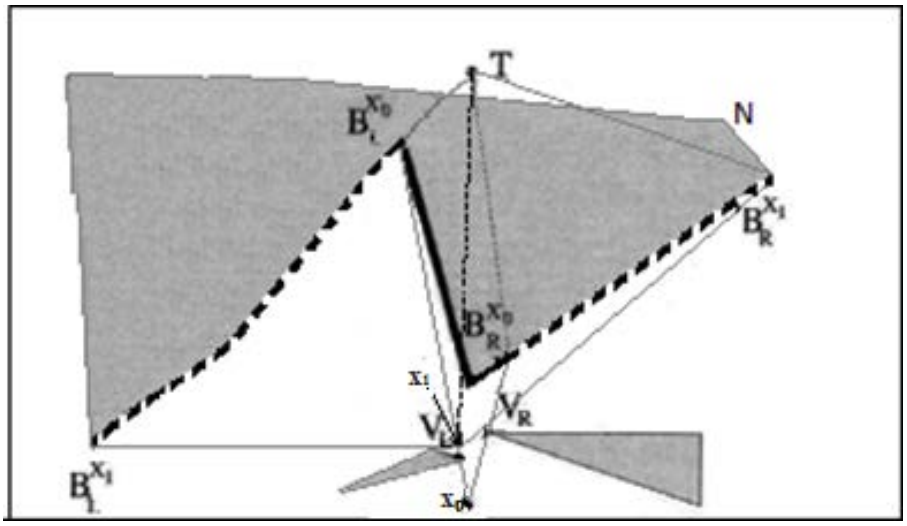


Figure 3.11 Illustration of a Type 3 event (Illustration created based on examples in[86])

In Figure 3.11, the grey area represents the obstacle, the solid line represents what is visible to the robot from the point x_0 and x_1 and the dashed line represents the portion of the obstacle boundary that is visible only from x_1. When the robot is at x_0, $B_L^{X_0}$ is the end point of the blocking obstacle that is visible to the robot on the left semi-plane and $B_R^{X_0}$ is the sensed end point of the blocking obstacle on the right semi-plane. Once again, the path through $B_L^{X_0}$ is evaluated to be shorter initially and therefore, it is chosen as the focus node. However, when the robot reaches x_1 as it approaches $B_L^{X_0}$, a greater part of the obstacle boundary becomes visible and from the current position x_1, it becomes apparent that the edge connecting x_1 to the left end point of the obstacle boundary, $B_L^{X_1}$, now subtends an angle greater than 90° with the line connecting x_1 to the target *T*. This makes the edge non-admissible and it is removed from the subgraph *G1*. A new focus node - $B_R^{X_1}$ is chosen causing a change in the robot's direction. A Type 3 event is said to have occurred.

Using the concept of the three event types, the following lemma asserts that the Hybrid Bug algorithm generates a piecewise-linear path.

Lemma 3.6*:* Using unlimited sensor range, the Hybrid Bug algorithm generates a piecewise-linear path.

Proof*:* During motion toward target (via m-line) behaviour of motion (Step 1 of the algorithm), the robot moves toward the current focus node *F*. The focus node remains the same obstacle vertex or m-line point until an event of the three types in Definition 3.1 occurs.

Kamon et al. [86] show that these events only occur at discrete points along the path to the target and the following lemmas: Lemma 3.7, Lemma 3.9, Lemma 3.10 and Lemma 3.11 discuss the validity of this statement in the context of the Hybrid Bug algorithm. Therefore, the path is piecewise linear in the motion toward the target behaviour. In the m-line coverage behaviour (Step 2 of Algorithm 3), the direction is fixed towards S, until the robot reaches encounters an obstacle or the previous hit point *H*, when it makes a 180° turn to return to the temporary target where it began this behaviour. Therefore, only 2 direction changes occur and the path during this behaviour is also piecewise-linear. In the boundary following behaviour (Step 3 of Algorithm 3), the direction is fixed, and changes only at the obstacle vertices or gradually changes with the obstacle's curvature. Hence the robot's path during this behaviour is also piecewise linear. During the transition phase from boundary following to motion toward the target, the robot moves in a straight line from a leave point L_i to the transition point Z_i.

Next, the number of direction changes associated with each of the three types of events is analysed, starting with Type 1 events.

Lemma 3.7: During motion toward the target using an unlimited range sensor, there are finitely many direction changes due to Type 1 events.

Proof: Lemma 3.5 shows that the distance of the robot from the target, *d(x, T)*, decreases monotonically along motion toward the target segments. Thus the robot may visit any obstacle vertex at most once. Also, from Corollary 3.2, subsequent values of T_{temp} have to be closer to *T* than previously defined values. From Lemma 3.4, since every temporary target $T_{temp} \neq T$ is reached, there exists at most 1 definition of a temporary target $T_{temp} \neq T$ per obstacle intersection of the m-line and therefore the number of temporary targets defined on the m-line are finite. Since Type 1 events occur when the robot reaches a convex obstacle vertex or when the robot reaches a temporary target T_{temp} on the m-line, the number of these events is finite.

Next, the occurrence of Type 2 events and associated direction changes are analysed. When an obstacle lies between the robot and the target, it is known as the blocking obstacle. The left and right endpoints of the obstacle are denoted by B_L and B_R[86]. The first node on the left is V_L and that on the right is V_R[86]. This is illustrated in Figure 3.12 (b). If the robot moves left to circumvent the obstacle, the path is called the *left optimal path*, otherwise it is called the *right optimal path* [86] .

Lemma 3.8: When a blocking obstacle exists, the augmented graph contains a left path if there is an admissible straight line path to the m-line segment $\overline{HT}$ on the left of the robot or if the edge that connects the current robot position x with B_L is admissible, in which case, the left optimal path passes through B_L. A similar result holds for m-line on the right or the right optimal path, which passes through B_R.

Proof: When the HybridBug algorithm is confronted by a blocking obstacle between itself and the target T, there are 2 possibilities:

Case 1. The robot is currently on the m-line: In this case, the optional T_{node} is absent and the robot constructs an LTG that is identical to that of the TangentBug algorithm and attempts to construct the shortest path to T. Kamon et al. [86] state that the shortest path to T taken by the TangentBug, considering only the visible obstacles, always passes through an endpoint of the blocking obstacle. Therefore, if the edge $\overline{xB_L}$ is admissible, the left shortest path must pass through this edge. If the edge $\overline{xB_L}$ is not admissible, the angle α between the edge $\overline{xB_L}$ and the line segment $\overline{xT}$ will be obtuse. Consequently, the LTG edges to the left of $\overline{xT}$ are also non-admissible, since the angles between these edges and the line $\overline{xT}$ exceed 90°. Hence there is no left path on the augmented graph. A similar argument holds for the right path.

Case 2. The robot is in the right or left semi-plane of the m-line: Without loss of generalisation, the case of the robot in the right semi-plane with the m-line on the left is discussed here and would apply to the other case as well. When the robot is in the right semi-plane, if there is an admissible straight line path to $\overline{HT}$ where H is the hit point where the obstacle was last encountered along the m-line, it is taken. However, if such a path does not exist, the T_{node} also does not exist. The LTG becomes identical to that in the case of the TangentBug algorithm and the analysis given in Case 1 above holds.

In keeping with the discussion in [86], the next two lemmas discuss the occurances of Type 2 events. The length of a virtual edge from a node V of $G1$ to T is denoted $e_len_x(V)$. Unlike the distance function, $d(x, T)$, this notation also enables the representation of the robot's current position x in the length function, which plays a key role in the perceived distance. The distance from the current position x to any vertex V, is denoted by $\|x - V\|$.

Lemma 3.9: When using unlimited range sensors, the events of Type 2 occur only when an LTG node, which is a tangent node, optional node, or an occlusion node, is removed or added to the LTG.

Proof: When the robot is in motion and no nodes are added to the LTG, the nodes can either remain fixed or some occlusion nodes slide along obstacle edges. Considering the case where the nodes are fixed, in the special case where T_{node} exists in the direction of $T_{temp} \neq T$, it will be the current focus node F. Since it is always reached according to Lemma 3.4, the direction remains constant until the node is reached, when a Type 1 event eventually directs it towards T after the deletion of the current T_{node}.

However, if there are no candidate nodes $T_{temp} \neq T$ or the robot is currently on the m-line, the algorithm operates similar to the TangentBug algorithm, attempting to find the shortest path to T. Suppose that the robot moves from x_0 to x_1 in a way that keeps all the LTG nodes fixed (an example is illustrated in Figure 3.12 (a)). Without loss of generality, an assumption is made that the shortest path from x_0 to T on the augmented graph is the left path. Since the left path is shorter than the right path, Kamon et al. [86] state,

$$\|x_0 - V_L\| + e_len_{X_0}(V_L) < \|x_0 - V_R\| + e_len_{X_0}(V_R) \quad (3.2)$$

Since all the LTG nodes remain fixed as the robot moves from x_0 to x_1, the left and right paths from x_1 pass through the same nodes V_L and V_R. Moreover, the lengths of the expected paths from V_L and V_R to T do not change as the robot moves from x_0 to x_1. Substituting $e_len_{X_0}(V_L) = e_len_{X_1}(V_L)$ and $e_len_{X_0}(V_R) = e_len_{X_1}(V_R)$ in eq. (3.2) gives

$$\|x_0 - V_L\| + e_len_{X_1}(V_L) < \|x_0 - V_R\| + e_len_{X_1}(V_R) \quad (3.3)$$

The term $\|x_0 - V_L\|$ on the left side is substituted with $\|x_0 - V_L\| = \|x_1 - V_L\| + \|x_0 - x_1\|$, since the points x_0, x_1, and V_L are collinear, giving

$$\|x_1 - V_L\| + e_len_{X_1}(V_L) < \|x_0 - V_R\| - \|x_0 - x_1\| + e_len_{X_1}(V_R)$$

Since $\|x_0 - V_R\| < \|x_0 - x_1\| + \|x_1 - V_R\|$ according to triangle inequality,

$$\|x_1 - V_L\| + e_len_{X_1}(V_L) < \|x_1 - V_R\| + e_len_{X_1}(V_R)$$

Thus the left path from x_1 is shorter than the right path from x_1, and the direction of motion remains toward V_L.

In the absence of a temporary target, $T_{temp} \neq T$, Type 2 events also do not occur when an occlusion node slides continuously along an obstacle edge, as described in [86] : without loss of generality, an assumption is made that the shortest path from x_0 to T on the augmented

graph is the left path. Suppose that B_R is an occlusion node which slides on an edge of the blocking obstacle.

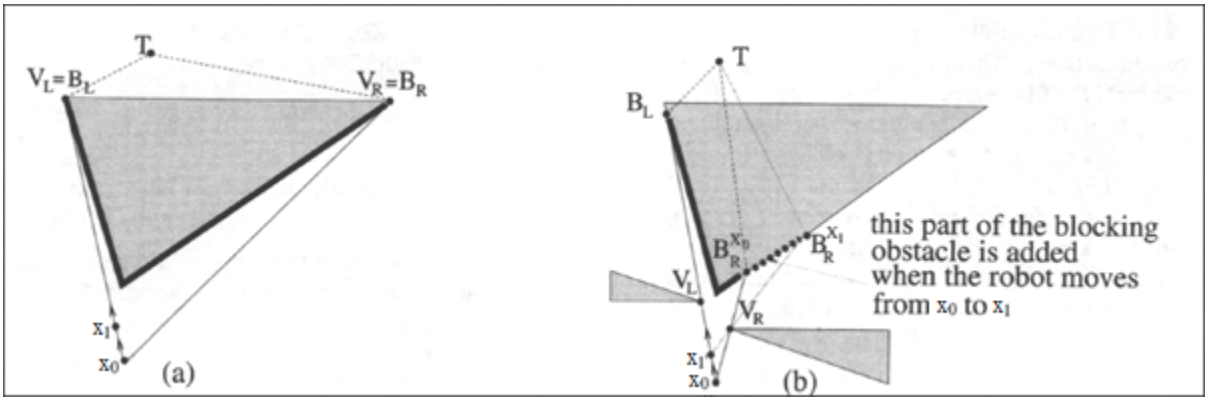


Figure 3.12 NoType 2 events occur when (a) LTG nodes remain fixed (b) occlusion nodes slide along the obstacle boundary [86]

As illustrated in Figure 3.12, the point B_R as seen from x_0 and x_1 are represented by $B_R^{X_0}$ and $B_R^{X_1}$. The part of the blocking obstacle boundary that is visible from x_1, (B_L, $B_R^{X_1}$) includes all the visible boundary from x_0, (B_L, $B_R^{X_0}$), and extends to the right side of (B_L, $B_R^{X_0}$). Therefore the path length from V_R to T via $B_R^{X_1}$ is longer than via $B_R^{X_0}$ (The same argument holds for more complex situations, where the shortest path from $B_R^{X_1}$ to T is not the straight line $\overline{B_R^{X_1}T}$). Hence the length of the shortest path from B_R to T increases when the robot moves from x_0 to x_1, i.e. $e_len_{X_1}(V_R) > e_len_{X_0}(V_R)$. According to Lemma 3.8, since the left path leads directly toward the blocking obstacle vertex B_L, one has $e_len_{X_1}(V_L) = e_len_{X_0}(V_L)$. Equation (3.2) holds at x_0, and consequently equation (3.3) holds true. Hence, using an argument similar to the one used above, the left path remains shorter than the right path at x_1, and the robot retains its direction toward B_L.

Lemma 3.10: During motion toward the target using an unlimited range sensor, there are finitely many successive direction changes due to Type 2 events.

Proof: For a given obstacle edge e, let the *extended edge* of e be the line segment that extends from e into the free space. According to Lemma 3.9, Type 2 events occur only when an LTG node is removed or added to the LTG. Such a change in the LTG can occur in one of the following three ways.

Case 1. When the robot crosses an extended edge, tangent nodes, optional nodes and occlusion nodes can change.

Case 2. When the robot crosses a *bitangent edge*, which is the line segment in the free space tangent to two convex obstacle vertices, only optional nodes and occlusion nodes can change.

Case 3. When the robot crosses any extended line $\overline{DC}$ where D is any candidate point for a temporary target T_{temp} on the unblocked part of the m-line and C is a convex obstacle vertex, optional nodes can be added.

First, the cases are considered in the context of the features exclusive to the Hybrid Bug Algorithm: When a candidate point $T_{temp} \neq T$ becomes visible on the m-line, either upon crossing a bitangent edge, extended edge or any line $\overline{DC}$ where D is any candidate point for a Temporary target $T_{temp} \neq T$ on the unblocked part of the m-line and C is a convex obstacle vertex, the optional node T_{node} gets added to the LTG and a Type 2 event causes a change in direction. A Type 2 event will direct the robot to T_{temp} in such events, and Type 1 events will carry it forward along the m-line. According to Lemma 3.7, there exists at most 1 definition of a temporary target $T_{temp} \neq T$ per obstacle intersection of the m-line and therefore the number of temporary targets defined on the m-line are finite. This implies that Type 2 events arising out of the addition of a T_{node} during motion toward the target behaviour are finite in number.

Next, the case where the robot crosses a bitangent edge, i.e. Case 2, is considered. If this action causes a candidate point for T_{temp} to become visible on the m-line, the optional node T_{node} gets added to the LTG and a Type 2 event causes a change in direction. However, if no candidates for T_{temp} are available, the scenario is as in [86]: according to Lemma 3.8, the candidate shortest paths always lead directly to the endpoints of the blocking obstacle. Hence the direction of motion can change only when the occlusion node which is removed or added to the LTG is one of the endpoints of the blocking obstacle. If an occlusion node appears on the blocking obstacle, the obstacle configuration must be such that just before the crossing the bitangent edge, some obstacle was obscuring the blocking obstacle. However, this contradicts the definition of the blocking obstacle, which is visible from the current robot location. Hence the crossing of a bitangent edge cannot cause the addition of an occlusion node to the blocking obstacle. The crossing also cannot cause the removal of an occlusion node either, and only transforms an occlusion node into a tangent node in the LTG. Since the location of the node does not change in this event, there is no change in the shortest path to T, and the robot continues in the same direction of motion. Thus there can be at most one

change of direction when the robot crosses a bitangent edge, and that is due to the addition of an optional node. Since the number of bitangent edges is finite, the number of Type 2 events arising out of crossing bitangent edges is also finite.

If the crossing of an extended edge, Case 1, does not cause a candidate point T_{temp} and hence T_{node} to become visible, the algorithm behaves like the TangentBug algorithm. The summary of the proof is described here. A more detailed version can be found in [86]. The robot always moves toward an endpoint of the blocking obstacle. It is supposed that while moving toward the left endpoint B_L, the robot crosses an extended edge. It is also supposed that the crossing causes a change in an LTG node which triggers a change of direction. Using Lemma 3.8 again, the candidate shortest paths always lead directly to the endpoints of the blocking obstacle. Thus, only a change in B_L or B_R can possibly cause a change of direction. Since the robot was moving toward B_L, the other endpoint, B_R, must have changed when the robot crossed the extended edge. Furthermore, the change in the LTG nodes must have occurred along the extended edge. Hence, immediately after the crossing, the robot must be moving toward B_R along the extended edge. The robot then traces the extended edge until it crosses another extended edge, or until an event of some other type (Type 1 or Type 3) occurs. Since the extended edges form a fixed arrangement of lines with a finite number of intersection points, the number of successive direction changes due to Type 2 events is finite.

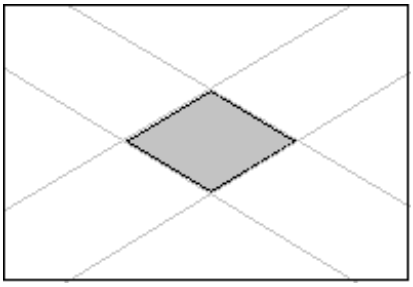


Figure 3.13 Extended edges partitioning the free space into cells

Lemma 3.11: During motion toward the target, there are finitely many Type 3 events.

Proof: Since it is guaranteed that any definition of $T_{temp} \neq T$ is reached along an admissible path on priority and since T_{temp} is in the line of sight of the robot, in the presence of $T_{temp} \neq T$, the admissibility of the direction to T_{temp} is not evaluated as the robots moves towards it. This rules out direction changes owing to Type 3 events in the context of the temporary target. The remainder of the proof follows from [86] and is summarised as follows: Let a *blocking extended edge* be an extended edge that starts at a convex obstacle vertex of the blocking obstacle. The blocking extended edges partition the free space into cells as shown in Figure

3.13. The non-generic case where the robot is crossing a blocking extended edge such that a Type 3 event occurs at the crossing point is ignored.

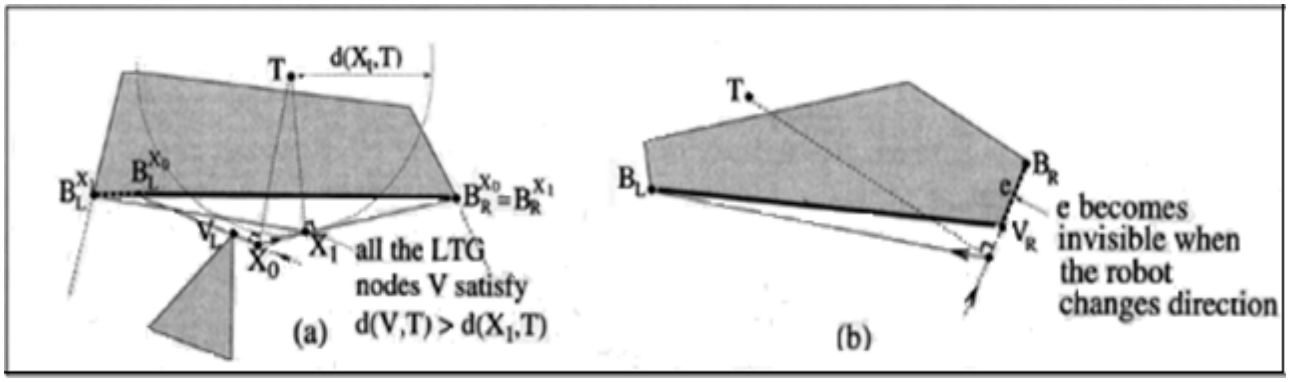


Figure 3.14 Robot moving (a) within a cell (b) along blocking extended edges [86]

First, the motion in the interior of a cell, illustrated in Figure 3.14 is considered. It will be shown that at most one Type 3 event occurs in the interior of a cell during each motion toward the target segment. Suppose a Type 3 event occurs at the point x_0 while the robot is moving toward the node $B_L^{X_0}$, since the edge $\overline{x_0 B_L^{X_0}}$ ceases to be admissible, x_0 is a minimum of the distance function *d(w, T)* along $\overline{x_0 B_L^{X_0}}$. In particular, $d(B_L, T) > d(x_0, T)$. Since the angle a between the edge $\overline{x_0 B_L^{X_0}}$ and the line segment $\overline{x_0 T}$ is 90°, all the LTG edges to the left of $\overline{x_0 B_L^{X_0}}$ also become non-admissible. Thus all the LTG nodes *V* to the left of the line segment $\overline{x_0 T}$ satisfy $d(V, T) > d(x_0, T)$. Now the robot continues toward the right endpoint B_R and reaches a point x_1, which still lies in the interior of the cell. Since the distance to *T* decreases during motion toward the target, $d(x_1, T) < d(x_0, T)$. Assume by contradiction that a Type 3 event also takes place at x_1. Using an argument similar to the one given above, all the LTG nodes *V* to the right of $\overline{x_1 T}$ satisfy $d(V, T) > d(x_1, T)$. Since the robot has not crossed any blocking extended edge, the region in the free space bounded by the edges $\overline{x_0 B_L^{X_0}}$, $\overline{x_0 B_R^{X_0}}$ and the blocking obstacle are also visible from x_1 (Figure 3.14 (a)). Moreover, there are no LTG nodes in the interior of this region. This information, together with the inequality $d(x_1, T) < d(x_0, T)$, implies that all the LTG nodes *V* at x_1 satisfy $d(V, T) > d(x_1, T)$. However, this is the condition that terminates the motion toward the target behaviour and initiates the boundary following behaviour. Hence the robot must

have switched to boundary following behaviour at some earlier point. Thus, at most one Type 3 event can occur in the interior of a cell.

Next, the case where a Type 3 event occurs while the robot is moving along a blocking extended edge is considered. It is known that the robot must be moving toward one of the endpoints of the blocking obstacle. Suppose it is moving along a blocking extended edge toward the right endpoint B_R, as illustrated in Figure 3.14(b). By definition of a blocking extended edge, there exists an edge, e, of the blocking obstacle along the blocking extended edge. After the Type 3 event occurs, the robot's new direction must be toward the left endpoint B_L. Since the robot leaves the blocking extended edge and moves toward B_L, the obstacle edge e must become invisible, as shown in Figure 3.14(b). Thus, each time the robot moves away from a blocking extended edge, some obstacle edge, e, becomes invisible. Further, until the robot reaches one of the endpoints of the blocking obstacle (a Type 1 event), an obstacle edge that has become invisible cannot become visible again. Now it is possible to summarise the two cases and say that the number of Type 3 events until a Type 1 event occurs is finite. Since according to Lemma 3.7 the number of Type 1 events is finite, the total number of Type 3 events is also finite.

Proposition 3.4: Using unlimited sensor range, each motion toward the target(via the m-line) segment has finite length.

Proof: The same reasoning for TangentBug algorithm [86] can be applied to the Hybrid Bug algorithm as well: According to Lemma 3.6 the Hybrid Bug algorithm generates a piecewise linear path. Hence, it suffices to show that the path consists of finitely many linear segments of finite length. Each linear segment corresponds to a particular direction of motion. According to Lemma 3.7, the number of direction changes due to Type 1 events is finite. According to Lemma 3.11, the same holds true for Type 3 events. According to Lemma 3.10, there are finitely many successive Type 2 events. Type 2 events are succeeded either by Type 1 or Type 3 events, or by termination of the motion toward the target. Thus, each motion toward the target segment has finitely many Type 2 events. Therefore, the number of linear segments in each motion toward the target segment is finite. The length of each linear segment is finite, since the robot moves toward a fixed focus node.

The following proposition, lemma and corollary are identical to [86] in their proofs.

Proposition 3.5: Using unlimited sensor range, each boundary following segment has finite length. Moreover, if the target *T* is reachable from the point where the robot switches to

boundary following behaviour, the leaving condition will cause the robot to leave the obstacle.

Lemma 3.12: Every switch point P_i, where the robot switches from motion toward the target behaviour to boundary following behaviour, has a corresponding local-minimum point M_i of the distance function $d(w, T)$, such that $d(M_i, T) < d(P_i, T)$.

Corollary 3.3: Let M_i be the local minimum of $d(w, T)$ associated with the switch point P_i. Then the distance to the target decreases between successive local-minimum points, that is, $d(M_{i+1}, T) < d(Mi, T)$.

For proofs to the above proposition, lemma and corollary, refer to [86]. With all the necessary lemmas, propositions and corollaries discussed, the proof of termination of the Hybrid Bug algorithm is now presented.

Theorem 3.4: Using an unlimited range sensor, the Hybrid Bug algorithm always terminates.
Proof: According to Lemma 3.12, every switch point P_i is associated with a local-minimum point M_i of $d(w, T)$. According to Corollary 3.3, $d(M_i,T)$ decreases at successive local-minimum points. Thus each point M_i is associated with at most one switch to the boundary following behaviour. Since the number of local minima of $d(w, T)$ is finite, the path consists of finitely many boundary following segments. Since two consecutive boundary following segments are interleaved by a single motion toward the target (via the m-line) segment, a possible m-line coverage segment and a single transition segment, there are also finitely many motion toward the target and transition segments. Next, considering the length of each motion segment, Proposition 3.4 guarantees that the path length of each motion toward the target (via the m-line) segment is finite. The path length of each m-line coverage segment is finite since it is bounded by the current value or $T_{temp} \neq$ T and the previous obstacle. A value of $T_{temp} \neq$ T is always assigned to T_{temp} when the robot is not on the m-line and a diversion from the m-line is necessitated by the presence of an obstacle along the m-line. Therefore an obstacle must exist in the direction of the start point *S*. In the unlikely event of the obstacle having moved away from the m-line, the algorithm instructs this behaviour to stop at *S* and return to T_{temp} along the m-line. Therefore, the path length of this segment is finite. The path length of each transition phase is finite because the robot moves toward a fixed focus point. Proposition 3.5 guarantees that the path length of each boundary following segment is finite. Therefore, the Hybrid Bug algorithm terminates after a finite length path.

The following theorem proves that the Hybrid Bug algorithm is complete and hence convergent.

Theorem 3.5: Using unlimited sensor range, Hybrid Bug always finds the target if it is reachable from the start point S.

Proof: As discussed in the proof of Theorem 3.4, there are finitely many boundary following segments. If the target *T* is reachable from the start point *S*, Proposition 3.5 guarantees that the leaving condition will cause the robot to terminate each boundary following segment and leave the obstacle boundary. Since every such segment is followed by a transition phase, there always exists a final transition phase. The final transition phase either terminates at the target *T*, or is followed by the last motion toward the target (via the m-line) segment, which terminates at *T*.

Thus the Hybrid Bug algorithm has been shown to terminate and converge in the ideal case of infinite sensor range, as is the case with the TangentBug algorithm upon which it is partially based. The integrity of the component algorithms have therefore been preserved.

3.5 Conclusion

The main aim of developing a new obstacle avoidance algorithm was to enable effective obstacle avoidance while maximising area coverage in the target scenario, where the shapes and position of the obstacles were unknown, but it was expected that the obstacles were small and navigable area was large in comparison. The Hybrid Bug algorithm was developed to efficiently merge the Bug2 algorithm with its property of extensively covering the pre-set path connecting the start and target points (the m-line) with the TangentBug algorithm that possesses effective obstacle avoidance features. Consequently, it was found that the resulting algorithm was very efficient as

1. It does not trace the boundaries of the obstacle closely, as in the case of the Bug2 algorithm. This results in a greatly reduced path length.
2. It uses local LTG shortcuts to further optimise path length.
3. Unlike the Bug2 algorithm, the Hybrid Bug algorithm does not have a fixed direction of obstacle circumvention and bases the direction of circumvention on the available local sensor information.
4. It uses the concept of temporary targets along the m-line – as a result greedily approaches the m-line, maximising the time spent on the planned path thereby increasing area coverage.

5. Upon reaching the m-line from the left or right semi-plane, the Hybrid Bug algorithm directs the robot to trace the m-line back to the obstacle, further increasing coverage of the m-line. This is the third and new behaviour incorporated into the algorithm, known as the m-line coverage behaviour.
6. A new parameter *t* is defined as the minimum distance that needs to be maintained from an obstacle. This is a configurable parameter. This enables the robot go closer to the obstacle along the m-line than range sensor based obstacle avoidance algorithms such as the TangentBug algorithm, which in turn increases the extent to which area coverage is achieved.

It has been proven that in the case where the sensor range is unlimited, this newly developed algorithm not only terminates, but also guarantees that any target that is reachable from the start point is reached. As a result, the Hybrid Bug algorithm guarantees the same level of confidence that the Bug2 and TangentBug algorithms have. Therefore, a new robust algorithm has been custom made to avoid obstacles during the complete area coverage of agricultural areas.

4 Robot Sensors and Navigation

4.1 Introduction

The main aim of this research was to explore complete area coverage strategies to be used by an autonomous robot. It aimed at developing a strategy that maximised the amount of area covered while minimising the path length the robot took to achieve it. The Hybrid Bug algorithm presented in Chapter 3 produces a shorter path during obstacle circumvention than contemporary algorithms such as the Bug1 and Bug2 algorithms. Simulations carried out in Chapter 5 show a comparison of path lengths of the Bug2 and Hybrid Bug algorithms. However, the algorithm presented in Chapter 3 considers the ideal case of unlimited sensor range and infinite angular resolution of the component sensors. This chapter deals with the real world aspects of the sensors involved. Constraints on the cost of the robot and demands to operate it both indoors and outdoors influenced the choice of sensors that would be used on a prototype that in turn influenced the area coverage strategy, which will be discussed in this chapter. Additionally, the area coverage context in which the Hybrid Bug algorithm would operate is also discussed in detail in this chapter.

Since the resulting strategy would be employed by autonomous robots primarily catering to the agricultural sector, a robotic prototype was developed to get a clearer understanding of what the low cost system may comprise of (Appendix III). While it was decided that ultrasonic sensors would be used for navigation purposes, the monitoring operations for which area coverage is carried out would be performed primarily by a vision system amongst other sensors. As part of the initial testing process, a survey of economical outdoor cameras was conducted and it was concluded that the Brickcom OB-100Ap megapixel outdoor camera would be a good choice of an image processing camera. Appendix I discusses that when placed on a typical height on a robotic platform, the field of view is about 1.7 m.

One way of implementing the area coverage is to use a boustrophedon coverage pattern. The boustrophedon track width decides the level of overlap that occurs between visually inspected segments. Having a proper overlap between visually inspected segments gives a more efficient coverage for regions shadowed by obstacles as shown in Figure 4.1. In this research it is assumed that for a 1.7m field of view of the on-board camera, a track width of 1m ensures a desirable 70cm overlap. The choice of a 1m track width is based on the field of

view of the Brickcom OB-100Ap camera and can be widened or narrowed without loss of generality.

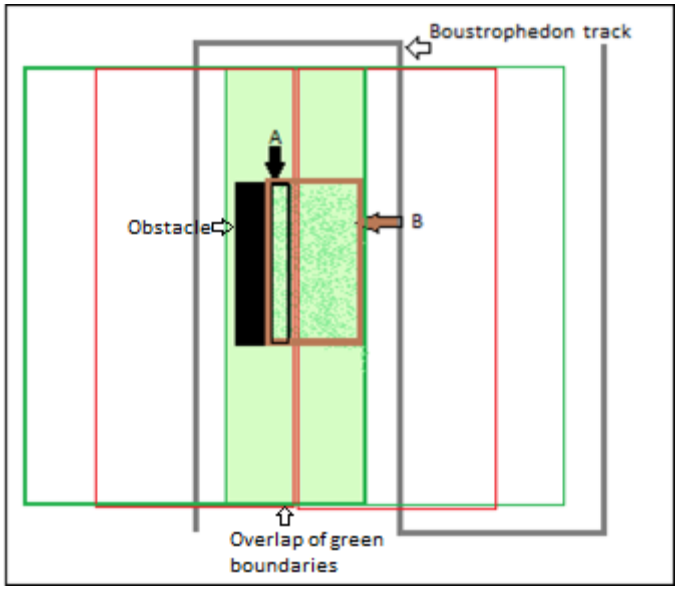


Figure 4.1 Shadow regions with and without overlapping coverage sections

Having no overlapping regions can cause regions shadowed by obstacles to be missed, as indicated by the section marked A in Figure 4.1, where the visually inspected regions are shown in red. In the case where the visually inspected regions are shown in green, there is significant overlap and the shadow region also gets inspected (Figure 4.1 B). The boustrophedon area coverage strategy proposed in this chapter allows the track width to be configured. So, for a robot equipped with a vision system as above, a 1m track width would be ideal. While considering typical agricultural applications where this research could be applied, such as free range chicken farms or grazing areas, the ratio of obstacle size to navigable space is extremely small. Further, the width of typical obstacles such as trees, feeding and watering stations, tractors etc. are usually comparable to the track width. The proposed area coverage strategy performs optimally when the widths of the obstacles are smaller than three track widths, although this is not a rigid constraint.

The effect of the chosen low cost ultrasonic sensors on the new Hybrid Bug algorithm developed in Chapter 3 is discussed in this chapter. These sensors have limited angular resolution which is a hindrance especially to the boundary following behaviour of the Hybrid Bug algorithm. Methods to overcome this limitation are discussed. Finally, the process of integrating these methods with the Hybrid Bug algorithm and using it to avoid obstacles along a strictly boustrophedon area coverage path is described.

4.1.1 Constraints posed by the ultrasonic sensor, SRF05

The constraints imposed on the cost and outdoor usability of the robot incited the use of ultrasonic sensors for robot navigation. The SRF05 sensors are extremely popular amongst hobby roboticists and professionals since they are cost effective, easily sourced and therefore replaceable and provide excellent ranging. As stated in 2.4.1, they suffer from poor angular resolution, since the distance to the point closest to the sensor along the span of their beam pattern is always returned by the sensor. With the sensed portion of the beam spanning 55° [89], these sensors have low azimuth resolution.

One option would be to use other low cost ultrasonic sensors that offer a more focussed beam. However, it has been found that ultrasonic sensors that offer more focussed beams, such as the SRF235, offering a sonar beam of just 15° [101], are likely to miss any sonar echoes that arise out of beams being incident at angels greater than 10° from the normal. Since echo-location is the means of operation of these sensors, missing an echo implies that obstacles could go completely undetected. For example, at just 10° from the normal, the echo is not sensed by the transducer, as illustrated by Figure 4.2 (a). Also, these sensors are very sensitive to *ghost echoes* from unintended objects. As can be seen in Figure 4.2 (b), the reflected beam or echo is initially missed and therefore the flat surface goes undetected by the SRF235. Additionally, this echo gets reflected back onto the flat surface off the surface of an object behind the sensor, which is in turn is reflected back to the transducer, wrongly giving the transducer the impression that the distance to the closest obstacle is over twice the actual distance to the flat surface. Also, the cost of using these sensors to get a 360° coverage becomes prohibitively expensive.

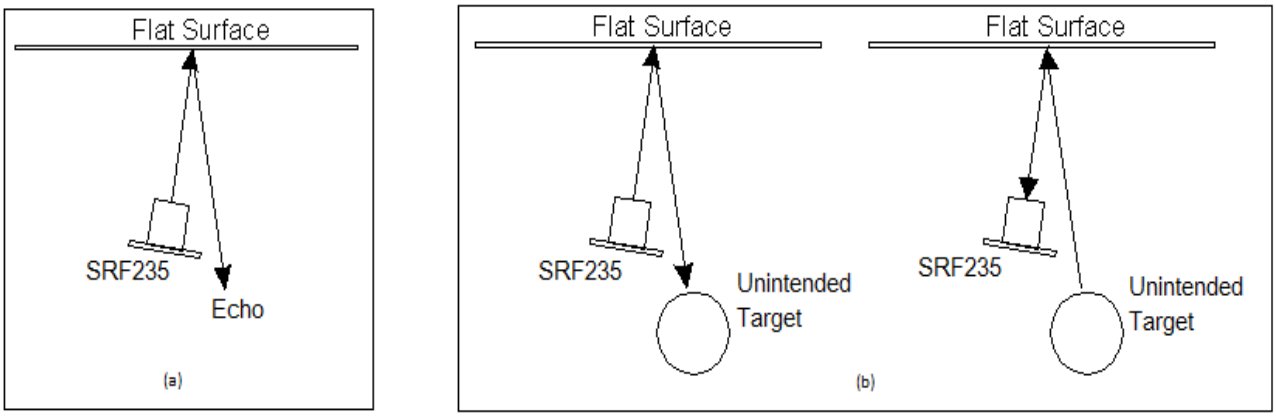


Figure 4.2 Effect of the narrow beam pattern in the SRF235 [101]

4.1.2 Analysis of existing methods to improve angular resolution of the SRF05 sensor

To overcome the problem of low angular resolution inherent to low cost ultrasonic sensors, it is necessary to develop a way to improve the perceived resolution. This section reviews some existing methods of dealing with low angular resolution.

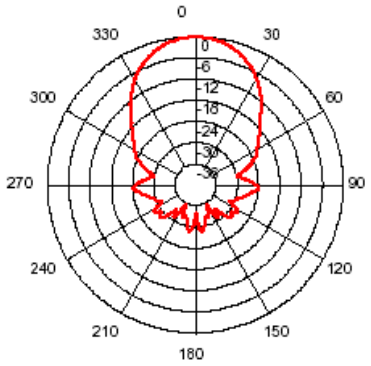


Figure 4.3 Beam pattern of SRF05 [89]

It is suggested in [40] that reducing receiver gain can improve angular resolution by a factor of 2 in some cases. From Figure 4.3, it is evident that the power of the beam drops from 0dB to -6dB between 330° and 30°. By reducing the receiver gain, the sensitivity of the receiver to pick up smaller signals is lost, thereby perceiving a more focussed section of the echo of the ultrasound. However, the chosen ultrasonic sensor SRF05 does not give the user control over the receiver circuitry and hence the receiver gain cannot be modified.

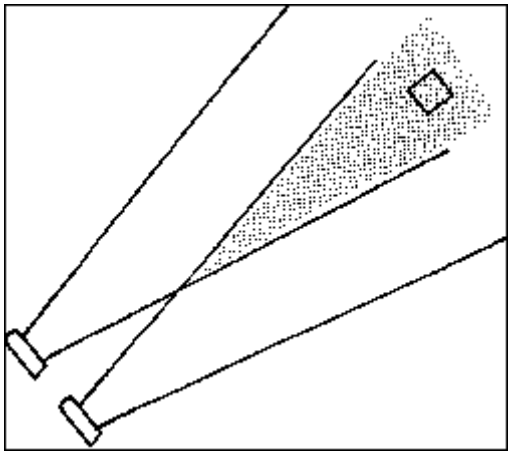


Figure 4.4 Beam splitting [40]

Techniques of beam splitting where sensor beams are overlapped [40] are said to improve resolution. In Figure 4.4, the 2 sensors have an overlapped region, dividing the sensed area into three subregions. If both sensors detect the obstacle, it is evident that it is in the overlapped region. Otherwise it can be observed that the obstacle lies in the region exclusive to the sensor that detects it.

The sensor arc is the portion of the beam emanating from the sensor reduced to 2 dimensions. In the specific case of the SRF05, while it can be expected from Figure 4.3 that this sensor's beam is a uniform cone in 330° to 30° range, it is actually a teardrop pattern tapering gradually to a point as the sensor range approaches 4m as shown in [88]. However, it can be safely approximated to a cone and therefore an arc in 2 dimensions, when the obstacle is less than 1.5m away. Even at this distance, a sensor arc subtending an angle of 55° has a length of 1.4m which gives rise to a lot of uncertainty regarding the position of the obstacle along the arc. One way of decreasing this uncertainty is to simply move closer to the obstacle, thereby forming a smaller sonar arc on the surface of the obstacle, since $s = r\theta$ where s is the length of the sonar arc and r is the distance to the obstacle and θ is the sensor resolution. Chapter 5 on simulation describes specialised algorithmic changes to address such situations in special cases.

The centreline model, described in section 2.4, is an intuitive and popular model which assumes that the obstacle is at the centre of the sonar arc. The biggest shortcoming of this method is the missing of gaps between obstacles due to a wrong estimation of the obstacles' position.

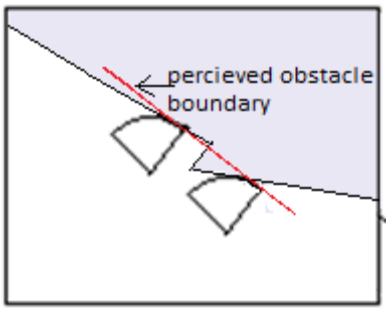


Figure 4.5 A situation in which McKerrow's method can give erroneous results

McKerrow's common tangent method [41] leverages the motion of the robot in order to locate where the plane face of an obstacle might be and is also discussed in section 2.4. (Kindly refer to section 2.4 as the details of this method are not repeated here) The disadvantage of the McKerrow method is that when the face of the obstacle is not strictly

planar, especially when the robot is approaching it, determining the common tangent's position can be erroneous as illustrated in Figure 4.5. Determining corners of polygonal obstacles can also be challenging by the McKerrow's common tangent method.

It is always true that sensed obstacles can lie anywhere along the sonar arc. When arcs intersect, intuitively the point of reflection of the sonar arc is likely to lie on the intersection. The arc transversal mean or ATM method [42] leverages on the robot's motion to find intersections between sonar arcs as the robot goes past an obstacle and is described in section 2.4. This method is best suited for non-specular walls (that do not cause mirror like reflections) and to detect corners. If the surface of the obstacle is specular and flat, the ATM approach gets reduced to the centreline approach due to the dearth of arcs that intersect transversally. It is also computationally intensive, as a high degree of sampling is required to detect continuous points on the obstacle face. These methods also do not give an estimate of how the face of the obstacle might change.

4.1.3 Examining the applicability of the existing techniques to improve angular resolution in the context of the Hybrid Bug algorithm

With a review of some of the popular methods of improving azimuth resolution, the applicability of some of these methods in the context of the Hybrid Bug algorithm is now examined as part of the investigation in this research. The Hybrid Bug algorithm has 3 basic behaviours:

- Motion toward the target behaviour
- M-line coverage behaviour and
- Boundary following behaviour

In the motion toward the target mode, the algorithm needs to build the local tangent graph (LTG), with the end points of the sensed obstacles. These end points can be thought of as a loss of continuity in the sensor readings, in the ideal case where an extremely high angular resolution sensor was available. With lower resolution, erring on the side of caution and using the extremity of the sensor arc as the location of the endpoint may compromise the path length but will ensure that collision with the obstacle is avoided. Beam splitting techniques can be used to get a better estimation of the end point. This is clearly an easier manoeuvre to make with low resolution sensors than those required for algorithms such as the VisBug algorithm, which expect to know the exact location of the obstacle intersection with the m-

line and hope to follow obstacle boundaries while stationary in order to look ahead at the path that the Bug2 or Quasi Bug2 algorithm would have taken. This is a practical advantage the Hybrid Bug and TangentBug algorithms have over Bug2 and VisBug algorithms.

During the m-line coverage operation, the range information is sufficient as the direction of motion is fixed. Additionally, as the robot approaches an obstacle, with a reduction in the distance to the obstacle, the length of the sonar arc reduces. Therefore, some interference from jagged edges in the extremities of sonar arcs noticed from a distance could get eliminated as the robot moves closer to the obstacle until it approaches the threshold distance.

The boundary following behaviour poses the biggest challenge to sensors with poor angular resolution. Since obstacles are assumed to be polygonal, it is sufficient to estimate the followed face. The McKerrow's method could be used to detect the orientation of a face– however, there are no guarantees that the faces of the obstacle are strictly planar. The ATM method could be used to detect the boundary, but involves intensive sampling and computation. Additionally, it is unknown if the obstacle faces are specular. Neither of these algorithms leverage the use of multiple sensors. Therefore it would be advantageous to develop a new algorithm to deal with low azimuth resolution of the chosen ultrasonic sensor.

4.2 Boustrophedon area coverage

Section 2.3.1 reviews some of the research that has been carried out on area coverage algorithms. One of the discussed algorithms, the boustrophedon area coverage algorithm, is utilised in this research. This section analyses some of the area coverage options available and justifies the choice of the boustrophedon pattern for area coverage in this research.

A depth first approach to area coverage, such as the spanning tree coverage (STC) algorithm reviewed in section 2.3.1, veers away from the traditional farming patterns of covering agricultural land, known as the boustrophedon (meaning 'way of the ox' [38]) method. A depth first pattern of area coverage would result in a coverage pattern that starts along the periphery and terminates at the centre in an obstacle free environment, as shown in Figure 4.6 (b). Apart from being the coverage pattern of choice in traditional farming practices, the boustrophedon coverage pattern has a small advantage over a depth first coverage pattern. In the scenario where the robot covers area from left to right, if an obstacle is encountered along a track and its position is remembered, the robot can use a small hack if it encounters an obstacle in the vicinity of the memorised position on the next track. Instead of

circumventing the obstacle from the left, it can do so from the right which will work to its advantage when obstacles are relatively small (typically less than three track widths - Appendix II). This is illustrated in Figure 4.6 (a). In this manner, obstacle circumvention is optimised by simply tracking obstacles along the previous track. However, if a depth first approach were used, in order to attain a similar benefit, all the obstacles encountered along the previous four tracks would need to be memorised to make an informed decision on the direction of circumvention. For example, in Figure 4.6 (b), if it is decided that the robot should circumvent the obstacle from the left (indicated in orange) with an aim to cover greater area, the following fact must be taken into consideration: even though the track to the left is clear of obstacles, the one above has obstacles in the vicinity and therefore increases the path length without covering the intended area. The path in maroon, which is the path taken when circumventing the obstacle from the right, would prove optimal in this case.

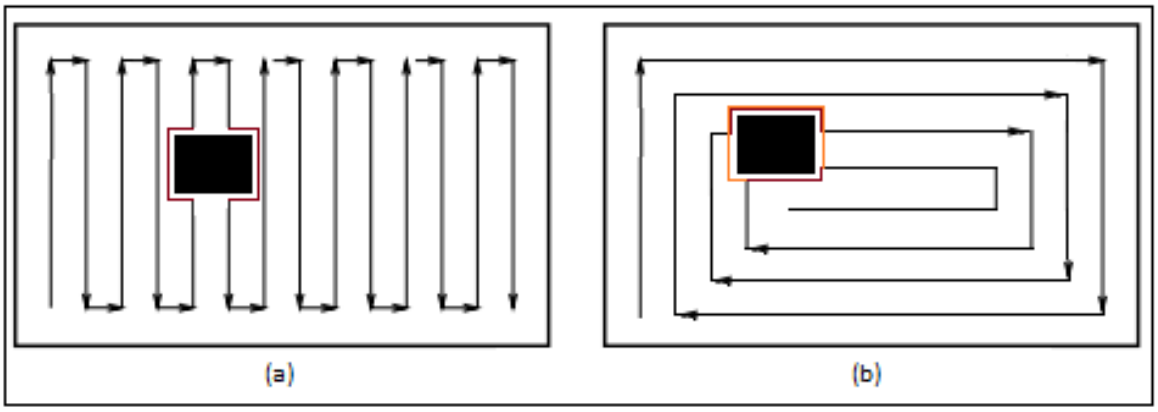


Figure 4.6 Optimised obstacle circumvention in a boustrophedon scenario versus a depth first scenario

As described in Chapter 3, this research has focussed on the development of an obstacle avoidance algorithm that works to efficiently avoid obstacles along a straight line path (the m-line). The algorithm works most effectively when obstacles are smaller, typically less than three tracks wide (Appendix II discusses that in the target scenario of an agricultural application, the obstacles are typically smaller). This is because with an increase in the number of tracks that an obstacle intersects, the number of times the periphery needs to be followed in order to circumvent the obstacle will be equal to the number of tracks circumvented, which will lead to larger path lengths. This is illustrated in Figure 4.7, where the obstacle is circumvented six times as it incepts six boustrophedon tracks. In such cases, algorithms based on approximate cellular decomposition [20], such as the BA*[34] algorithms may yield more optimal path lengths.

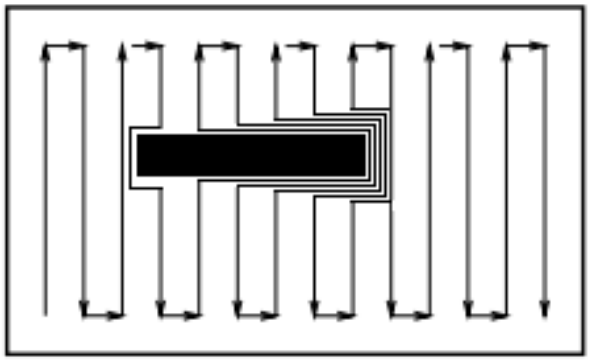


Figure 4.7 Large Obstacle in the course of a boustrophedon coverage

However, when obstacles are relatively small and comparable in dimension to the track width, a simple boustrophedon based area coverage with obstacle circumvention has one significant advantage over algorithms such as the BA*[34]. In agricultural settings where the area to be covered is large and obstacles are sparse and small, algorithms based on approximate cellular decomposition have to backtrack to locations where tracks were left partially covered or uncovered owing to the presence of an obstacle. This is illustrated in Figure 4.8 (b), where the distance to be backtracked is shown is maroon. This distance is substantially larger than the excess produced by obstacle circumvention of the small obstacle, shown in Figure 4.8 (a), which incidentally covers more of the intended area as well. The scale of the areas to be covered is kept consistent with illustrations in Figure 4.7. However, if the area to be covered is larger, which is to be expected, the backtracking distance also increases.

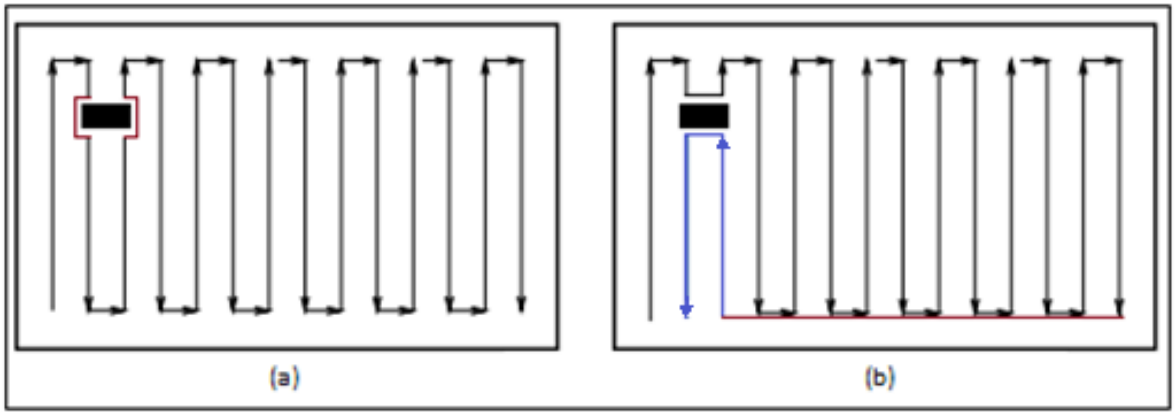


Figure 4.8 Effect of algorithms based on backtracking when obstacles are small in comparison to the navigable area

Coverage algorithms based on approximate cellular decomposition also tend to avoid cells that are only partially covered by obstacles. The granularity of the cells is configured before the process of area coverage and is typically based on the robot's footprint [20]. However, it

is sometimes based on the ranges or dimensions of on-board tools, effectors or sensors that are mainly responsible for covering the area, as in the case of the STC algorithm [30]. In such cases, it is possible that even if it is possible for a robot to follow a boustrophedon track through a cell, the path is not taken as the cell is partially occupied by an obstacle. Finally, a simple boustrophedon approach to area coverage that circumvents obstacles as they occur lends itself well to be used by cooperative robots, due to the ability to concisely communicate details of the area covered by a robot by simply stating the start position and current position of the robot. There will be no ambiguity regarding the path it has taken when obstacles are encountered, which may be the case with coverage algorithms based on approximate cellular decomposition. In these algorithms, entire grid maps of the area to be covered with cells that have been covered by each robot will have to be exchanged by robots working on shared workspace in order to optimise area coverage.

4.3 Development of a new method for the detection of obstacle boundaries using low resolution sensors

In the scope of this research, the obstacles are typically polygonal or approximated as polygons. The azimuth resolution of the ultrasonic sensors used in this research (SRF05) is low with a beam angle of 55°. In the context of the Hybrid Bug algorithm, the boundary following behaviour relies heavily on understanding where the boundary of an obstacle lies and also its orientation. In the ideal case where navigation sensors have extremely good angular resolution, it would be easy to follow the boundary by simply determining the normal to a surface (using comprehensive range information for every angular increment) at every quantum move of the robot and moving orthogonally to it [56]. However doing so, by assuming that the closest point of the obstacle is in the centre of the sonar arc of 55° can often be grossly erroneous, especially at larger distances from the obstacle. McKerrow's method [41] of determining an obstacles planar surface is very useful. However it is better suited to mapping applications where obstacle boundaries can be retrospectively constructed since it lacks any form of prediction as to how the orientation of the currently pursued face changes.

To overcome such issues, a new method that leverages the use of multiple sensors has been developed in this research. This method aims to improve the low azimuth resolution of the sensor and also provides insight into how the orientation of the face of the obstacle may change has been developed. In this method, it is assumed that 10 SRF05 ultrasonic sensors have been placed at every 36° around a sensor ring to obtain 360° coverage. The sonar cones

overlap each other by 19°, allowing for some resolution improvement using the beam splitting technique.

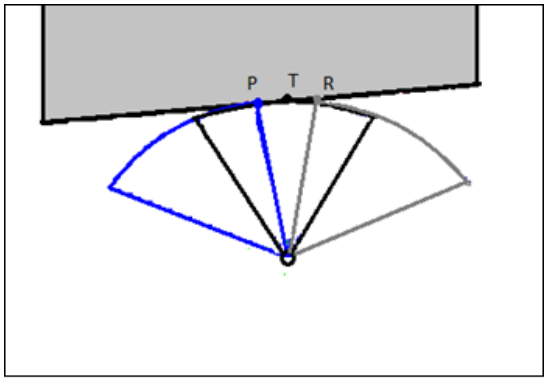


Figure 4.9 Sensor arcs of 3 adjacent sensors in different colours

Figure 4.9 shows the overlap between adjacent sensors. The sonar cone of each sensor is represented in a different colour. The two dimensional approximation of the cones is used. When an obstacle is encountered, the sensor facing the obstacle, which has the lowest range reading is considered along with its neighbouring sensors. In the absence of corners and interference from other obstacles, the following observations can be made:

- The boundary of the obstacle is tangential to the sonar arc of the sensor with the minimum reading
- The adjacent sensors have their minimum readings emerging from the extremities closest to the sensor with the lowest reading.

It is noted that given the arc of a circle and an external point that a tangent to the arc has to pass through, the slope and hence the equation of the tangent(s) can then be determined. Depending on the position of the external point and the size of the arc, one or both of the tangents that pass through it will lie on the arc.

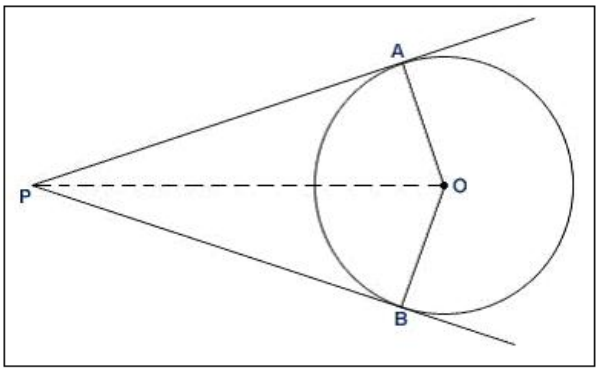


Figure 4.10 Tangents to a circle from an external point P

For example, in Figure 4.10, the coordinates of the point *P*, denoted by (P_x, P_y) , the coordinates of the centre of the circle, (O_x, O_y) and the radius of the circle *r* are all known. It is assumed that the tangent to the circle has a slope *k*, the equation of the tangent passing through *P* would be

$y - P_y = k(x - P_x)$ or

$$-kx + y + kP_x - P_y = 0 \qquad (4.1)$$

which is of the form $Ax + By + C = 0$

Since the radius of the circle is perpendicular to the tangent, the perpendicular distance of the centre of the circle from the tangent must be equal to the radius. Using the perpendicular distance formula,

$$\text{distance}((x_0, y_0),\ Ax + By + C = 0) = \frac{\left|Ax_0 + By_0 + C\right|}{\sqrt{A^2 + B^2}}$$

Therefore,

$$\text{distance}((O_x,O_y),\ -kx + y + kP_x - P_y = 0) = \frac{\left|-kO_x + O_y + kP_x - P_y\right|}{\sqrt{k^2 + 1}} = r$$

or $$(-kO_x + O_y + kP_x - P_y)^2 = r^2(k^2 + 1) \qquad (4.2)$$

Solving for *k* gives the two possible slopes for the two tangents which when substituted in (4.1) will give the two equations of the lines. Subsequently, the points of contact of the tangents with the circle can be determined, which could aid the process of filtering out the tangent that does not touch the arc. Therefore, with reference to Figure 4.9, the central arc can be considered as part of a circle whose centre *O* is the current robot position and radius is the minimum sensor reading, the shortest distance to the obstacle. The boundary of the obstacle is tangential to the central arc and *P* and *R* are the external points also lying on this line. Using (4.2) it is possible to find tangents to the central arc that pass through the external points *P* and *R*. The slope of the tangent gives the orientation of the boundary of the obstacle. In the case where both tangents through an external point are found to meet the circle within the span of the arc, only the tangent lying on the side of the line *OP* closer to the obstacle is considered. This tangent is termed as the *admissible tangent*. The points where the tangents

through *P* and *R* contact the central arc are computed independently. When they coincide, there is increased confidence in the fact that the orientation of the face of the obstacle is given by the slopes of the tangents. When these points do not coincide (with a certain degree of tolerance), it can be due to 2 reasons:

- A corner has been detected
- Another obstacle or another part of this same obstacle has interfered with sensor readings.

Algorithm 4 is a module within the integrated boustrophedon area coverage algorithm that incorporates the Hybrid Bug algorithm. It is used to determine the obstacle boundary continually while performing a boundary following operation to circumnavigate the obstacle. This algorithm also addresses situations where the points of contact of the tangents with the central arc do not coincide.

Algorithm 4: Boundary perception and following for sensors with low angular resolution

Step 1: Initialize m_{prev} = NAN (Not a number) where m_{prev} is a variable reserved to hold the previous value of the tangents' slope once they have been found to be equal.

Step 2: While in the boundary following mode, repeat

- From the readings of adjacent sensors, compute the extreme points on the sonar arcs closest to the sensor with the minimum reading (points *P* and *R* in Figure 4.9)
- Find an admissible tangent to the central sonar arc that passes through each of these points. Let the points of contact of the tangents through the external points *P* and *R* be T_P and T_R respectively and the slopes be m_P and m_R respectively.

If T_P=T_R and m_P=m_R (The tangents contact the central arc at the same point and have the same slope)

- The orientation of the face of the obstacle is said to be m_R and the robot travels along this direction. Set $m_{prev} = m_R$.

Else

- If $m_{prev} \neq$ NAN, $m_{prev} = m_P$ or $m_{prev} = m_R$ and the corresponding point of contact of the tangent with the central arc, T_P or T_R respectively, (extended to form a circle) lies within the span of the sonar arc

- Travel along m_{prev} . A possible corner has been detected.

Else (previous values of the slope do not exist or do not match either of the current slope of the tangents)

- Assume the shortest point is along the centreline of the sonar arc with the smallest reading and move orthogonally to the line connecting the robot to this point (Centreline model).
- Reinitialize m_{prev} = NAN.

This method enables the following of planar boundaries and in the context of the Hybrid Bug algorithm, it is sufficient to negotiate corners as well, since the value of the previous slope, m_{prev}, will keep the robot travelling along a straight line until such a time that the obstacle boundary stops being tangential – i.e. a corner is reached, at which point, the centreline model will help the robot change its orientation and navigate till either motion toward the target behaviour is initiated or another planar surface of the obstacle is visible to three sensors such that the robot can follow it. A flowchart incorporating this snippet is presented in Figure 4.11. Section 5.2.2.1 proposes a further improvement to decrease the turn radius taken by the robot while navigating corners based on collinearity of points detected on the new face.

4.4 An integrated approach to boustrophedon area coverage incorporating the Hybrid Bug algorithm with the enhanced obstacle boundary detection method to enable boundary following

While incorporating the Hybrid Bug algorithm into the boustrophedon coverage path in order to avoid obstacles seems fairly straightforward, there are some more minor modifications that need to be made to the Hybrid Bug algorithm to provide better overall results in optimising the path length. The Hybrid Bug algorithm attempts to make decisions about finding the most optimal path to the target via the m-line, based on the local information available to it. However, while performing boustrophedon sweep of one track, the robot gains more global information about the obstacles along that track. Since map building is neither desirable nor a feature of the new coverage pattern, a simpler flag like variable can be maintained to indicate that an obstacle lies on the current track, if the obstacle appears stationary (i.e. it has substantial dimensions). Optionally, a list of locations where the obstacles were encountered can also be recorded. This can then be used to make decisions about circumventing obstacles

on the next track – The Local Tangent Graph (LTG) nodes closer to the vicinity of encountered obstacles on the previous track can be removed thereby biasing the Hybrid Bug algorithm to use the other direction to circumvent obstacles. Since obstacle widths are expected to be typically less than 3 track widths, this modification results in only having to circumvent less than two thirds of the obstacle width in the second encounter and less than a third in the third encounter. The simulations in Chapter 5 incorporate this enhancement. During the simulations, the algorithm was further optimised by adding a clause to change the direction in which the robot is following the boundary and set a flag, if the robot is following the border of the coverage area, that is not part of the planned boustrophedon path, for more than a threshold distance. This allows the algorithm to avoid having to skirt the border of the entire coverage area. If the robot happens to follow a second boundary up to a threshold distance while attempting to circumvent a single perceived obstacle, it is assumed that the target is unreachable. In such a case, the target is simply shifted to the next available boustrophedon track.

With these enhancements incorporated, an integrated approach to boustrophedon area coverage using on the Hybrid Bug algorithm is presented as Algorithm 5. Some new variables are defined in the algorithm. The *current_track_number* records the index of the current track so that the positions of obstacles along the current track can be recorded for use on the next track and the start and target points can be assigned in accordance with the pre-planned boustrophedon path. The *direction_change* flag is set when the robot starts following the boundary of the area to be covered due to the proximity of the obstacle to the boundary. Once this flag is set, the robot attempts obstacle circumvention from the other direction. If it follows an area boundary again, the algorithm concludes that the target is unreachable and moves to the next track, if one is planned. The *action* enumeration indicates if the robot is covering a track or transitioning between tracks. The hit point of every track is initialised to the start point of the track in order to prevent m-line tracing from continuing indefinitely.

Algorithm 5 - Boustrophedon area coverage using the Hybrid Bug algorithm

Note that the input to this algorithm are: the boundaries of the area to be covered, track width w and threshold nearest distance to obstacle t

Step 1: Plan the boustrophedon path by assessing the boundaries of the area to be covered and dividing them into parallel tracks separated by the configured track width *w*.

Step 2: Travel to the beginning of the first track. Initialise the variables involved. Assign *S* = start point of the first track and *T*= end point of the first track. Set current_track_number = 1; *H*=*S*; Set the *direction_change* flag = 0 and set *action* = track coverage.

Step 3: Reset the *direction_change* flag. Construct a Local Tangent Graph (LTG), as suggested in section 3.3. If *action* = track coverage and obstacles were encountered within a vertical range of 3×*w* along the previous track, remove the LTG nodes towards the direction of the previous track and edges connecting them. Move towards the target *T*, via the m-line when possible or along the locally optimal direction on the current LTG's subgraph, until one of the following events occurs:

- The target is reached. Goto Step 8.
- The robot reaches the m-line at a point T_{temp} from the left or right semi-plane. Go to Step 4.
- A local minimum is detected: Define a hit point *H* as the point on the junction of the m-line and the obstacle boundary.
    - If a threshold value *t* is defined that is less than the sensor range *r* and the robot is on the m-line, proceed until the obstacle is at a distance *t* from the robot.
    - Go to Step 5.

Step 4: Suspend motion toward the target behaviour and survey the direction of the m-line towards the start *S* for uncovered segments of the m-line. Travel down the m-line till the previous obstacle or the last hit point *H* is encountered and return to T_{temp}. The robot ensures keeping a distance *t* to the previous obstacle. If the previous obstacle is encountered at a leave point *L*, the pair consisting of the hit point *H* and the leave point *L* is added to a list of obstacles in order to maintain a record of obstacles along the current track. Reset T_{temp} to *T*. Go to Step 3.

Step 5: Choose a boundary following direction. Move along the boundary using Algorithm 4, while recording d_{min} (T), until one of the following events occurs:

- The target is reached. Go to Step 8.
- The leaving condition holds: $\exists V_{leave} \in$ LTG s.t. $d(V_{leave}, T) < d_{min}(T)$. Go to Step 7.
- The robot completed a loop around the obstacle. The target is unreachable. Go to Step 8.
- The robot starts following the boundary of the area to be covered, due to the proximity of the obstacle to the boundary. This is not part of the planned path. Go to Step 6.

Step 6: If the *direction_change* flag is set,

- The target is unreachable. Reset the *direction_change* flag. Go to Step 8.

Else

- Switch the boundary following direction and set the *direction_change* flag. Go to Step 5.

Step 7: Perform the transition phase. Move directly toward V_{leave} until reaching a point Z that satisfies $d(Z, T) < d_{min}(T)$. Go to step 3.

Step 8: If the robot's action = track coverage,

- If more tracks exist,
    - Calculate the start point of the next boustrophedon track. Assign S=current robot position and T = start point of the next track; Set action = switching tracks; Repeat algorithm from Step 3.

Else

- Indicate successful area coverage and exit algorithm.

Else (the robot has been switching tracks)

- Set the list of obstacles to the previous track to the list recorded on the current track; Set current_track_number = current_track_number +1; Clear the list of obstacles on the current track; Assign S=start point of current

track and *T* = end point of the current track; Set *H=S;* Set action = track coverage; Repeat algorithm from Step 3.

The flowcharts in Figure 4.11, Figure 4.12 and Figure 4.13 summarise the working of Algorithm 5.

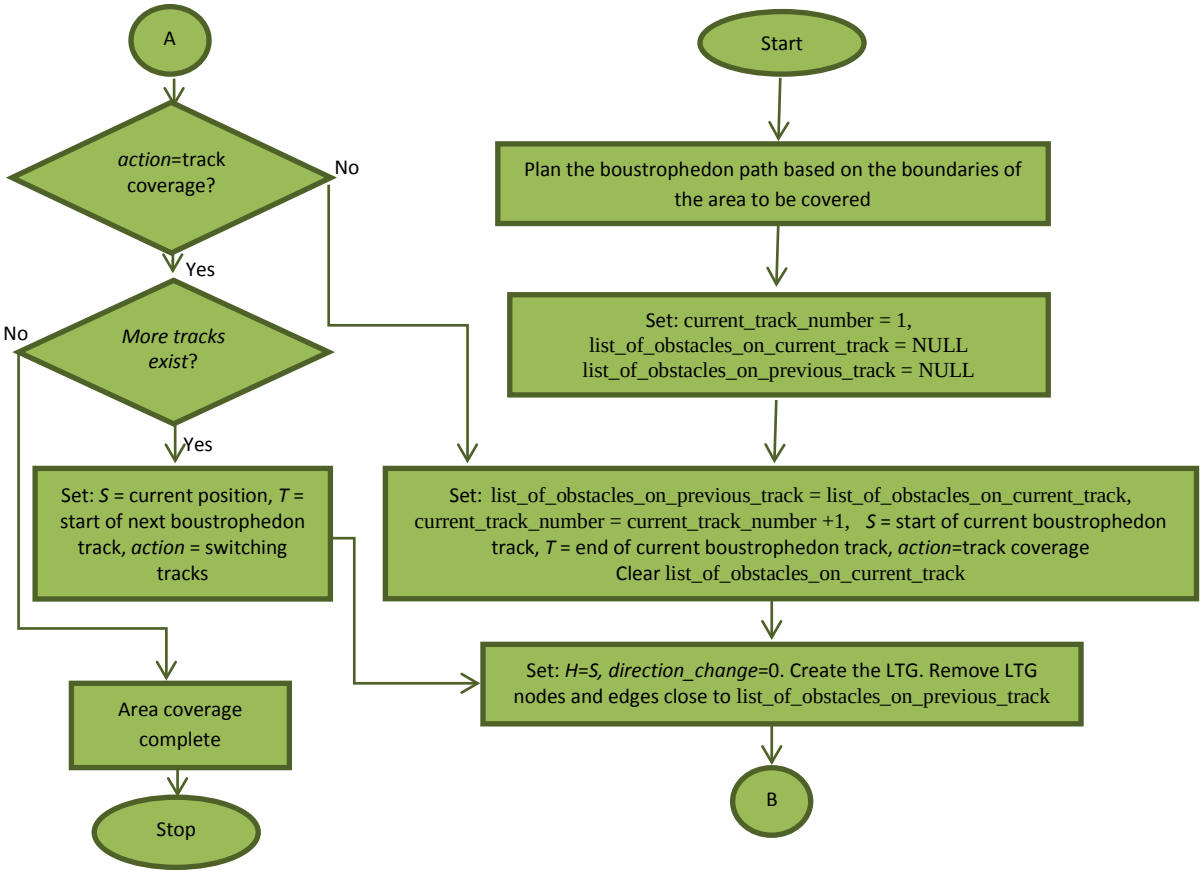


Figure 4.11 Flowchart representation of the boustrophedon area coverage algorithm

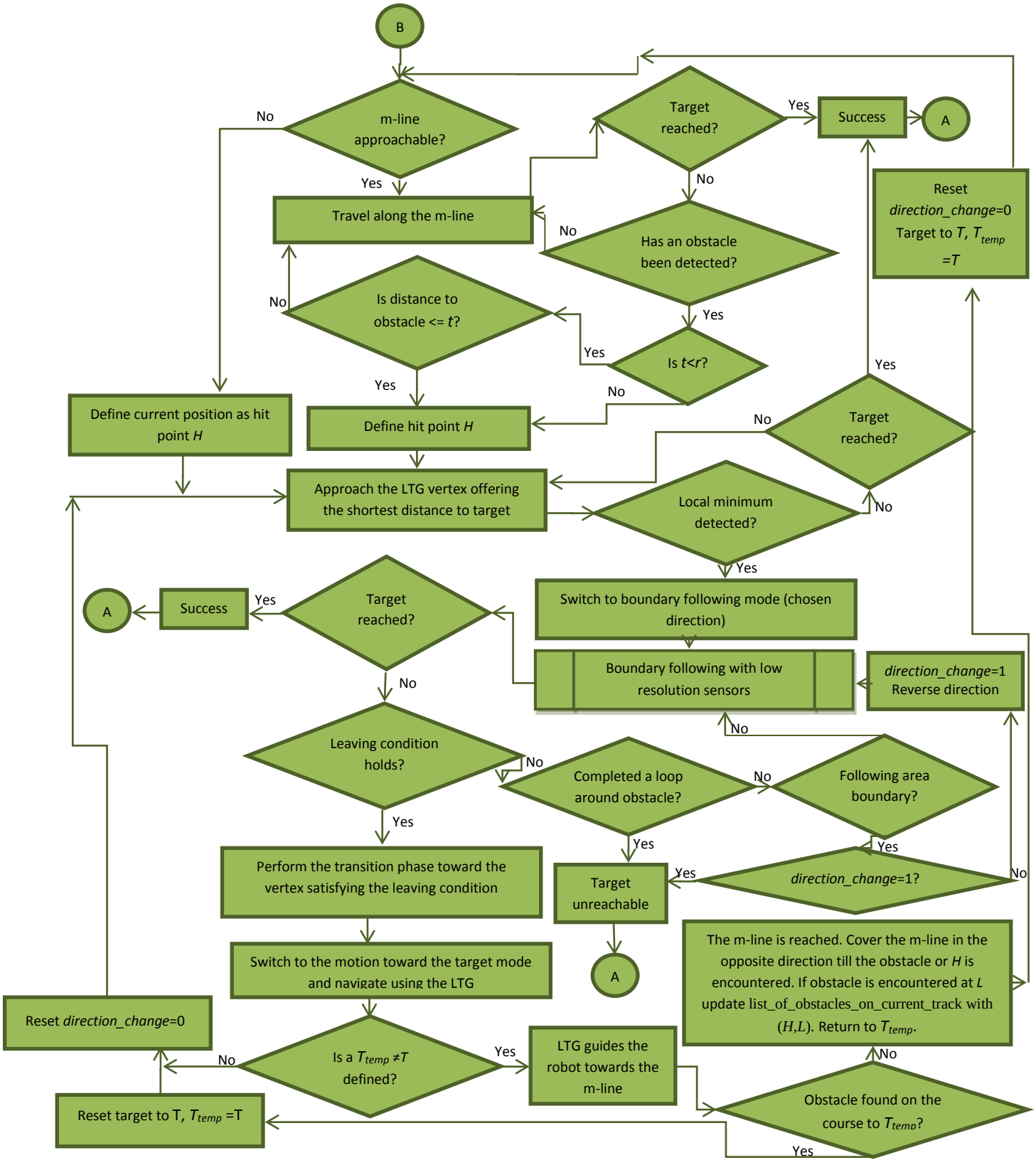


Figure 4.12 Flowchart representation of the boustrophedon area coverage algorithm (continued)

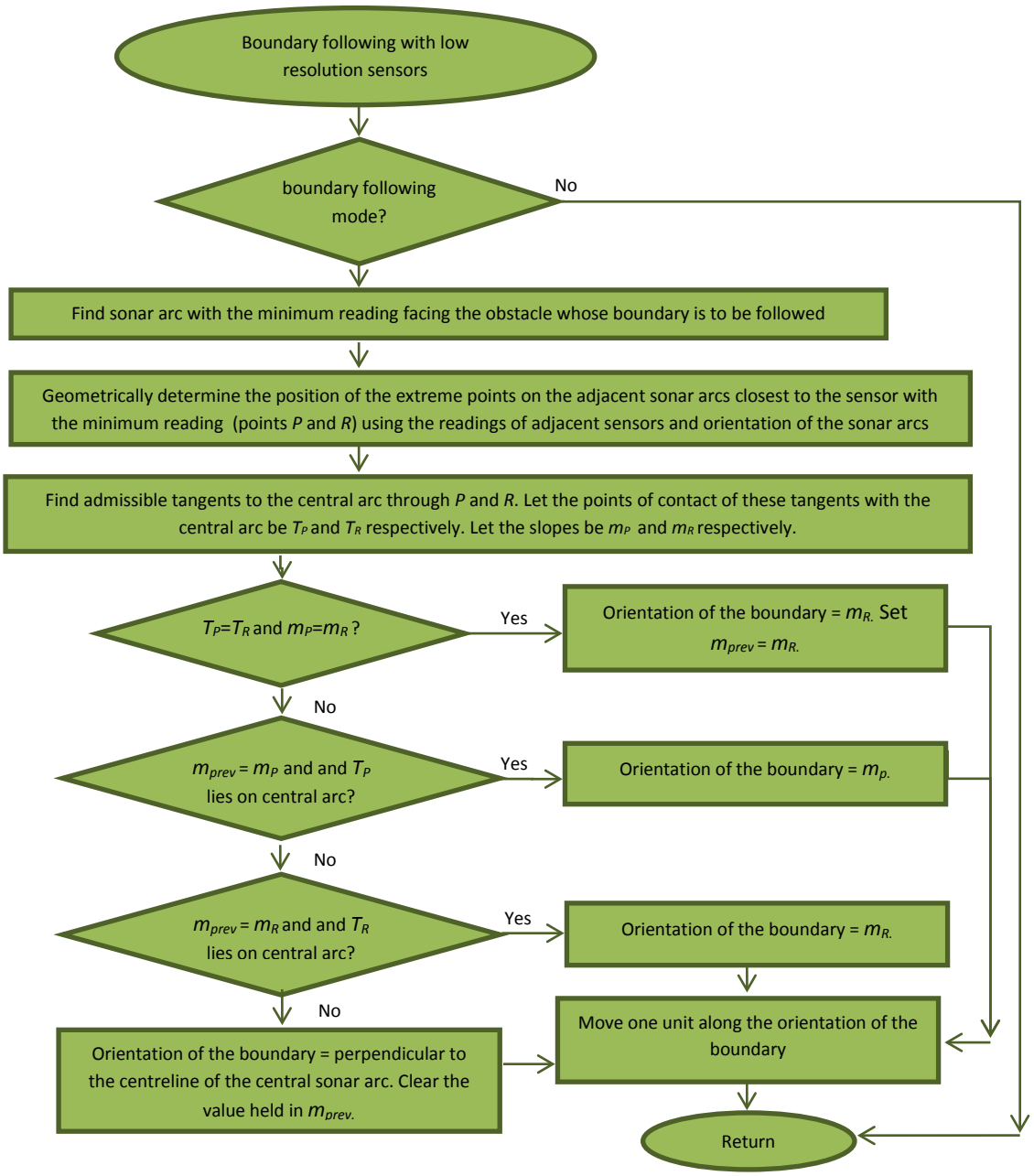


Figure 4.13 Flowchart representation of the boundary following module

4.5 Conclusion

It was seen in this chapter that simple boustrophedon motion based area coverage would be a highly suitable coverage mechanism for agricultural areas where the navigable area is large and obstacles are simple polygonal and typically small structures. With effective obstacle avoidance in place, a reduction in path length by way of eliminating the need for backtracking is a realistic expectation for online complete area coverage algorithms.

Simulation based studies comparing the performance of algorithms requiring backtracking with a Boustrophedon – Hybrid Bug approach are presented in the next chapter.

Additionally, while the Hybrid Bug algorithm described in Chapter 3 is for the situation where the sensors have ideal angular resolution, this chapter proposed techniques of making the algorithm work for real world low cost ultrasonic sensors considered in this research. The boundary following behaviour which relied heavily on the angular resolution of the sensors was enabled using a new algorithm to allow boundary perception using low cost ultrasonic sensors.

5 Simulation

5.1 Introduction

In this chapter, the efficiency of the Hybrid Bug algorithm and the integrated boustrophedon area coverage algorithm incorporating the Hybrid Bug algorithm are evaluated using a simulation study. This is achieved by creating an input data set of cases that included different areas to be covered and obstacles of various shapes and sizes distributed within them. These input scenarios were simulated with a point robot that aimed to maximise area coverage and minimise path length.

Simulation studies for evaluating the Hybrid Bug algorithm independently sought to prove its ability to consistently achieve shorter path lengths than the Bug2 algorithm and superior m-line coverage than the TangentBug algorithm. Simulation studies for evaluating the integrated boustrophedon area coverage algorithm incorporating the Hybrid Bug algorithm modelled sensors after real world ultrasonic sensors and verified the algorithms ability to circumnavigate obstacles. Simulation studies comparing the efficiency of the integrated boustrophedon area coverage algorithm with an existing online area coverage algorithm – the BA* algorithm were performed to verify the efficiency of the approach. Equipped with similar sensor models and given the same input scenarios, the performance of the two algorithms was evaluated based on their ability to optimise the parameters of path length and total area coverage. The termination of the algorithm for the input set of cases has also been verified.

5.1.1 Modelling for the simulation study

The obstacles that are likely to be encountered on agricultural farms such as trees and feeding stations are discussed in Appendix II. In keeping with the scope of research and the analysis of the Hybrid Bug algorithm, only static obstacles are simulated. A boustrophedon sweep size of 1m is chosen, as outlined in Appendix I. Consequently, it is assumed that the obstacles are typically less than 3 sweep sizes in width. Most of the simulations focus on obstacles of this size. Some extreme cases are however tested and presented herein. A database of potential environments was created using a matlab function that allowed the user to define obstacles by clicking to denote their vertices. Obstacles are represented as their nearest polygonal approximations. Since the robot is required to move in a two dimensional plane, the

environment and sensor beam patterns are represented along the XZ plane, from an aerial perspective.

The SRF05 ultrasonic sensors are modelled in the simulation using results presented in [88] as well as the technical specification of the SRF05 [89]. The ultrasonic sensor beam has a beam angle of approximately 55°[89]. Although these sensors provide accurate range information up to 4m, the detection zones presented in [88] along the XY, YZ and XZ planes shows that the sonar beam is actually more shaped like a teardrop than a cone, as usually assumed. One cross-section of the beam perceived by the experiments carried out in [88] is shown in Figure 5.1. It can be noted from these experiments that the beam can be approximated to a cone up to a distance of 1m and slightly beyond and begins to plateau between the 1m and 1.5m range. Beyond the 1.5m mark, the sonar beam starts tapering. In initial experiments, this teardrop pattern was modelled by not considering any obstacles detected outside the detection zone. However, this resulted in excessive and unnecessary use of the CPU cycles in the detection of obstacles falling out of the detection zone using ray casting techniques [110].

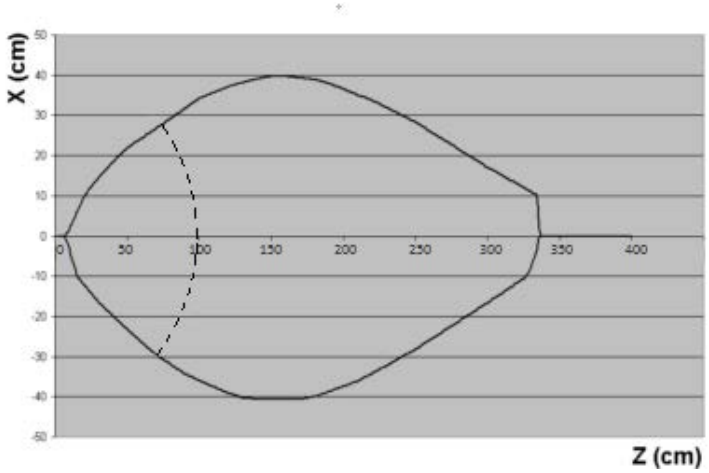


Figure 5.1 The detection zone along the XZ plane at y = -10 [88]

Since the sonar beam's detection zone is a reasonably uniform cone up to a range of 1m, all of the simulations that incorporate the SRF05 sensors in the Hybrid Bug algorithm and Boustrophedon motion use only range readings less than 1m. Any detection higher than 1m is ignored. Therefore, the sensor range is taken to be 1m for all the simulations performed.

Principles of ray casting techniques [110] are applied in order to allow the simulated sensors to find the closest points on the obstacle edges. This gives a closer representation of the real word and unknown environments while also allowing for multiple versions of algorithms to

be run on the same dataset. Intersections of the obstacle edges (which represent faces of the obstacle) and 'rays' or straight lines emanating from the robot position in all directions at a resolution of 1° are evaluated and minimum readings available in every 55° segment representing the sonar cone are calculated. This is expected to be the reading from the SRF05 sensor present in that position.

5.2 Results and discussions

The Hybrid Bug and integrated area coverage algorithms were implemented in MATLAB 7.11.0(R2010b). All simulations have been executed on a computer with an Intel®Core™ i3-3110M CPU clocked at 2.4 GHz, 4 GB RAM and Windows 8 64 bit operating system. Simulations were carried out in two steps. In the first step, the ideal case of the Hybrid Bug algorithm discussed in Chapter 3 was implemented. The angular resolution of the sensors were assumed to be ideal and simulated as 1°. In the second step, this algorithm was implemented as part of boustrophedon area coverage and the low resolution sensors were simulated (with an angular resolution of 55°). In both cases, a point mobile automaton was assumed as the robot dimensions can be subject to changes.

5.2.1 Simple Hybrid Bug algorithm

The simple Hybrid Bug algorithm described in Chapter 3 was implemented in 2D space along the XZ plane. Here, ray casting techniques [110] were used to provide obstacle information at a resolution of 1°, which is much greater than the angular resolution offered by the ultrasonic sensors. The implementation of the Hybrid Bug algorithm for the ideal case of high angular resolution was to compare its effectiveness with its predecessors which have contributed to the development of the algorithm – namely the Bug2 and TangentBug algorithms. The range resolutions for all the simulations were chosen to be 1m, consistent with other simulations done as part of this research. The sensor readings were obtained continually at intervals corresponding to every 2cm of the robot's movement.

5.2.1.1 Comparison with existing Bug Algorithms

The Hybrid Bug algorithm was designed to have maximal m-line coverage while allowing for optimal circumvention of the obstacle. In cases where obstacle shapes are fairly regular and there is sufficient spacing between obstacles, the performance of the Hybrid Bug algorithm closely mimics that of the Bug2 algorithm. However, when obstacle boundaries contain local minima or when multiple obstacles are clustered in a small space, the benefit of the Hybrid

Bug algorithm over the Bug2 algorithm becomes evident, as illustrated in Figure 5.2. Here, the Hybrid Bug uses concepts such as the local tangent graph to avoid local minima while navigating back to the m-line. In this case, the path length of the Bug2 algorithm was 544cm. and that of the Hybrid Bug was 464cm. Another case where the difference between the two algorithms becomes apparent is when the Hybrid Bug algorithm can use the local sensor range data to its advantage. For example, in Figure 5.3, the Bug2 which is programmed to circumvent the obstacle from the right side takes the longer route to the goal via the m-line. However, the Hybrid Bug uses the range sensing information and takes the locally optimal direction to circumvent the obstacle. In this case, the path taken by the Bug2 algorithm is 400cm long and that taken by the Hybrid Bug is 336cm long.

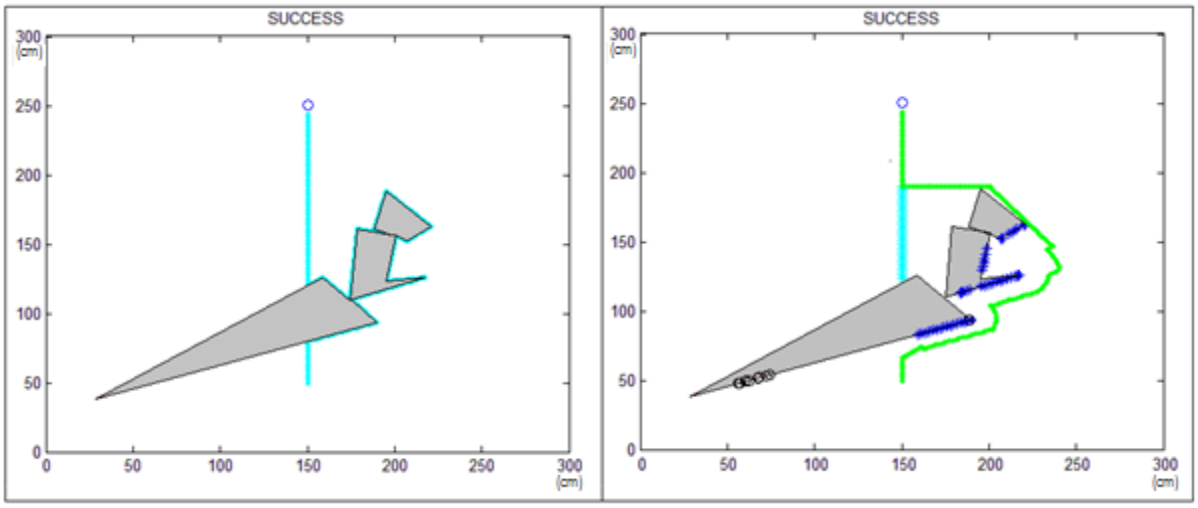


Figure 5.2 The Bug2 (Left) follows a longer path to the m-line than the Hybrid Bug (Right)

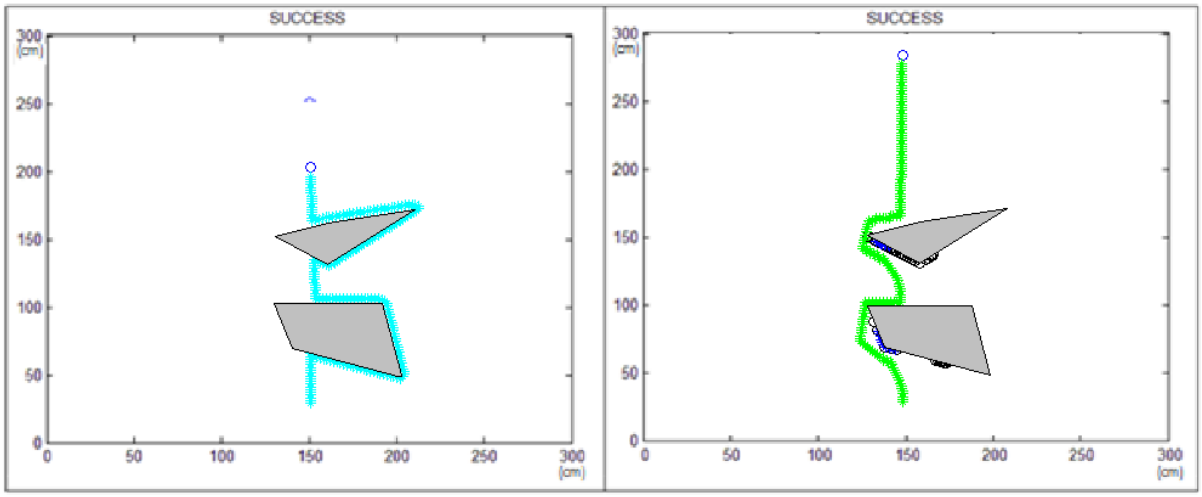


Figure 5.3 The Bug2 (Left) follows a longer path to the m-line than the Hybrid Bug (Right) due to a fixed direction of circumvention

The Hybrid Bug algorithm also achieves m-line coverage that is almost equal to that of the Bug2 algorithm and substantially higher than that of the TangentBug algorithm. Figure 5.4 illustrates the Hybrid Bug algorithm maximising the m-line coverage including the section between two obstacles along the m-line while this is not a priority for the TangentBug algorithm.

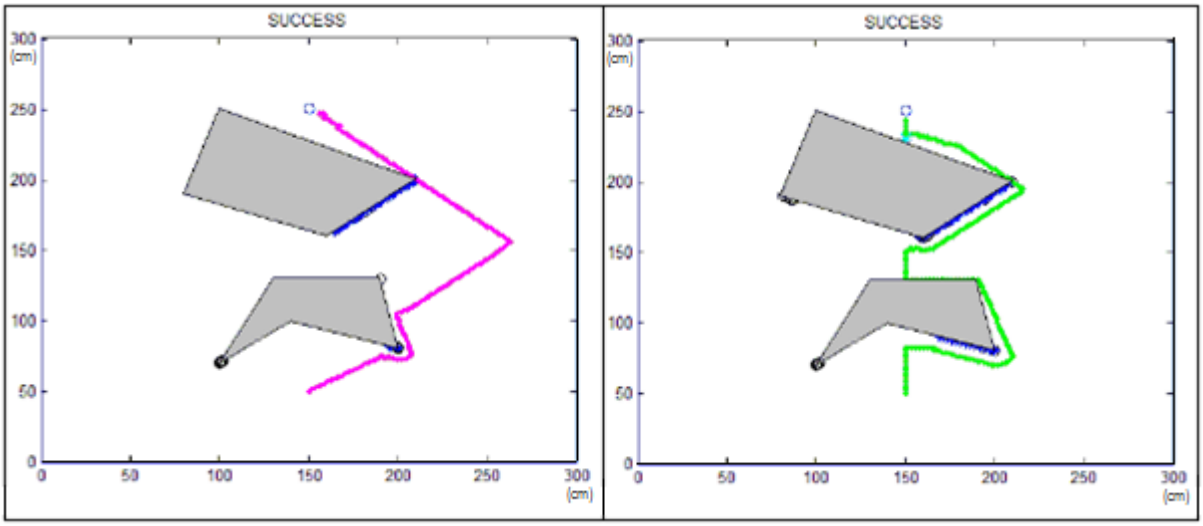


Figure 5.4 The TangentBug (Left) covers less of the m-line than the Hybrid Bug (Right)

Figure 5.5 shows a superimposition of the paths taken by the three algorithms. Although such a case (with extremely convoluted deep cavities within obstacles) is fairly rare, this figure is used to highlight the savings in path cost as well as the m-line coverage by the Hybrid Bug algorithm over the other two algorithms.

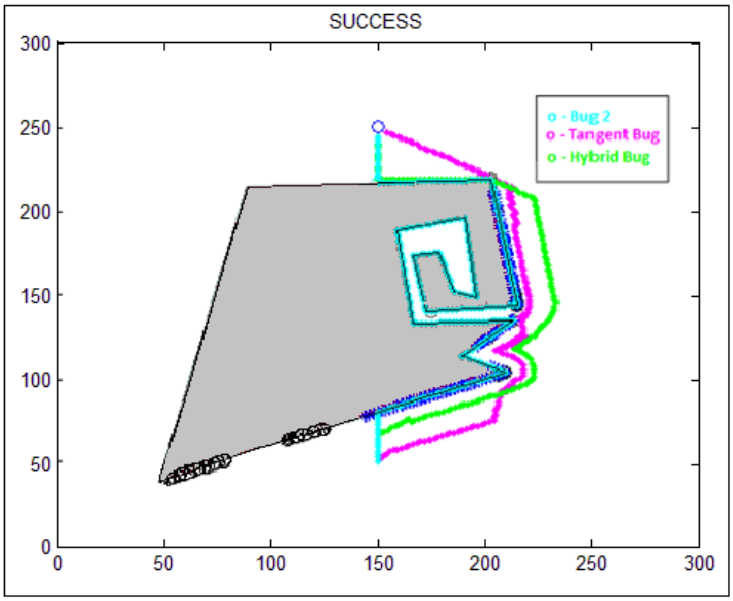


Figure 5.5 Illustration of the three bug algorithms on an obstacle with a maze like cavity

Table 5.1 records the average values of the results of over 40 simulations executed on the sample dataset comprising of obstacle courses over distances of 200cm and 600cm, with obstacles of different sizes, proximity and number lying along the course. The length of the obstacle course was chosen to accommodate obstacles conforming to the typical shapes illustrated in Figure 5.6, and having a maximum width of 3m (Appendix II). A maximum of 2 obstacles were placed along the m-line, as this was sufficient to evaluate the performance of the algorithm in covering sections of the m-line between obstacles. Reduced path lengths of the Hybrid Bug algorithm over the Bug2 algorithm were noted and the m-line coverage was seen to be drastically higher than that produced by the TangentBug algorithm. A subset of the tabulated results is provided in Appendix V.

Table 5.1 A table indicating average path lengths and m-line coverage values by different bug algorithms

Algorithm / **Parameter**	Bug2	TangentBug	Hybrid Bug	Distance between start and target points (cm)
Path Length (cm)	408	334	348	200
Path Length (cm) (Convoluted obstacles)	1200	770	900	600
m-line coverage(%) (Available Space)	100	7.93	91	N/A

5.2.2 Hybrid Bug Algorithm with simulated ultrasonic sensors and boustrophedon motion

The Hybrid Bug Algorithm was implemented with the SRF05 ultrasonic sensor stubs described in section 5.1.1. It was also adapted to fit into the Boustrophedon area coverage pattern by overriding the intrinsic behaviour of the Hybrid Bug algorithm which choses a direction in which to follow a boundary based on locally available sensor information with a more globally informed decision making approach: The robot is made to circumvent the obstacle along the right semi-plane if the boustrophedon track on the left sensed obstacles in the vicinity of this point. If no obstacles were sensed in the vicinity on the track on the left, it would be advantageous to use the left semi-plane. The resulting algorithm is called the boustrophedon area coverage using the Hybrid Bug algorithm (Algorithm 5). Obstacles in the input areas to the simulations subscribed to the assumptions outlined in Appendix II.

The below notation is used to represent various features detected in the simulations and used in the figures shown in this section:

Yellow Circle o: The point detected on the face of an obstacle (The point at which the obstacle edge is tangential to the arc of the sensor with the lowest admissible range reading).

Blue Star *: The external point detected by the sensor which is the right neighbour of the sensor with the lowest admissible range reading. The tangent passing through this point, tangential to the central arc could possibly represent the face of the obstacle.

Magenta Star *: The external point detected by the sensor which is the left neighbour of the sensor with the lowest admissible range reading. The tangent passing through this point, tangential to the central arc could possibly represent the face of the obstacle.

Black Star *: The path taken by the robot in the boundary following behaviour.

Green Star *: The path taken by the robot in the motion toward the target behaviour.

Cyan Star *: The path taken by the robot during the boundary following behaviour, when it is using the rear sensor (the sensor to the right or left of the sensor with the minimal reading when the robot is travelling towards the left or right respectively) and previously detected slope to follow the boundary. Depending on the situation this also denotes the uncovered path on the m-line, in the direction towards the start point that the robot covers on arrival to the m-line post boundary following (the m-line coverage behaviour).

Black Circle o: The points of discontinuity on the perceived obstacle boundary where the consecutive sensor readings change beyond a cut-off value.

5.2.2.1 Comparison with existing Coverage Algorithms

The BA* algorithm described in [34] was chosen as a candidate for comparison studies as it is one of the very few online boustrophedon area coverage algorithms available. The BA* algorithm was also simulated for the purpose of this research using the same sensor stubs as the Hybrid Bug algorithm. To simulate the BA* algorithm, the area to be covered had to be broken down into cells whose dimensions in turn would determine the track width. Therefore each cell was set to have a width of 1m and a height of 5cm, which is the distance by which the robot moved at every step. A cell was marked as unoccupied if the robot was able to approach its midpoint in one of the four directions – North, South, East or West in that order. i.e., if there were no obstacles in the line of sight. The dimensions of the area to be covered were chosen to be 700cm×700cm as this was sufficient to accommodate different obstacle densities and sizes (in keeping with the assumption that obstacles were less than 3 track

widths wide) while being small enough to execute multiple simulations. Sensor simulation involved obstacle detection using ray casting techniques [110] every time a reading was required. Since sensors were read at intervals corresponding to every 5cm of the robot's motion and obstacle detection using ray casting was time consuming, having extremely large areas to be covered would have very high execution times. Consequently, the number of simulations possible in this research would be reduced. However, this problem would not arise in real world experiments since sensor readings can be obtained relatively quickly.

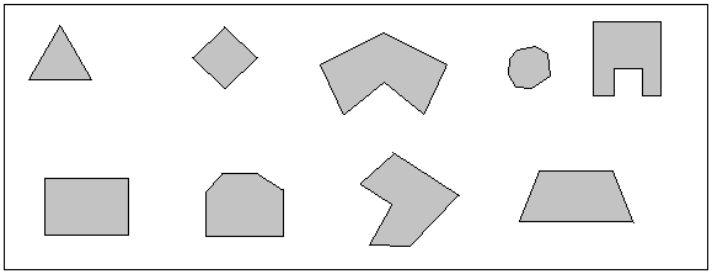


Figure 5.6: Typical obstacle shapes used in the simulations

Figure 5.6 gives an overview of the typical obstacle shapes used in the simulations (apart from a few exceptions where more complicated shapes were used in order to stress test the algorithm). These shapes were chosen to represent most scenarios likely to occur in the target applications, such as trees, feeding equipment, wheelbarrows etc. Curves were represented by their polygonal approximations. Different sizes, dimensions, densities and orientations of these basic shapes were considered. Over 60 simulations were carried out on over 30 different courses with assorted obstacle placements and densities. The results (averaged values in each category) are summarised in Table 5.3 and Table 5.3. Appendix V gives a subset of the tabulated results of the simulation study.

Table 5.2 Comparison of the path length generated by the Boustrophedon area coverage using the Hybrid Bug algorithm and the BA* algorithm

Obstacle Size	BA* (cm)	Boustrophedon area coverage using the Hybrid Bug algorithm (cm)	Improvement (%)
Small (< 1 track width)	7786.14	5855	33.29
Large (> 2 track widths)	12004.4	11125.3	5.59
Random (Average of complete data set)	7195.65	6188.33	18.26

During simulations, it was seen that the A* algorithm would attempt to cross over obstacles while finding the shortest path to a backtracking point, in situations where an obstacle lay only partially within a cell and was not detected in the line of sight detection method. This possibly had to do with the cells having an extremely large width in comparison to the height. The Hybrid Bug algorithm covers more of the m-line than the BA* algorithm and this is summarised in Table 5.3. As is evidenced, m-line coverage is independent of size.

Table 5.3 Comparison of the m-line coverage of the Boustrophedon area coverage using the Hybrid Bug algorithm and the BA* algorithm

Obstacle Size	BA* (cm)	Boustrophedon area coverage using the Hybrid Bug algorithm (cm)	Improvement (%)
Small (< 1 track width)	3916.25	4270	9.03
Large (> 2 track widths)	3299.4	3597.64	9.04
Random (Average of complete data set)	3711.4	3975.83	7.13

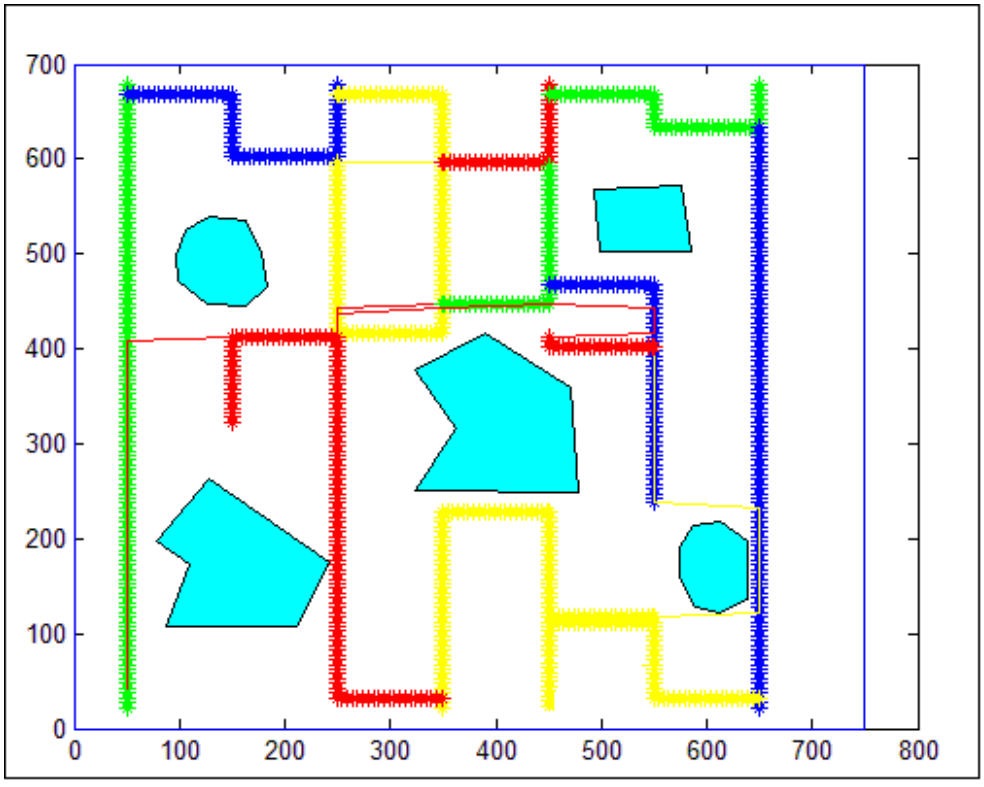


Figure 5.7 : Area coverage by the BA* algorithm

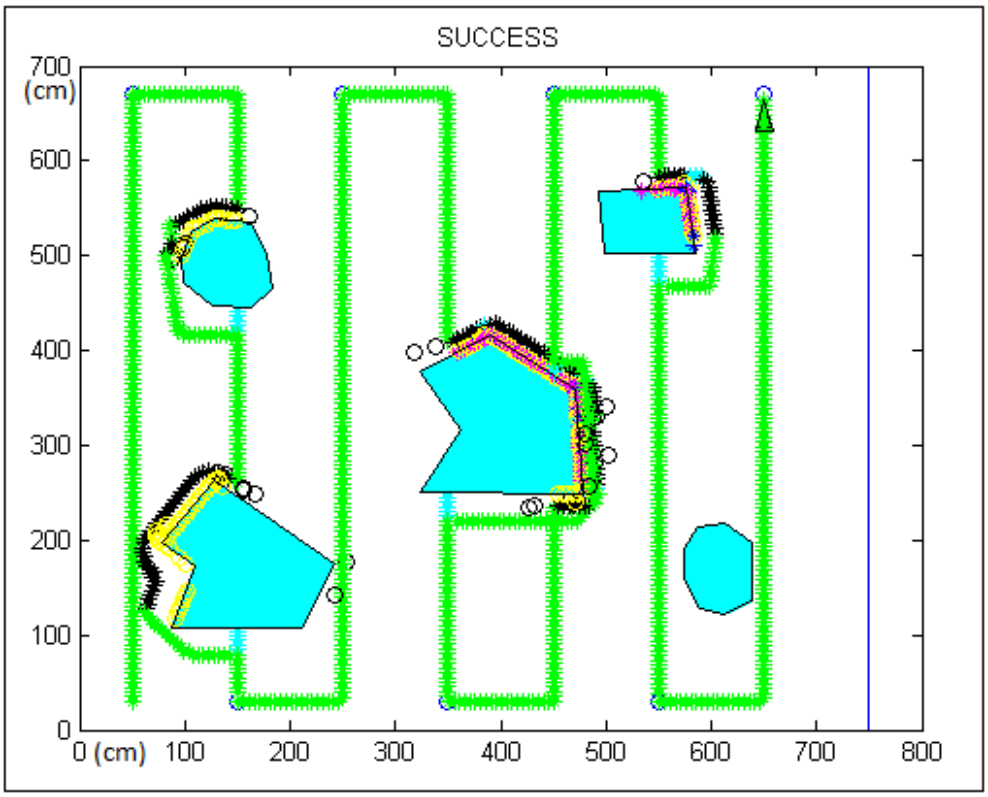


Figure 5.8 The same area as in Figure 5.7 covered by the boustrophedon area coverage using the Hybrid Bug algorithm algorithm

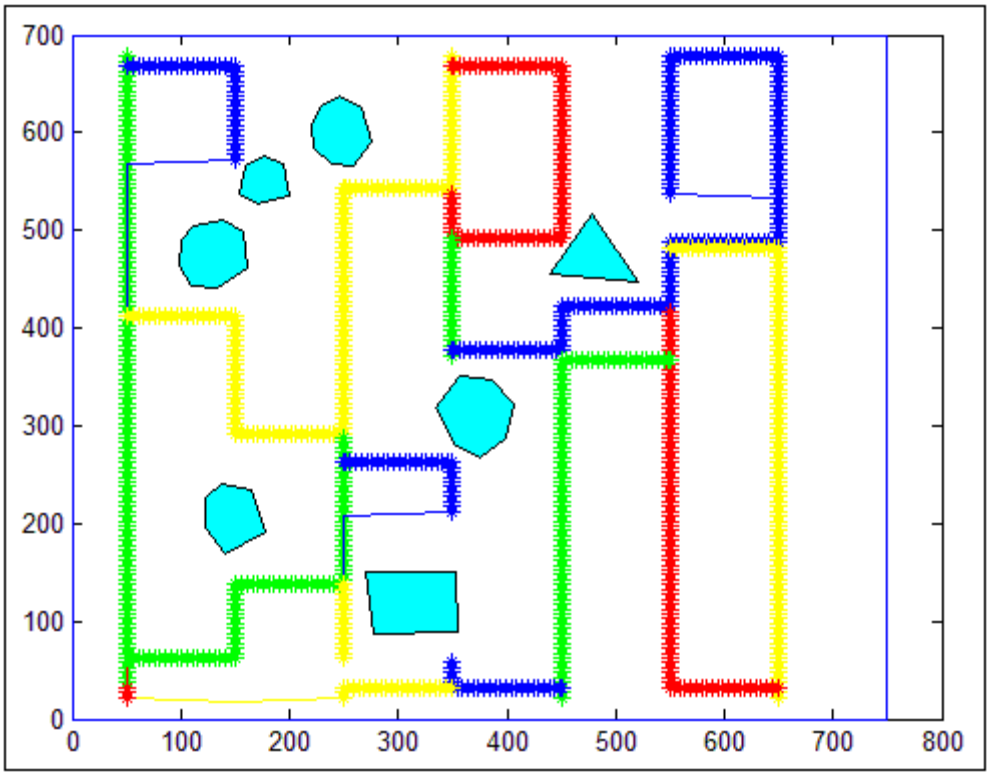


Figure 5.9 Area coverage of a second obstacle course by the BA* algorithm

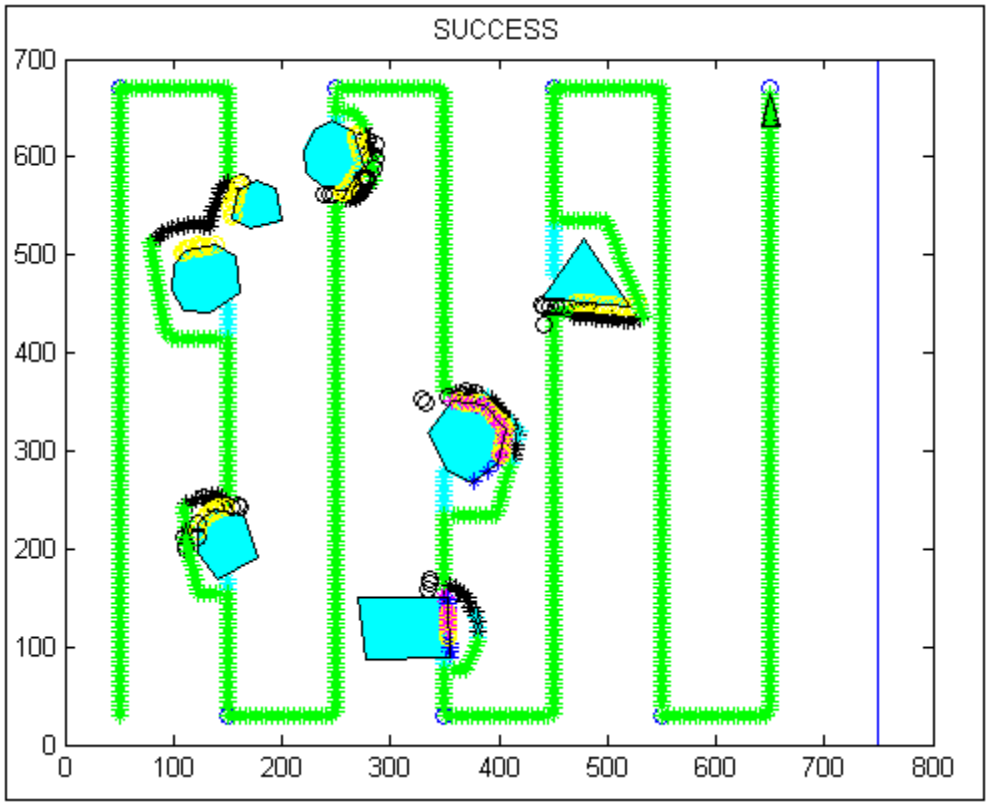


Figure 5.10 The same area as in Figure 5.7 covered by the boustrophedon area coverage using the Hybrid Bug algorithm algorithm

Sample comparisons of the performance of the BA* algorithm and the boustrophedon area coverage using the Hybrid Bug algorithm on the same work area is illustrated in Figure 5.7 and Figure 5.8 as well as Figure 5.9 and Figure 5.10. The different colours in Figure 5.7 and Figure 5.9 indicate the different runs in the BA* algorithm before backtracking to a new point upon reaching a critical point. For the purpose of argument, Figure 5.11 and Figure 5.12 illustrate the performance of the two approaches to area coverage on larger obstacles. While a few large obstacles were included in the simulations, the majority of the simulations had courses with obstacles similar to those described in Appendix II. Simulations showed that on an average 96% of the intended area was covered. However, the boustrophedon area coverage using the Hybrid Bug algorithm produced path lengths that were only 0.67% smaller than the BA* algorithm on an average due to the fact that wide obstacles are circumvented multiple times.

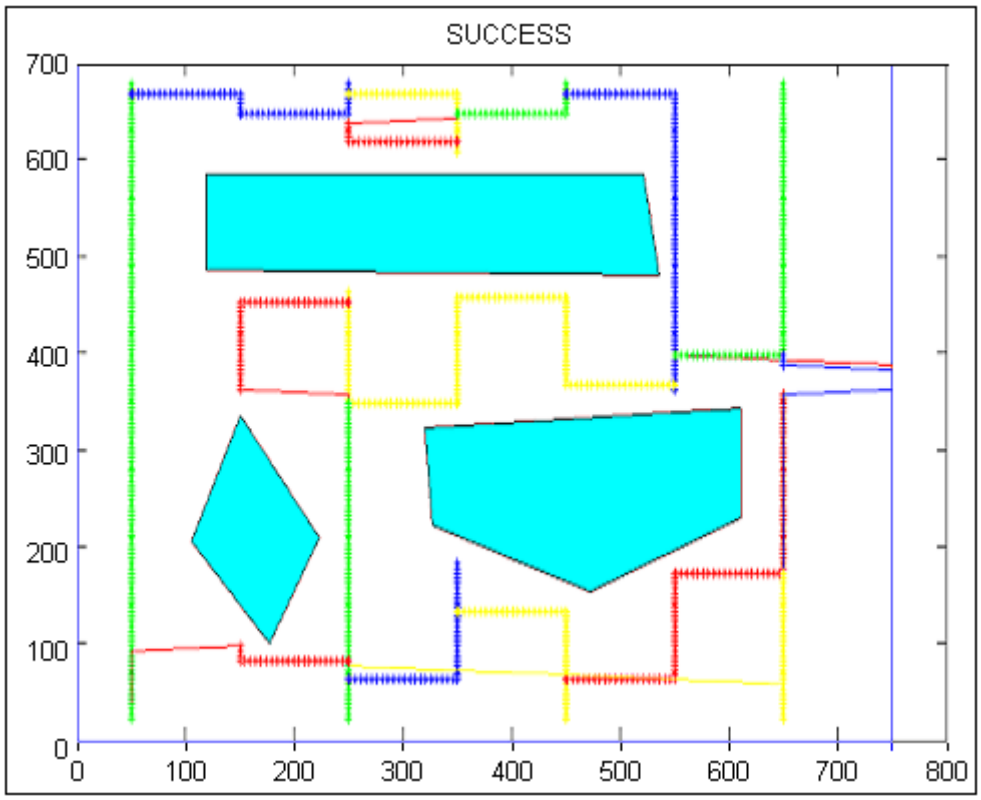


Figure 5.11 An example of BA* coverage of areas with large obstacles

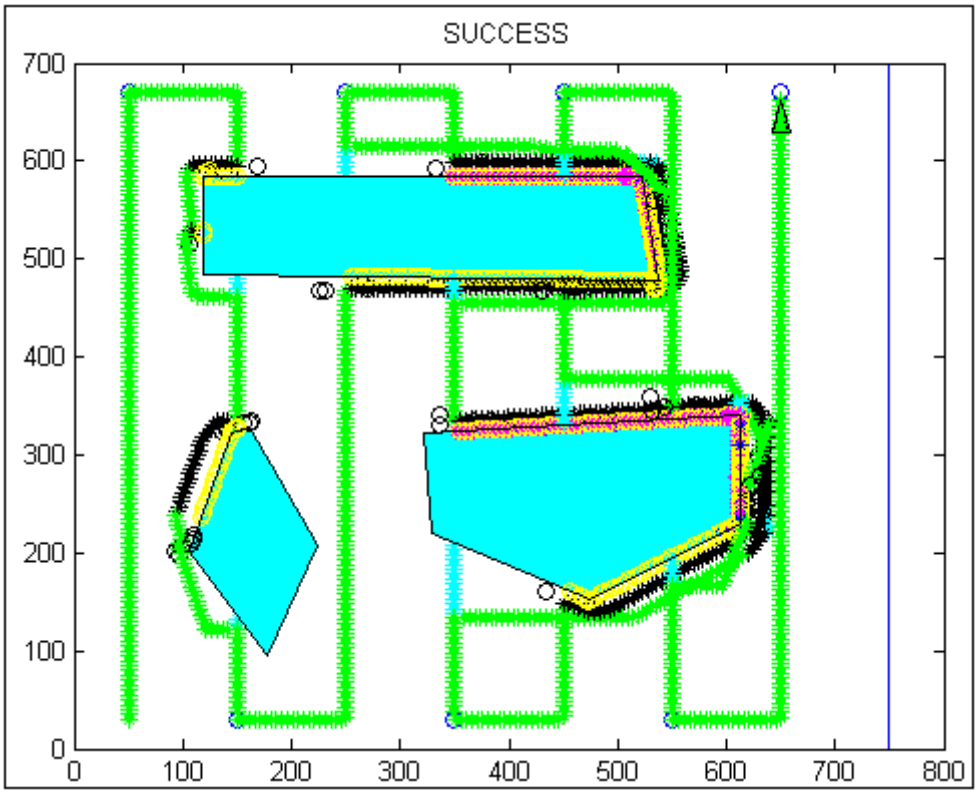


Figure 5.12 The same area as in Figure 5.11 covered by the boustrophedon area coverage using the Hybrid Bug algorithm algorithm

5.2.2.2 Special Cases

This section addresses a few special cases that can arise in the target applications and the performance of the performance of the boustrophedon area coverage using the Hybrid Bug algorithm in such cases.

The first special case is navigating corners - In the real world, if obstacles lack plain faces and are mostly curved, the Hybrid Bug algorithm will use the centreline model to navigate during boundary following behaviour. The centreline model is also used to navigate corners. This has worked to navigate corners and polygonal approximations of curved surfaces in most of the simulations. However, issues arise when corners or curved surfaces lie very close to other obstacles as shown in Figure 5.13 where a narrow passage between two obstacles appears blocked due to a combination of high turn radius of path followed using the centreline model and poor azimuth resolution.

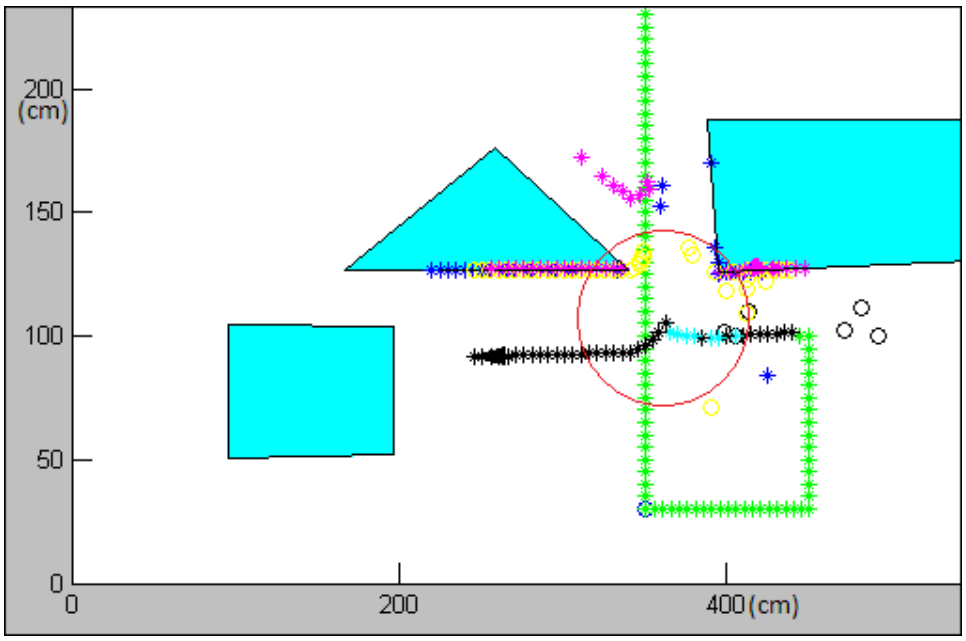


Figure 5.13 A narrow passage between two obstacles

As shown in Figure 5.13, the gap between the triangular and quadrilateral obstacles is missed while approaching it from below, since the robot is in the boundary following behaviour of the Hybrid Bug algorithm and uses the techniques outlined in section 4.3. To recapitulate, when boundary following behaviour is initiated, the robot uses the sensor readings on both sides of the sensor with the shortest admissible range reading as follows: tangents to the sensor arc with the shortest admissible range reading are drawn through external points lying on the extremities of two neighbouring sensor arcs that are closer to the central arc. If these two tangents produce points of contact where they are tangential to the central arc that lie within a radius of tolerance of each other and have slopes that are close to each other (ideally,

the slopes would be equal), it is assumed that the planar face of an obstacle has been detected and the robot moves along the slope of the tangents. However, if the tangents do not coincide within the defined parameters of tolerance but one of the tangents has the same slope as the previously calculated slope, this is assumed to be the boundary face to be followed. This assumption has the effect of causing the robot to ignore the readings of the sensor that first detects a new face and continue following the current face, only attempting to turn the corner when both slopes begin to deviate from the previously followed direction and the robot begins to follow the centreline model to turn. This has proved an effective strategy in most cases. However, when the free space between obstacles is narrow as in Figure 5.13, it can cause the robot to perceive the space as blocked by obstacles.

There are methods to improve this navigation. One such technique is an opportunistic approach where the algorithm looks for collinearity of a set of three points detected from three consecutive robot positions, on the new boundary. These points are indicated in Figure 5.14 by the blue stars on the boundary A. Each of these points lies on the extremity of the sensor arc neighbouring the sensor with the lowest admissible range reading as above. However, the slope of the tangent to the central sensor arc, through these points would have changed discontinuously unlike the slope of the tangent through the other neighbouring point. Once three collinear points on the new face are established, there is a greater level of confidence that these new readings are reflected from a new face of the obstacle and are not due to interference from other obstacles. The advantage of using this method is that it allows the robot to perceive narrow openings at low computing cost. One limitation is that if the face A of the obstacle is curved or very small and does not allow for the detection of three collinear points, this method will not be applicable.

This method of following obstacle corners is an optional enhancement of the navigation technique and has been used successfully in over 60 instances of the simulations and shows a marked improvement at larger boundary following distances. For a boundary following distance of 40cm, this method has shown a 40% reduction in turn radius in simulations. Simulations using different boundary following distances were performed on a small set of 4 obstacle courses with multiple obstacles and the results are summarised in Table 5.4.

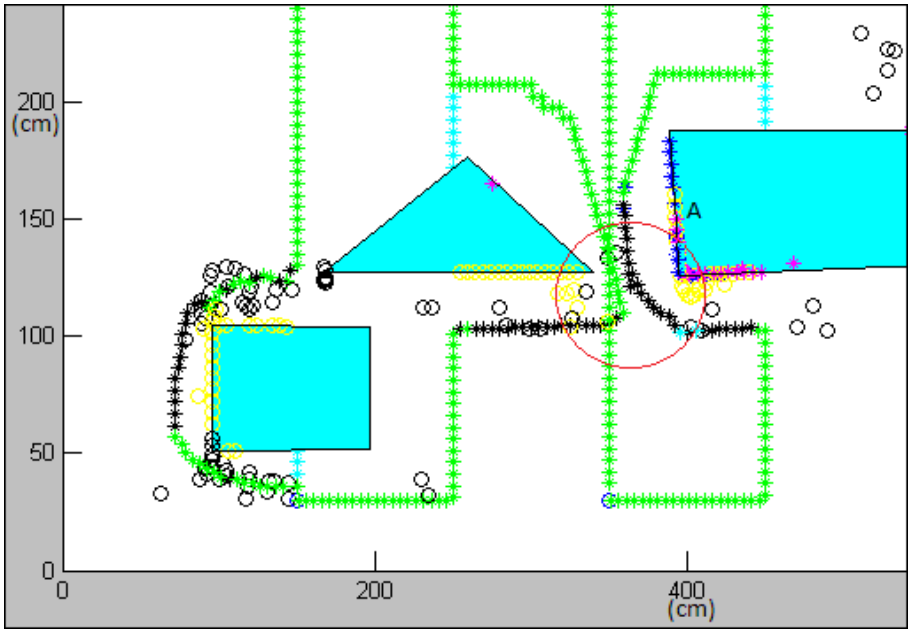


Figure 5.14 Navigation of corners using collinear points on the new face A

Table 5.4 Improvement of turn radius during boundary following behaviour by detecting the new obstacle face using collinear points

Boundary following distance (cm)	With collinear point detection (cm)	Without collinear point detection (cm)	Improvement (%)
15	12	17	29.41
30	23	33	30.30
40	32	53	39.62

The second special case involves approaching the m-line - The Hybrid Bug algorithm uses the greedy approach to maximise distance traversed along the m-line. This means that when an obstacle no longer blocks the shortest path to the m-line, along the perpendicular or any admissible path to the m-line, the robot proceeds in that direction. However, with a combination of good range resolution and poor angular resolution, the SRF05 sensors can cause the robot to miss passages between obstacles, as seen in Figure 5.15, where due to the relatively larger distance to the obstacles, the sonar arcs that detect them are also quite large as they subtend an approximate angle of 55°. Note that at this point, the robot is no longer following the boundary. This makes the obstacles appear much larger than they are. One method to overcome this is to approach the m-line in instalments. That is, the robot makes

stepwise progress towards the m-line when the distance along the normal to the m-line equal to that pre-decided step value is free of obstacles. From that point, the robot decides whether to progress towards the m-line or to use the LTG to avoid the obstacle. This is seen in Figure 5.16. In cases where the boundaries of the obstacles are very contoured, this behaviour may offset the benefits of effectively avoiding local minima on obstacle boundaries that is provided by the Hybrid Bug algorithm, and therefore this modification to the algorithm can be reserved for cases where the obstacle shapes are fairly regular and the algorithm does not need to be strictly online. As the intention was to keep the algorithm strictly online, this modification was not used extensively in the simulations.

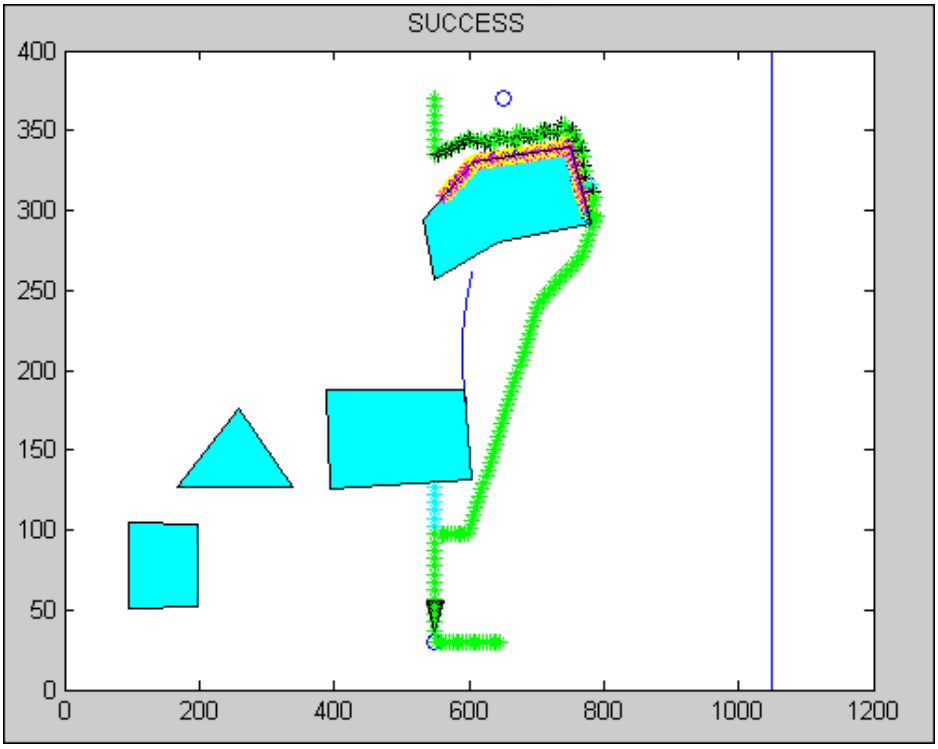


Figure 5.15 A section of the m-line is missed in between the 2 obstacles since the larger distance to the obstacles makes them appear large enough to block the passage.

The third special case is when the start/ target of an m-line is blocked or cordoned off - When either the start point or the target of an m-line, which forms one of the end points of one boustrophedon sweep, is blocked, the boustrophedon motion continues by shifting the focus to the next sweep, as outlined in Algorithm 5 of Chapter 4.

During the simulations, the algorithm was further optimised by adding a clause to change the direction in which the robot is following the boundary and set a flag, if the robot is following the border of the coverage area for more than a threshold distance. This allows the algorithm to avoid having to skirt the border of the entire coverage area.

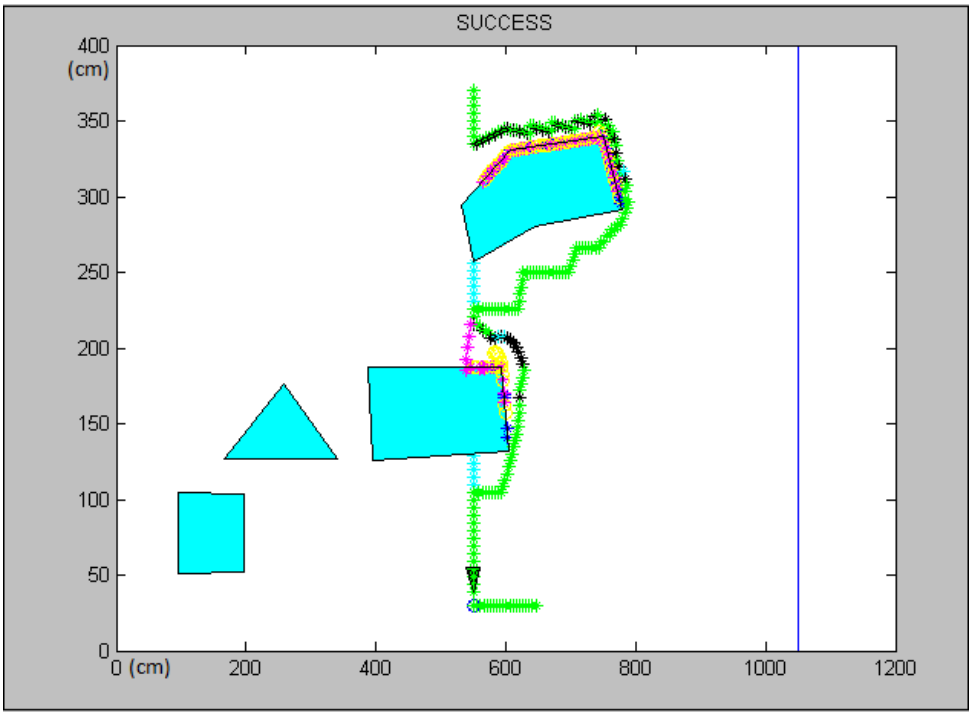


Figure 5.16 m-line coverage is maximised by the robot greedily approaching the m-line in shorter segments along the normal.

If the robot happens to follow a second boundary the threshold number of times while attempting to circumvent a single perceived obstacle, it is assumed that the target is unreachable (as shown in Figure 5.17). In such a case, as described in section 4.4, the target is simply shifted to the next available key point in the boustrophedon coverage path. i.e. the start point of the next boustrophedon track. If this point is also determined to be unreachable, the target is shifted to the target point of the same track and so on. This logic is illustrated in Figure 5.17 and respectively.

The scope of this research is such that the entire search area cannot be cordoned off by obstacle(s). In such a case, new boundaries representing the new reduced search area will have to be predefined.

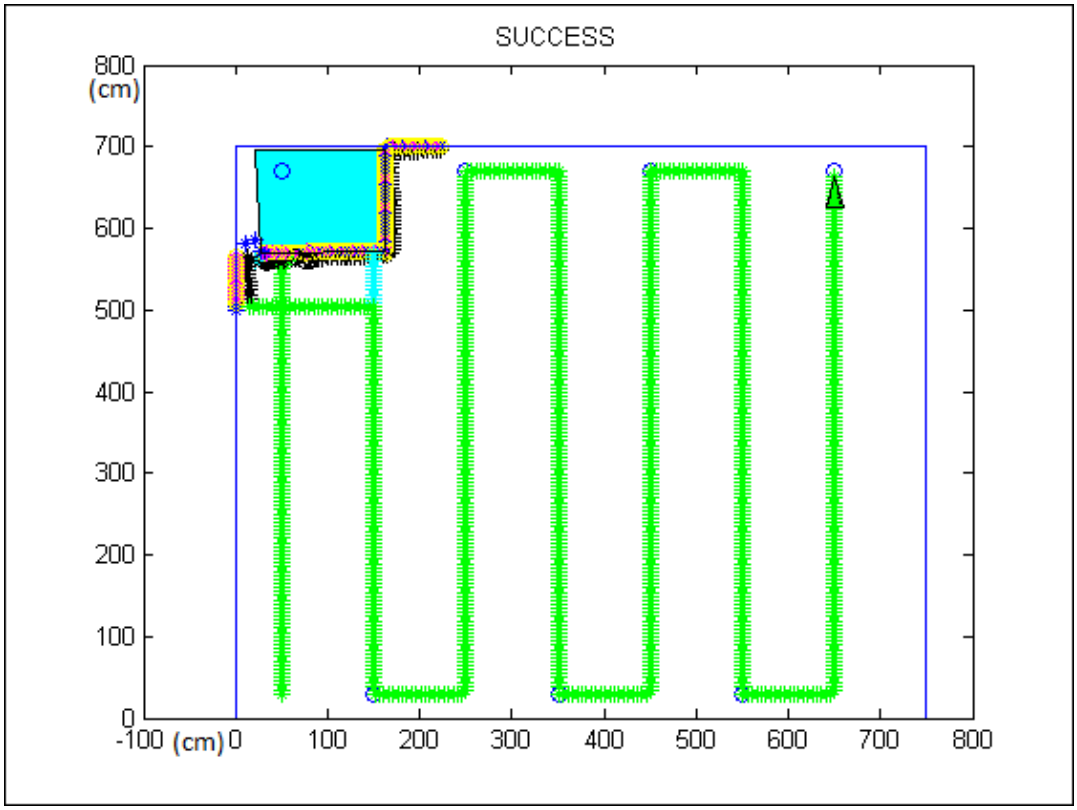


Figure 5.17 Boustrophedon path generated by the boustrophedon area coverage using the Hybrid Bug algorithm when the target of the first track is blocked by an obstacle

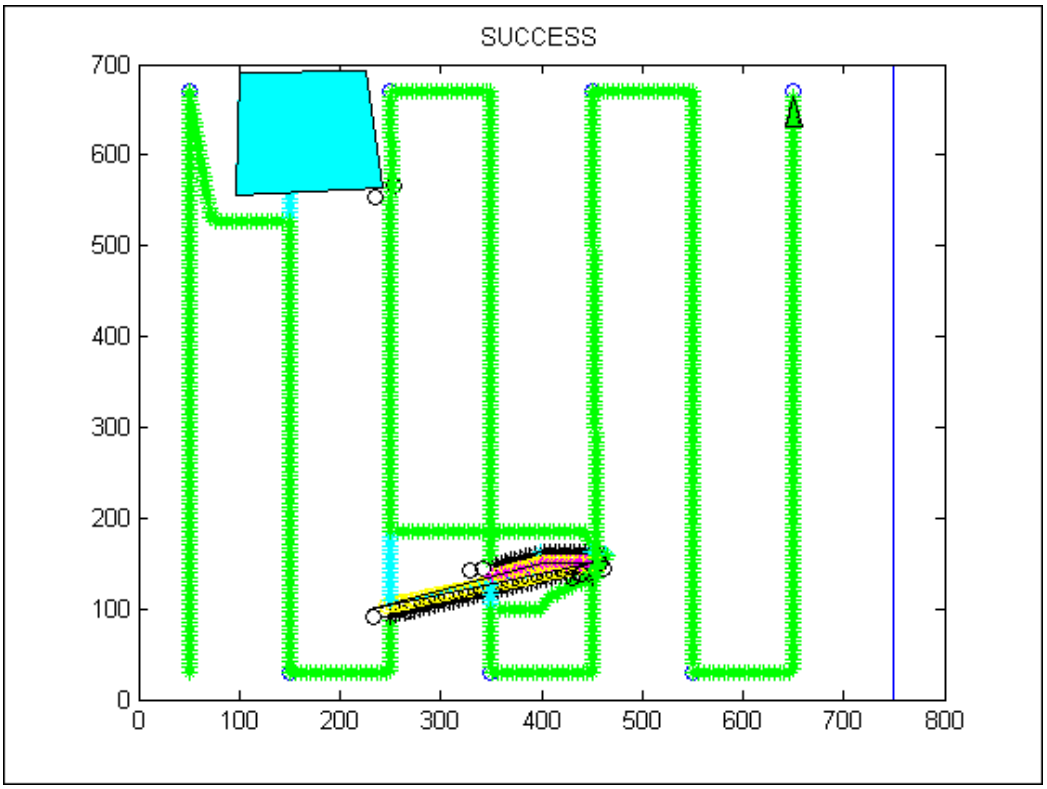


Figure 5.18 Boustrophedon path generated by the boustrophedon area coverage using the Hybrid Bug algorithm when the start of the second track is blocked by an obstacle

The final special case analyses the performance of the boustrophedon area coverage using the Hybrid Bug algorithm when it encounters *U-traps* - Approaches to obstacle avoidance have often stumbled with issues of the robot being trapped in areas of local minima such as concave obstacle faces. This problem is called the U-trap problem and is especially common in algorithms based on random repositioning to exit the local minima trap. Artificial Potential Field based algorithms [90] and particle swarm optimisation algorithms [91] suffer from such issues. The boustrophedon area coverage using the Hybrid Bug algorithm, however escapes this problem primarily due to the boundary following behaviour and this is illustrated in Figure 5.19, where the depth of the U-trap is deliberately chosen to be greater than the sensor range (1m in this case).

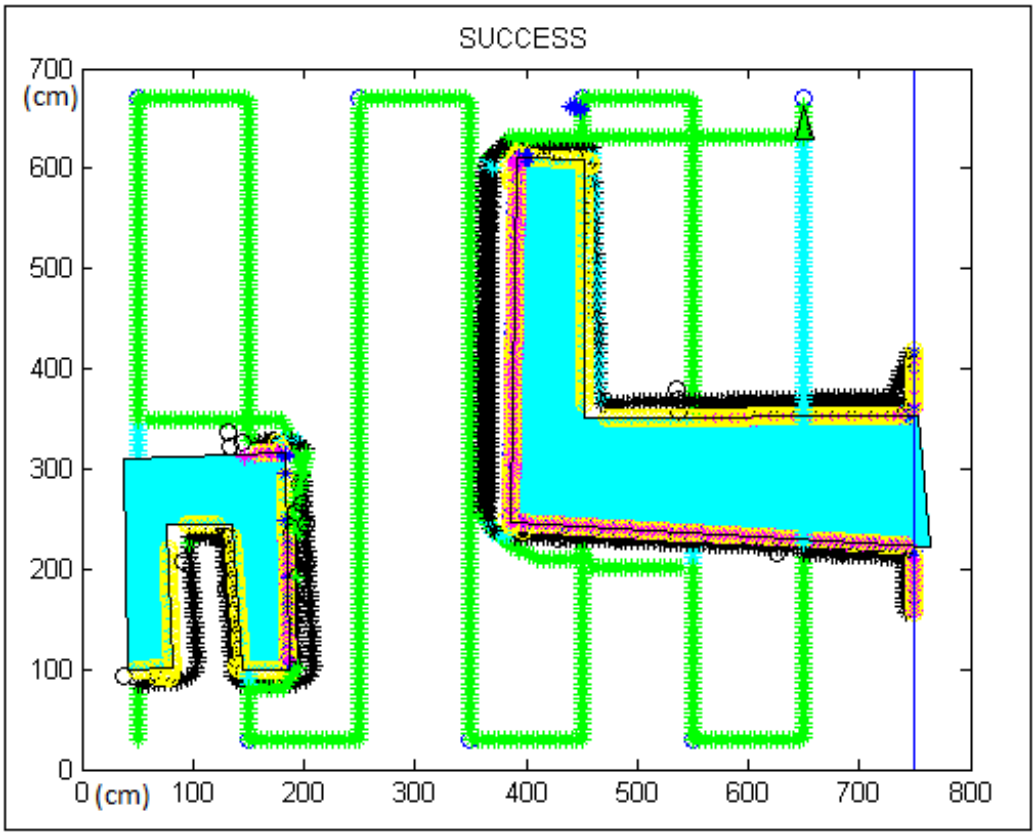


Figure 5.19 Performance of the boustrophedon area coverage using the Hybrid Bug algorithm algorithm in U-trap situations

5.3 Conclusion

This chapter describes the two sets of simulation studies carried out to verify the efficiency of the Hybrid Bug algorithm (Algorithm 3) and the algorithm for boustrophedon area coverage using the Hybrid Bug algorithm (Algorithm 5) formulated for this research. The simulation study for the Hybrid Bug algorithm (Algorithm 3) compared it with the Bug2 and TangentBug algorithms in order to assert that a combination of the two algorithms gave a better overall result. The sensors were modelled with an ideal angular resolution of 1°, in keeping with the algorithm's description in Chapter 3. Obstacles were typically 1m to 3m in width and results were evaluated for obstacles along one set of start and target points, 200cm or 400cm apart. Several simulation courses comprising of combinations of typical obstacles shown in Figure 5.6 were designed. Over 40 simulations were carried out. The Bug2 algorithm and the Hybrid Bug algorithm were executed over these courses. The results showed that the average path length for the Hybrid Bug algorithm 437 and that for the Bug2 algorithm was 515.The Hybrid Bug algorithm was observed to have a path length 15% less than the path length for the Bug2 algorithm on an average. A subset of these results is presented in table A.1 in Appendix V. When the TangentBug and the Hybrid Bug algorithms were executed over the same courses, the Hybrid Bug covered 91% of the obstacle free space on the m-line on an average, which is 83.07% higher than that achieved by the TangentBug. A subset of these results is presented in table A.2 in Appendix V. Additionally, from simulation studies of over 60 simulations, it was found that the boustrophedon area coverage using the Hybrid Bug algorithm (Algorithm 5) produced an average coverage of 95.54%. In simulations where all obstacles were small, typically less than 1 track width, the average area coverage was found to be 96.73% and simulations with large obstacles blocking significant lengths of the m-line produced area coverage of 92.92%. These figures were obtained by using the path that the robot had taken and factoring in the assumed sensor range of 1m. Areas that were regions shadowed by obstacles, that were not covered by passes on subsequent tracks were also accounted for. This algorithm was further compared with a contemporary online area coverage algorithm, the BA* algorithm and resulted in an improved overall result. The boustrophedon area coverage using the Hybrid Bug algorithm was found to cover 7.13% more of the m-line than the BA* algorithm and showed an 18.26% reduction in path length over the BA* algorithm for typical obstacle distributions. The dimensions of the robot have not been accounted for in these simulations as they have been idealised to the case of a point mobile automaton. This allows for a more generalised

evaluation of the results. As a case for future simulations, different robot dimensions can be considered, but it is expected that this will not have a great impact on the area coverage.

The boundaries have been perceived with an accuracy of over 99% by the Hybrid Bug algorithm with simulated SRF05 sensors. The accuracy was measured as the number of points of the obstacle boundary that were located with an error of over half the track width, which is 50cm in this case. While the simulation results for obstacles spanning one to three track widths showed virtually no perceived boundary points that were grossly erroneous, the simulations having obstacles that could be bounded by a circle of 50cm showed points that were off the mark by more than half the track width. The rate of occurrence of such points was found to range from 0.1 to 0.5%.

Therefore, it is noted that the Boustrophedon coverage of agricultural areas using the Hybrid Bug algorithm for obstacle avoidance covered over 95% of the area to be covered on an average. Not only was this higher than the coverage produced by contemporary online coverage algorithms such as the BA* algorithm, the boustrophedon area coverage using the Hybrid Bug algorithm attained this coverage at shorter path lengths.

6 Conclusion

This chapter discusses the significant contribution of this research and the future work that can be done to enhance the research. The primary aim of this research was to develop a strategy for complete or near complete area coverage to be performed by low cost autonomous robots in an indoor or outdoor environment where obstacle distribution, positioning and orientation are unknown. The expectation was that it would deliver optimal performances when the navigable space was large and obstacle widths were very small in comparison, such as agricultural areas.

6.1 Main contribution of this research

Several area coverage strategies were described in Chapter 2 and a few candidate algorithms were analysed in Chapter 4. The efficacy of a simplistic boustrophedon coverage pattern was examined in the context of the target application. This discussion is presented in Chapter 4. There was found to be foreseeable advantage in using boustrophedon pattern for area coverage with optimal obstacle circumvention when encountering an obstacle.

The new Hybrid Bug algorithm was developed and simulated to leverage on the benefits of the Bug2 and TangentBug algorithms for obstacle circumvention with the aim that the avoidance paths would possess shorter lengths. Chapter 3 shows that this new Hybrid Bug algorithm terminates in the ideal case of unlimited sensor range and infinite angular resolution. The concept of the m-line was adopted from the Bug2 algorithm as it was a convenient way to represent one boustrophedon track. Since the spacing of the boustrophedon tracks can be configured based on the range of the sensors responsible for monitoring the area, the extent to which the boustrophedon tracks are covered would be a gross representation of the extent of area coverage. Consequently, the percentage of the m-line covered reflects the extent of the area covered in most cases. Therefore, efficiency of the Hybrid Bug algorithm was measured by two parameters – The extent of m-line coverage achieved and path length. The results from over 40 simulations revealed that, with reference to the path length parameter, the Hybrid Bug algorithm showed an improvement of 15% around simple obstacles and 25% around convoluted obstacles, on an average, when compared to the Bug2 algorithm. The comparison of the Hybrid Bug algorithm with the TangentBug algorithm revealed an improvement of over 83% in m-line coverage on an average, while the path length of the Hybrid Bug algorithm was only 4% longer than the path length generated by the TangentBug algorithm on an average for courses containing simple

obstacles. Therefore, it can be concluded that the increase in path length is negligible when compared to the substantial increase in m-line and subsequently area coverage that was achieved.

The Hybrid Bug algorithm, which navigates purely based on local sensor data was then reinforced by the availability of a limited amount of global data as a consequence of the boustrophedon motion. This whole scheme was put to the test by simulating side by side, the boustrophedon area coverage using the Hybrid Bug algorithm and one of the better online area coverage algorithms called the BA* algorithm. Factors such as the low angular resolution of the potential sensors on the robot were taken in to account in both cases. The results of simulation from over 60 simulations were in the favour of the boustrophedon area coverage using the Hybrid Bug algorithm, with it claiming up to 33.3% reduction in path length and 9% increase in area coverage as compared to the BA* algorithm, in the target environment. From these simulations, it was also found that the Boustrophedon coverage using the Hybrid Bug algorithm produced an average coverage of 95.54%.

This research has therefore successfully developed a new navigation algorithm for the efficient, near complete coverage of agricultural spaces comprising of vast navigable areas and few obstacles. The algorithm has a substantial edge over some other existing online area coverage algorithms that can be employed for the monitoring of free range agricultural farms by autonomous robots. Additionally, the Hybrid Bug algorithm could have applications in search and mining operations where the landscapes may be similar to those of free range farms.

6.2 Future work

This section introduces the potential future directions for this research.

Most simulations in this research focussed on the coverage of rectangular areas. A variation of the boundary shapes of the areas to be covered can be considered as an extension to this research.

If obstacles are closely spaced, depending on the distance from the boundary, they may be recognized as a single obstacle, as described in section 5.3.1. For a boundary following distance of 15cm., when obstacles are less than 14.4cm. apart, they can be clustered into a single obstacle. Figure 6.1 illustrates this phenomenon. While the size of the robot itself can restrict the robots entry into the passages between obstacles, the algorithm might occasionally

overlook passages that can allow the robot to pass through. This issue is a direct consequence of the feature of the algorithm that attempts to circumvent obstacles using an optimal path. Closely following the periphery of the obstacles can avert this problem at the cost of an increased path length. Research into perceiving small gaps between obstacles using low cost ultrasonic sensors can be undertaken if this issue is found to have substantial consequences in real world implementations.

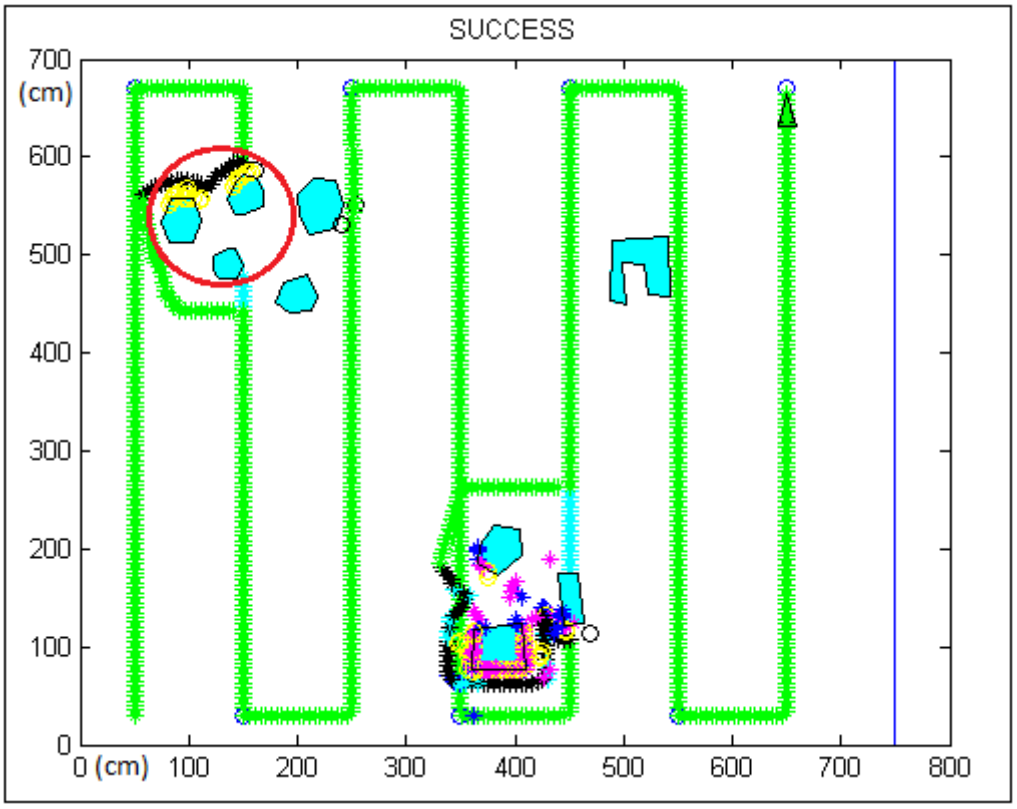


Figure 6.1 Clustered obstacles perceived as one – encircled in red

Aspects of the developed area coverage scheme such as being strictly online and not building maps and a greedy approach to the boustrophedon track whenever possible and not being bound to obstacle perimeters make it ideal for use with mobile obstacles such as livestock. However, experiments relating to the detection of soft tissues by the ultrasonic sensors have not been performed to determine how accurately mobile obstacles might be perceived. Literature on this matter is also sparse and therefore, in order to show/develop the resilience of this algorithm with any degree of credibility, practical experiments involving livestock need to be performed.

Additionally, the simulation studies developed in this research idealise the robot to a point and ignores the possibility of dead reckoning errors and wheel slippage. These issues can be addressed by the implementation of the algorithm on a mobile robot.

In cases where the width of the obstacle is typically greater than three times the sweep width, elements of backtracking algorithms can be employed. However, an effect of such an integration will have to be verified and the new version of the algorithm proven to be convergent. However, such obstacles do not commonly occur in the target applications considered in this research.

Appendix I. Boustrophedon track width for simulations

A track width of 1m was used in all the simulations of the boustrophedon area coverage using the Hybrid Bug algorithm. The reasoning for this is that when a prototype of the robot was constructed, various cameras were evaluated on parameters such as

1. Ability to identify smaller objects such as eggs on a poultry farm
2. Cost
3. Performance outdoors

The BrickCom OB-100A Outdoor bullet camera [100] was considered to be the best suited camera amongst those surveyed as it was primarily an outdoor bullet camera with IP67 enclosure that makes them weather and dust resistant [100]. It also had an adjustable megapixel resolution (a maximum resolution of 1280×800) [100] which made zooming effective when identifying smaller objects and was cost effective. Additionally, this camera provides a 3.6× optical zoom.

The camera uses a motorised lens with Auto-iris that enables users to adjust the focal length via the user interface [100]. The focal length ranges from 3.3mm ~ 12 mm. The angle of view is 63.6° ~17.9° horizontally, 46.5° ~13.5° vertically and 82.7° ~ 22.4 ° diagonally.
A suitable location for the camera on the prototype was found to be in the front, at a height of 1m from the ground, having accommodated the battery and control circuitry. When the camera is placed on robotic platform at a height of 1m at an angle of 45° in order to inspect the ground, as shown in Figure A.1, it gives a horizontal field view of 1.7m and vertical view of 1.38m. At a resolution of 1280×800, this makes an egg of approximately 45mm equal to about 33×26 pixels. From visual inspection and with the aid of algorithms like EDISON [95], this appears to be sufficiently large to extract predominant features. A 30cm chicken will approximately be 226×173 pixels. With 3.6× optical zoom, after an initial image recognition process identifies objects, the optical zoom can be used to get a closer look.

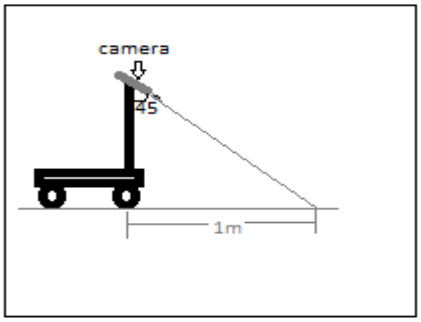


Figure A.1 Positioning of the camera on the robotic platform

Since the horizontal field of view is 1.7m, having a track width of 1m ensures a 0.7m overlap (0.35m on either side) which is very advantageous. Therefore, a track width of 1m was chosen in simulations.

Appendix II. Typical obstacles on a farm

The algorithms developed in this thesis were specialised to perform optimally in the target applications of free range farming, specifically free range poultry farms that include barns. The typical stationary obstacles on poultry farms include feeding stations, watering stations and trees. Additionally some equipment to move supplies and waste such as wheelbarrows might remain on the farm for short periods of time. It can be seen that these obstacles are all very small in dimensions relative to the area to be covered, often less than 1 square metre. Clusters of trees can sometimes exist on the farm and the dimensions of these clusters are typically less than 3 square metres. Therefore, the stationary obstacles on the farm are typically expected to be less than 3m wide, which is equivalent to three track widths (Appendix I).

Appendix III. Development of a low cost robotic platform

The specification for the mechanical design of the robot is described first. The initial ground work was done by Benoit Challian in his report[13]. The requirements governing the design of the robot are:

i. The robot should support a payload of 20 kg in addition to its own weight of 20 kg.

ii. The robot should be able to climb an incline of 30%.

iii. The robot is expected to move close to a speed of up to 2 km/h and attain this speed in 2 seconds.

iv. The robot is to be running for at least 2 hours before requiring a recharge.

A decision was made to have a two wheel differentially driven system. With these requirements in place, the next step was to determine the torque required to move the robot forwards in the worst case, which is when the robot is on a 30% incline. A free body diagram was used to analyse the forces acting on the system in order to calculate the torque to be exerted by each of the two motors of the two wheel drive system.

The three kinds of forces acting on the system are

i. The Gravitational force acting on robot: The robot and its payload together weigh 40kg. While investigating the parts and their procurement, it was discovered that the weight of the robot could be reduced to around 12kg, making the weight of the system for the prototype 32kg. When the robot climbs an incline, the gravitational force acting on the robot due to its mass will cause it to roll backwards.

ii. -Dynamic force: The force required to propel the system forward

iii. Frictional force: The force acting on the wheels of the robot due to friction from the rough surface it traverses. The frictional force opposes the direction of motion of the robot and is proportional to the component of the gravitational force that is normal to the surface. The frictional force will be directed against the direction of the motion of the robot.

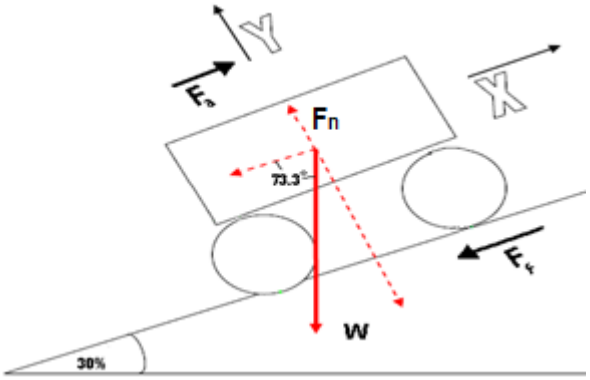


Figure A.2 A free body diagram showing the different forces acting on the robot

Let W be the gravitational pull on the robot. Let m be the mass of the entire system and g be the acceleration due to gravity. If W_x is the component of the force along the x axis, when the robot is climbing an incline of 30%, or 16.7°,

$W_x = W\cos(90-16.7°)$ (from fig. 2.12)
$W = m \times g$
$\therefore W_x = m \times g \times \sin(16.7°)$
$W_x = 32 \times 9.81 \times 0.29 = 91N$

In a 2 wheel drive system, the equivalent force produced by each driving motor is halved. Hence, if F_{eq} is the equivalent force produced by each motor to just keep the system in equilibrium, it would effectively be

$F_{eq} = 91/2 = 45.5N$

The robot is to attain a maximum speed of 2 km/h, or alternatively, 0.55 meters per second in 2 seconds. Let a be the acceleration required to achieve this speed. If F_{acc} is the dynamic force required to produce this acceleration and F_{acc} is the force on each wheel,

$a = 0.55/2 = 0.275 m/s^2$
$F_{acc} = m \times a$
$F_{acc} = 32 \times 0.275 = 8.8N$
$F_a = F_{acc}/2 = 8.8/2 = 4.4N$

Therefore, an accelerative force, F_a of 5.5N would be required and would have to be produced by each motor.

At every point of contact that the robot has with the ground, it experiences a force due to the friction of the surface. Since the robot is built for a rough terrain, the coefficient of kinetic

friction: $\mu = 0.4$ can be assumed. If F_n is the normal force acting on the system and F_f is the frictional force acting on the system, and F_{feq} is the frictional force to be overcome by each wheel,

$F_n = W_y = W\cos(16.7)$ (From figure 2.12)

$F_n = mg\cos(16.7)$

$F_f = \mu \times N$

$F_f = 0.4 \times 32 \times 9.81 \times \cos(16.7) = 120.27N$

$F_{feq} = 120.27/2 = 60.14N$

Therefore, the total force acting on the system, F_T is

$F_T = 45.5 + 4.4 + 60.14 = 110.04N$

The radius of the proposed wheel to be used is 0.13m.

If T is the torque required by each motor,

$T = \text{Total Force} \times \text{Radius of the wheel}$

$T = 110.04 \times 0.13 = 14.3Nm$

Therefore a torque of 14 Nm will be required with a wheel of 26cm diameter.

The robot is required to attain a speed of 2 km/h or 0.55 m/s. The wheels would cover a distance equal to their circumference per rotation on even ground.

$\text{Distance covered per turn of the motor} = 2 \times \pi \times radius$

$= 2 \times \pi \times 0.13 = 0.816m$

This implies that to achieve 0.55 m/s or 33m/min,

the required motor speed = 33/0.816 = 40rpm.

Therefore, a motor and gear head able to produce a torque of 13Nm at rotational speeds of 40rpm should be considered.

The most suitable motor found for these specifications was the Z5D60-12G motor [104] which is operated with 12V, 60W and 9A loaded. It offers 11.57 Nm torque at 33 RPM - which means that the robot may need to slow down to 1.615 km/h since:

$\text{Speed} = \text{Circumference of the wheel} \times \text{rotational speed}$

$= 0.816 \times 33$

$= 26.93\ \text{m/min}$

$= 1.615\ \text{km/h}$

At the outset, the torque may look like it is going to further reduce the speed of the robot. However, due to the low voltage and current requirements of this motor, the battery is likely to be lighter and will sufficiently reduce payload.

For the motor chosen above, a 12V battery would be required. The battery would have to power the robot for at least 2 hours. The parameters that define the battery capacity are the current drawn by each motor, the number of motors, the time for which the robot is to be running and depth of discharge of the battery. The motor chosen draws a current of 9A. If this motor is considered, with a typical deep cycle lead acid battery with a 50% depth of discharge or a recommended utilisation factor of 2, the battery capacity can be obtained by

$$\text{Battery Capacity} = \text{Current drawn} \times \text{Time} \times \text{Number of motors} \times \text{Recommended use}$$
$$= 9\text{A} \times 2h \times 2 \times 2$$
$$= 72\text{Ah}$$

If instead, a LiFePO4 battery is considered, an 80% depth of discharge can be leveraged. This means that in addition to being lighter, this battery offers a utilisation of 1.25.

`

12V, 45AH, LiFePO4 batteries are commercially available and sometimes weigh as less as 6.5 kgs, improving greatly over their lead acid counterparts.

To summarise, Z5D60-12G motors [104], 26cm diameter wheels and a 12V, 45AH, LiFePO4 battery were sourced for the development of this robot.

The differential drive is controlled by a Sabertooth 2×12 motor driver [105]. This motor driver is in turn controlled by the different voltage levels resulting from passing the Pulse Width Modulated (PWM) output of a microcontroller through a low pass RC filter. The microcontroller generating the PWM output signal is a PIC16F877A which processes the feedback from the encoders on the motors and uses a Proportional Integral Derivate controller (PID controller) to generate the response signal. This microcontroller is part of a circuit board called elab D50 [106] developed by a South Australian company called elabtronics. The board is easily programmable using a fourth generation flowchart based language. Additionally, it has a second controller, a dsPIC33FJ128MC, which enables communication with a master controller, which can be a more advanced processer capable of image processing. Several channels are available on the elab D50 to process inputs from various analog and digital on-board sensors. Therefore, it is possible to implement the entire path planning logic on this board. An Intel Atom D525 dual core processor was chosen to process images from a BrickCom OB-100A Outdoor bullet camera [100]. This processer is expected to identify objects and make decisions on approaching them based on their appearance.

Appendix IV. Admissibility criterion for a modified LTG edge

The TangentBug algorithm [86] defines a local tangent graph LTG, whose nodes are defined as the end points of the sensed obstacles, the current robot location *x* and the additional optional node T_{node} in the direction of the target. An edge is said to be admissible if travelling along the edge monotonically reduces the distance to the target *T*.A subgraph *G1* of the LTG consisting of admissible edges emanating from *x* is defined as

$$V_{G1} = \{\{x\} \cup \{V \in \text{LTG} : (V - x).(T - x) > 0\}\}$$

Where *V* represents an LTG vertex. The criterion $(V - x).(T - x) > 0$ implies that the included angle between the edges *Vx* and *xT* must be less than 90°. However, as the Hybrid Bug algorithm has to make a detour through the m-line whenever possible. Therefore the definition of the optional node T_{node} changes as does the criterion for admissibility.

Let *T* be the target and *x* the current robot position, when it is not on the m-line. Let *B* be any point on the m-line that is visible from *x*. Then the edge *xB* is said to be admissible in the context of the Hybrid Bug algorithm if

$$\angle TxB < \frac{\pi}{2} - \frac{\angle xTB}{2}$$

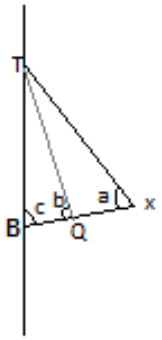


Figure A.3 Admissibility criterion

In order to prove this, we need to prove that with reference to Figure A.3,

if $\angle TxB < \frac{\pi}{2} - \frac{\angle xTB}{2}$ the distance to *T* from any point *Q* on $\overline{Bx}$ is less than $\overline{Tx}$. i.e.

given $\angle TxB < \frac{\pi}{2} - \frac{\angle xTB}{2}$, $|\overline{QT}| < |\overline{Tx}|$ for all $Q \in \overline{Bx}$

Proof: Let Q be a random point on $\overline{Bx}$. Denote the angle $\angle TxB$ as a, $\angle TQB$ as b and $\angle TBQ$ as c.

Given $a < \frac{\pi}{2} - \frac{\angle xTB}{2}$

$$2a < \pi - \angle xTB$$

$$2a < (a + \angle TBx)$$

$$a < \angle TBx \qquad \text{(A.1)}$$

$\sin(\angle TQB) = \sin(180 - \angle TQB) = \sin(\angle TQX)$

$$\sin(b) = \sin(\angle TQX) \qquad \text{(A.2)}$$

From (A.1) and (A.2),

$\sin(b) = \sin(\angle TQX) > \sin(a)$ (and will become less than or equal to sin(a) only when $\overline{TQ}$ crosses over $\overline{TB}$)

$$\frac{|\overline{BT}|}{\sin(b)} < \frac{|\overline{BT}|}{\sin(a)} \qquad \text{(A.3)}$$

Also, from applying the sine rule to triangles BTx and BTQ,

$$\frac{|\overline{BT}|}{\sin(b)} = \frac{|\overline{QT}|}{\sin(c)} \text{ and } \frac{|\overline{BT}|}{\sin(a)} = \frac{|\overline{Tx}|}{\sin(c)}$$

From (A.3)

$$\therefore \frac{|\overline{QT}|}{\sin(c)} < \frac{|\overline{Tx}|}{\sin(c)}$$

Or $|\overline{QT}| < |\overline{Tx}|$ for all $Q \in \overline{Bx}$

Appendix V. Subset of the simulation results obtained

This section presents some of the results obtained during the simulation and comparison study of the Hybrid Bug algorithm and the boustrophedon area coverage using the Hybrid Bug algorithm.

Table A.1 Comparison of the path length generated by the Bug2, TangentBug and Hybrid Bug algorithms

Bug2 (cm)	TangentBug (cm)	Hybrid Bug (cm)
620	360	318
296	338	304
250	196	216
270	234	260
322	288	342
384	340	418
460	360	400
376	340	316
560	244	286
490	440	414
1256	770	978
1196	720	880

Table A.2 discusses the m-line coverage by the three listed algorithms and gives a percentage of free space (space not occupied by obstacles) on the m-line covered by the algorithms.

Table A.2 Comparison of m-line coverage by the Bug2, TangentBug and Hybrid Bug algorithms

Values in cm			Percentage values		
Bug2	TangentBug	Hybrid Bug	Bug2	TangentBug	HybridBug
56	8	50	100	14.29	89.29
88	8	78	100	9.09	88.64
72	10	66	100	13.89	91.67
64	4	60	100	6.25	93.75
142	8	136	100	5.63	95.77
92	6	80	100	6.52	86.96
120	6	112	100	5.00	93.33
78	2	66	100	2.56	84.62
60	2	56	100	3.33	93.33
104	8	100	100	7.69	96.15
246	2	222	100	0.81	90.24
280	8	256	100	2.86	91.43

Table A.3 Comparison of the path length generated by the boustrophedon area coverage using the Hybrid Bug algorithm and the BA* algorithm

Boustrophedon area Coverage using Hybrid Bug algorithm (cm)	BA* (cm)
5960	7641.6
6010	6680.85
5830	6685.75
5310	6995.85
5488	7009.5
5860	8387.25
6690	6445.85
5085	5840.25
5605	6345.75
5810	7305.85
8790	8096.8
6400	8306.5
5500	8566.35
5205	6080.25
5348	7100.15
6825	8379.5
7055	7770.3
5720	6510.6
5800	7761.1
7650	6606.5
9410	7727.1
6725	8337.35
6315	6796.35
5710	6580.85
5800	7411.65
6215	6800.5
5530	6085.25
5440	7195
5890	7309.35

Table A.4 Comparison of the m-line coverage by the boustrophedon area coverage using the Hybrid Bug algorithm and the BA* algorithm

Boustrophedon area Coverage using Hybrid Bug algorithm (cm)	BA* (cm)
3835	3445
3515	3255
4000	3775
4380	4070
3980	3650
4095	4010
3990	3570
4485	4420
4250	4260
4325	4250
3085	2780
4135	3745
4385	4005
4223	4010
3830	3770
3385	2750
4360	4115
4275	4050
4285	3865
3350	2995
3275	2850
3400	3155
3595	3315
3990	3770
3625	3245
3685	3400
2910	2790
4430	4215
4180	3900

Table A.5 Area covered by the boustrophedon area coverage using the Hybrid Bug algorithm

Area covered by the boustrophedon area coverage using Hybrid Bug algorithm (cm^2)	Total Free Space (cm^2)	Total coverage (%)
331510	372356	89.03
378374.98	399225.87	94.78
348113.75	361993.07	96.17
397365.83	429278.17	92.57
432056.185	441069.8	97.96
409500	428208.78	95.63
423583	446105	94.95
399000	412667.03	96.69
448344.78	451344.78	99.34
438640	447000.93	98.13
425000	434202.92	97.88
423484.5	431943.85	98.04
413500	436048.73	94.83
438500	446201.93	98.27
427500	438589.78	97.47
428500	441286.24	97.1
335000	354565	94.48
327500	365356.2	89.64
340000	360963.8	94.19
359500	382355.57	94.02
399000	408071.47	97.78
367500	400748.92	91.7

REFERENCES

[1] AV Inc., *Nano Hummingbird product page*, Viewed on 14/7/2011, http://www.avinc.com/nano/

[2] Wikipedia: Remotely operated underwater vehicle wiki, Last updated on 18/7/2011, Viewed on 14/7/2011, http://en.wikipedia.org/wiki/Remotely_operated_underwater_vehicle

[3] Mars Rover Home Page, Viewed on 14/7/2011, http://marsrover.nasa.gov/home/

[4] Kelly, A.E., K.E. Iagnemma, and A.E. Howard, *Field and Service Robotics* : *Results of the 7th International Conference*. Springer Tracts in Advanced Robotics Ser. 2010, New York :: Springer

[5] Husqvarna robotic mowers, *Official Website*, Viewed on 14/7/2011, http://www.husqvarna.com/au/homeowner/products/robotic-mowers/husqvarna-robotic-mowers-for-homeowners/

[6] iRobot, *iRobot Roomba 577 product page*, Viewed on 14/7/2011, http://www.roomba.com.au/roomba577.php

[7] Arthur Ed LeBouthillier, Robotics Society of Southern California, Viewed on 1/11/2014, http://www.rssc.org/content/w-grey-walter-and-his-turtle-robots

[8] Coppock, G.E., *Harvesting early and midseason citrus fruit with tree shaker harvest systems*, Floriad Agricultural Experiment Stations Journal Series, vol.2824, 1967

[9] Erik Sofge, *Three new Farm Bots Programmed to Pick, Plant and Drive.* September 2009, Popular Mechanics

[10] Evan Ackerman, *Google's Autonomous Car Takes To The Streets*, Last updated on 12/10/ 2010, Viewed on 1/8/2011, http://www.botjunkie.com/2010/10/12/googles-autonomous-car-takes-to-the-streets/

[11] Battery Cages Banned in Europe. Last updated on 27/3/2012, Viewed on 1/1/2014, http://www.animalsaustralia.org/features/eu-bans-battery-hen-cages.php

[12] Compassion in world farming. Viewed on 1/1/2014, http://ciwf.org.uk/the_big_move/about_the_ban.aspx

[13] R. E. S. Thomas A. Carter, "Drinking water quality for poultry," *Poult. Sci. Technol. Guid.* The North Carolina Agricultural Extension Service

[14] SPlAS: University of Waterloo - Research » Research excellence » Research success stories. Viewed on 14/4/2014 https://uwaterloo.ca/research/research-excellence/research-success-stories/new-technology-will-change-water-safety

[15] Feeding Chickens for best health and performance : The Poultry Site. Last Updated on 10/1/2003, Viewed on 12/3/2014, http://www.thepoultrysite.com/articles/94/feeding-chickens-for-best-health-and-performance

[16] Š. Mihina, M. Sauter, Z. Palkovičová, I. Karandušovská, and J. Brouček, "Concentration of harmful gases in poultry and pig houses *," vol. 30, no. 4, pp. 395–406, 2012.

[17] Wikipedia: Electronic Nose, Last updated on 29/1/2014, Viewed on 22/3/2014, http://en.wikipedia.org/wiki/Electronic_nose#Working_principle

[18] IR sensors, Last Updated on 30/10/2011, Last Seen 16/4/2014 http://www.icrobotics.co.uk/wiki/index.php/IR_Sensors

[19] Vision Supplies. Sick LMS151-10100, Last Seen 16/4/2014, http://vision-supplies.com/p/76909/sick-lms151-10100

[20] H. Choset and P. Pignon, "Coverage Path Planning : The Boustrophedon Cellular Decomposition," *F. Serv. Robot.*, pp. 203–209, 1998.

[21] H. Choset, "Coverage for robotics - A survey of recent results," *Ann. Math. Artif. Intell.*, vol. 31, pp. 113–126, 2001.

[22] D. MacKenzie and T. Balch, Making a clean sweep: Behavior based vacuuming, in: AAAI Fall Symposium, Instationating Real-World Agents (1996).

[23] T. Balch and R.C. Arkin, Communication in reactive multiagent robotic systems, Autonom. Robots 1(1) (1995).

[24] T. Balch, The case for randomized search, in: *Workshop on Sensors and Motion, IEEE International Conference on Robotics and Automation*, San Francisco, CA (May 2000).

[25] D. Gage, Randomized search strategies with imperfect sensors, in: Proc. SPIE Mobile Robots VIII, Boston, MA (September 1993) pp. 270–279.

[26] I. S. T, D. Sistemas, and A. V. R. Pais, "Complete Coverage Path Planning and Guidance for Cleaning Robots," pp. 677–682.

[27] S. Hert, S. Tiwari and V. Lumelsky, A terrain-covering algorithm for an AUV, Autonom. Robots 3 (1996) 91–119.

[28] H. Choset and P. Pignon, "Coverage Path Planning : The Boustrophedon Cellular Decomposition," *F. Serv. Robot.*, pp. 203–209, 1998.

[29] E.Acar and H. Choset, Critical point sensing in unknown environments, *IEEE International Conference on Robotics and Automation*, San Francisco, CA (2000).

[30] Y. Gabriely and E. Rimon, “Spanning-tree based coverage of continuous areas by a mobile robot,” *Proc. 2001 ICRA. IEEE Int. Conf. Robot. Autom. (Cat. No.01CH37164)*, vol. 2, 2001.

[31] W. H. Huang, “Optimal line-sweep-based decompositions for coverage algorithms,” *Proc. 2001 ICRA. IEEE Int. Conf. Robot. Autom. (Cat. No.01CH37164)*, vol. 1, 2001.

[32] R. C. Prim Shortest connection networks and some generalizations, *Bell System Tech J*. 36:1389-1401,1957.

[33] Hart, P. E.; Nilsson, N. J.; Raphael, B. (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics SSC4*, Vol. 4, no. 2: 100–107

[34] Ntawumenyikizaba, “An Online Complete Coverage Algorithm for Cleaning Robots based on Boustrophedon Motions and A * search,” pp. 401–405.

[35] R. H. Esther M. Arkin, “Approximation Algorithms For The Geometric Covering Salesman Problem.”

[36] H. Choset, “Coverage for robotics - A survey of recent results,” *Ann. Math. Artif. Intell.*, vol. 31, pp. 113–126, 2001.

[37] Y. Gabriely and E. Rimon, “Spanning-tree based coverage of continuous areas by a mobile robot,” *Proc. 2001 ICRA. IEEE Int. Conf. Robot. Autom. (Cat. No.01CH37164)*, vol. 2, 2001.

[38] H. Choset and P. Pignon, “Coverage Path Planning: The Boustrophedon Cellular Decomposition” *International Conference on Field and Service Robotics.*, pp. 203–209, 1998.

[39] A. Ntawumenyikizaba, “An Online Complete Coverage Algorithm for Cleaning Robots based on Boustrophedon Motions and A * search,” pp. 401–405.

[40] Jesse Jackson and Jerry Burton, *Encoder-Newsletter of the Seattle robotic society*, Viewed on 284/2014, http://www.seattlerobotics.org/encoder/may97/sonar2.html

[41] P. J. McKerrow, “Echolocation—from range to outline segments,” *Robot. Auton. Syst.*, vol. 11, no. 4, pp. 205–211, 1993

[42] H. Choset, K. Nagatani, and N. a. Lazar, “The arc-transversal median algorithm: a geometric approach to increasing ultrasonic sensor azimuth accuracy,” *IEEE Trans. Robot. Autom.*, vol. 19, no. 3, pp. 513–522, Jun. 2003.

[43] New Mechanical Milkers, *Popular Mechanics*, pp. 868, Aug 1907.

[44] Trevelyan, J.P., "Sensing and control for sheep shearing robots," *Robotics and Automation, IEEE Transactions on* , vol.5, no.6, pp.716,727, Dec 1989

[45] Lely Discovery Introductory page, Last Seen 1/5/2014, http://www.lely.com/en/home/media-centre/news-en-events/news/the-newly-introduced-lely-discovery-90sw-barn-cleaner-features-a-wet-s

[46] Michael Krystolovich and Neil Anderson, "OMAFRA Articles and Updates: Livestock", *ON Organic Newsletter*

[47] Sanders, K.F., "Orange Harvesting Systems Review", Biosystems Engineering, vol.90, no.2, 2005, pp.115-125.

[48] Brown, C.E.S.a.G.K., "Basic considerations in mechanizing citrus harvest", Transaction of the ASAE, 1968, pp.343-346.

[49] D'Esnon, A.G., "Robotic Harvesting of Apples", *ASAE; SME*, no.01-85, 1985, 210-214.

[50] A. Grand D'Esnon, G.R., R. Pellenc, A. Journeau, M.J. Aldon, "Magali: a Self-Propelled Robot to Pick Apples", *American Society of Agricultural Engineers*, vol.87, no.1037, 1987.

[51] Tillett, N.D., "Robotic Manipulators in Horticulture: A Review", *J.agric. Engng Res.*, vol.55, 1993, pp.89-105.

[52] G. Rabatel, A.B., F. Sevila, F. Juste, "*Robotic Harvesting of Citrus: State-Of-Art and Development of the French Spanish EUREKA Project*, 1995.

[53] F. Juste, F.S., "Citrus: An European Project to Study the Robotic Harvesting of Oranges", *Proceedings of the 3. International Symposium on: Fruit, Nut, and Vegetable Harvesting Mechanization Jordbrugsteknik Agricultural Engineering*, no.67, 1992, 331-338.

[54] Tillett & Hague Technology website, Viewed on 2/5/2014, http://www.thtechnology.co.uk/applications.html

[55] Vision Robotics Technology website, Viewed on 2/5/2014, http://www.visionrobotics.com

[56] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: MIT Press, 2005, pp. 17–38.

[57] M. A. Batalin and G. S. Sukhatme, "Spreading Out : A Local Approach to Multi-robot Coverage", *Distributed Autonomous Robotic Systems* ,pp. 373–382, 2002.

[58] I. A.Wagner and A. M. Bruckstein, "Cooperative Cleaners – A Study in Ant-Robotics in Communications, Computation", *Control and Signal Processing: A Tribute to Thomas Kailath (*KluwerAcademic) pp. 289–308, 1997

[59] I.A.Wagner, M. Lindenbaum and A.M. Bruckstein, “Smell as a computational resource – a lesson we can learn from the ant”, *ISTCS’96* pp. 219–230, 1996.

[60] I.A. Wagner, M.Lindenbaum and A.M. Bruckstein, “Distributed covering by ant-robots using evaporating traces”, *IEEE Trans. Robotics Autom.* vol 15, no. 5, pp.918–933, 1999

[61] I.A.Wagner, M.Lindenbaum and A.M. Bruckstein, “Efficiently searching a dynamic graph by a smell- oriented vertex process”, *Ann. Math. Artif. Intell.* vol 24, no. 1-4, pp. 211–223, 1998.

[62] D. Kurabayashi, J. Ota, T. Arai and E. Yoshida, “Cooperative sweeping by multiple mobile robots”, *Int. Conf. on Robotics and Automation*, 1996.

[63] I. Rekleitis, G. Dudek and E. Milios, “Multi-robot exploration of an unknown environment, efficiently reducing the odometry error”, *IJCAI, International Joint Conference in Artificial Intelli- gence*, Vol. 2, Nagoya, Japan, pp. 1340– 1345, August 1997.

[64] Z.J. Butler, A.A. Rizzi and R.L. Hollis, “Complete distributed coverage of rectilinear environments”, *Proc. of the Workshop on the Algorithmic Foundations of Robotics* , March 2000.

[65] A. Zelinsky, R.A. Jarvis, J.C. Byrne and S. Yuta, “Planning paths of complete coverage of an unstructured environment by a mobile robot”, *Proceedings of International Conference on Advanced Robotics*, pp. 533–538, November 1993

[66] Wikipedia: Remotely operated underwater vehicle wiki, Last updated on 14/5/2014, Viewed on 14/5/2014, http://en.wikipedia.org/wiki/A*

[67] Cormen, Thomas H.; Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford , "Section 24.3: Dijkstra's algorithm", *Introduction to Algorithms (Second ed.)*, MIT Press and McGraw–Hill, 2001, pp. 595–601.

[68] Mars Rover Home Page, Viewed on 14/7/2011, http://marsrover.nasa.gov/newsroom/pressreleases/20140116a.html

[69] Yang, X.-S. *Nature-Inspired Metaheuristic Algorithms*. Luniver Press, 2008.

[70] Hanyong Choi; Ohmori, S.; Yoshimoto, K.; Ohtake, H., "Improvement of Particle Swarm Optimization: Application of the mutation concept for the escape from local minima", *Supply Chain Management and Information Systems (SCMIS), 2010 8th International Conference on* , vol., no., pp.1,5, 6-9 Oct. 2010

[71] J. Borenstein,” Internal Correction of Dead-reckoning Errors With the compliant Linkage Vehicle”, *Journal of Robotic Systems*, Vol. 12, No. 4, April 1995, pp. 257-273.

[72] N. Bulusu, J. Heidemann and D. Estrin, "GPS-Less Low-Cost Outdoor Localization for Very Small Device", *Special Issue on Smart Spaces and Environments, IEEE Personal Comm.*, Oct. 2000, pp. 28-34.

[73] Chuck Gilbert , "How is the accuracy of a GPS receiver described", Last Updated on 02/1/2003, viewed on 14/7/2011, http://www.romdas.com/technical/gps/gps-acc.htm

[74] Martinelli, A. ; Siegwart, R, "Estimating the Odometry Error of a Mobile Robot during Navigation", *ECMR03 Conference*, 2003

[75] D.C. Conner, P.R. Kedrowski, C.F. Reinholtz, and J.S. Bay, "Improved dead reckoning using caster wheel sensing on a differentially steered 3-wheeled autonomous vehicle" , *proceedings of the SPIE Mobile Robots XV conference*, November 2000.

[76] Kwang-Min Jung and Kwee-Bo Sim, "Development of Advanced Particle Filter for Localization of mobile robot", *International Conference on Control. Automation and Systems*, Oct. 27-30, 2010. pp 1573-1575

[77] Siegwart, R. and I.R. Nourbakhsh, "Introduction to autonomous mobile robots", *Intelligent robots and autonomous agents*, 2004, Cambridge, Mass.: MIT Press, pp 1-45

[78] Lucas, G.W., "A Tutorial and Elementary Trajectory Model for the Differential Steering System of Robot Wheel Actuators", Viewed on 14/7/2011, http://rossum.sourceforge.net/papers/DiffSteer/DiffSteer.html

[79] Charles Smart, Marc Zinck, "Gyro-Encoder Hybrid Localization", Last Updated on 29/5/2001, Viewed on 14/7/2011, http://www.frc.ri.cmu.edu/projects/colony/pdf_docs/gyropaper.pdf

[80] N. Rao, S. Iyengar, C. Jorgensen, and C. Weisbin, "On terrain acquisition by a finite-sized mobile robot in plane," *Proc. 1987 IEEE Int. Con5 on Robotics and Automation*, Raleigh, NC, Apr. 1987, pp. 1314-1319.

[81] RAO, N.S.V., S.S. IYENGAR, J.B. OOMMEN, and R.L.KASHYAP, "Terrain acquisition by a point robot amidst polyhedral obstacles", *Proc. 3rd IEEE Conf. on AI Applications*, Orlando, F1, Feb. 1986.

[82] Howie Choset with slides from G.D. Hager and Z. Dodds, "Robotic Motion Planning : What ' s Special About Bugs", Viewed on 1/2/2012, http://www.cs.cmu.edu/~motionplanning/lecture/Chap2-Bug-Alg_howie.pdf

[83] V. J. Lumelsky and A. A. Stepanov, “Dynamic path planning for a mobile automaton with limited information on the environment,” *Autom. Control. IEEE Trans.*, vol. 31, no. 11, pp. 1058–1063, Nov. 1986.

[84] Wikipedia: Mobile Robots, Last updated 29/6/2011, Viewed on 4/8/2011, http://en.wikipedia.org/wiki/Mobile_robot

[85] V. Lumelsky and A. Stepanov, “Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape,” *Algorithmica*, vol. 2, no. 1–4, pp. 403–430, 1987.

[86] I. Kamon, E. Rimon, and E. Rivlin, “TangentBug: A Range-Sensor-Based Navigation Algorithm,” *Int. J. Rob. Res.*, vol. 17, no. 9, pp. 934–953, Sep. 1998.

[87] T. S. Lumelsky, VladimirJ., “Incorporating Range Sensing in the Robot,” *IEEE Trans. Syst. , man Cybern.*, vol. 20, no. 5, pp. 1058–1069, 1990.

[88] A. Wickramasooriya, G. Hamilan, L. S. I. Jayawardena, W. M. D. Wijemanne, and S. R. Munasinghe, “Characteristics of Sonar Range Sensor SRF05,” *2008 4th Int. Conf. Inf. Autom. Sustain.*, pp. 475–480, 2008.

[89] Specification, “SRF05 - Ultra-Sonic Ranger,” pp. 1–4, 2008.

[90] J. hyun J. * and M. cheol L. ** Min g y d Park *, “OBSTACLE AVOIDANCE FOR MOBILE ROBOTS USING ARTIFICIAL POTENTIAL FIELD APPROACH WITH SIMULATED ANNEALING,” *Eng. Intell. Mech.*, pp. 1530–1535, 2001.

[91] R. C. Eberhart, “Engineering optimization with particle swarm,” *Proc. 2003 IEEE Swarm Intell. Symp. SIS’03 (Cat. No.03EX706)*, pp. 53–57.

[92] C. Yang, S.O. Parsher, J.A. Landry, H.S. Ramaswami and A. Ditommaso, “Application of artificial neural networks in image recognition and classification of crop and weeds”, *Journal of Canadian Agricultural Engineering*. 2003.

[93] Farzin Mokhtarian , Alan K. Mackworth, “A Theory of Multiscale, Curvature-Based Shape Representation for Planar Curves”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v.14 n.8, p.789-805, August 1992

[94] Lowe, David G. "Object recognition from local scale-invariant features", *Proceedings of the International Conference on Computer Vision*, pp. 1150–1157, 1998.

[95] Chris M. Christoudias, Bogdan Georgescu, “Code for the Edge Detection and Image Segmentation system”, Last Updated on 5/5/2009, Viewed on 14/7/2011, http://coewww.rutgers.edu/riul/research/code/EDISON/index.html

[96] Matlab Central Team, “Open implementation of the SIFT detector and description”, Last Updated on 19 /7/2009, Viewed on 14/7/2011, http://www.mathworks.com/matlabcentral/linkexchange/links/1408-open-implementation-of-the-sift-detector-and-description

[97] Sparkfun product page, Viewed on 30/5/2014, Updated on 30/5/2014, https://www.sparkfun.com/categories/160

[98] Akihisa Ohya, Akio Kosaka, and Avinash Kak, “Vision-Based Navigation by a Mobile Robot with Obstacle Avoidance Using Single-Camera Vision and Ultrasonic Sensing”, *IEEE Trans. Robotics Autom.* vol 14, no. 6,1998

[99] Liu, H., and Arimoto, S., “Path planning using a tangent graph for mobile robots among polygonal and curved obstacles”, *Int. J. Robot. Res.*, vol 11, no 4, pp. 376-382, 1992

[100] Technical Specification, “BrickCom OB-100A Outdoor bullet camera”

[101] SRF235 - Pencil Beam Ultrasonic Range Finder – Product page, Viewed on 30/5/2014, http://en.manu-systems.com/SRF235.shtml

[102] Tangents through an external point, Viewed on 30/12/2014, http://www.mathopenref.com/consttangents.html

[103] Y. Sarig, “Robotics of Fruit Harvesting: A State-of-the-art Review,” *J. Agric. Eng. Res.*, vol. 54, no. 4, pp. 265–280, Apr. 1993.

[104] Crest Manufacturing Industries, *Official Website*, Viewed on 14/7/2011, http://www.crestmi.com.au/

[105] Sabertooth 2×12 product page, *Official Website*, Viewed on 1/1/2015, https://www.dimensionengineering.com/products/sabertooth2x12

[106] eLabD50, *PIC Development Board CoreChartTM Workshop (PIC16F877A)*

[107] Foody, G.M.; Ajay Mathur, "A relative evaluation of multiclass image classification by support vector machines," *Geoscience and Remote Sensing, IEEE Transactions on* , vol.42, no.6, pp.1335,1343, June 2004

[108] Wikipedia: Graph traversal, Last updated on 18/1/2015, Viewed on 18/2/2015, *http://en.wikipedia.org/wiki/Graph_traversal*

[109] Tim Wescott, *PID without a PHD*, viewed 14/7/2011, http://igor.chudov.com/manuals/Servo-Tuning/PID-without-a-PhD.pdf

[110] Wikipedia: Ray casting, Last updated on 25/12/2014, Viewed on 18/2/2015, http://en.wikipedia.org/wiki/Ray_casting