

THE APPLICATION OF DISCRETE-EVENT SIMULATION
FOR DEMINING STRATEGY EVALUATION

by

HUI-CHIAO JEN

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2008

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my advisor, Dr. Brian Huff, for his keen insight of the research topic, his technical advice, and his friendly and patient assistant to me in all aspects of this dissertation. I also would like to thank Dr. Don Liles for providing financial support and, as a committee member, guidelines for this research. Thank you to my other committee members- Dr.Rogers, Dr.Kim and Dr.Dogan for their precious comments and advice through the dissertation process. Thanks to my dear friends Joye Homles and Libby Leatherman for undertaking the arduous task of reading and revising for this dissertation, and their superior encouragement. I would like to thank Rory Cannaday who initially coded an independent scout vehicle model on which this research was able to make further development. I want to thank all that I have mentioned.

Finally, I would like to thank all of my friends for their encouragement, advise, and listening to me about this work. Most of all, I would like to express my gratitude to my parents, my aunts, my brother, my sister in law, and my dear lover, Li-Chieh Hsiao, for providing me the support, understanding and patience that I needed. I could not have done this work without them.

April 17, 2008

ABSTRACT

THE APPLICATION OF DISCRETE-EVENT SIMULATION FOR DEMINING STRATEGY EVALUATION

Hui-Chiao Jen, PhD.

The University of Texas at Arlington, 2008

Supervising Professor: Brian Huff

Teams of mobile autonomous robots have been proposed for the detection and clearing of landmines. Several competing strategies for automated mine detection have been proposed. This paper will discuss the development of a tool that will support the quantitative analysis of various automated mine detection approaches used in Humanitarian Demining and Military Mine Field Breaching. The approaches used to model mine detection in a scout, breaching, and Humanitarian Demining scenario using a discrete-event simulation analysis tool are discussed. The model assumes that each UGV is autonomous rather than being directly controlled by a centralized command and

control system. The algorithm used to drive a single UGV along a simulated breach path is presented. An alternative algorithm set is also proposed for the control of a coordinated team of UGV's. The extension of the tool to support humanitarian mine clearing and the use of multiple waves of automated mine detectors are also presented. A simulation system constructed on the WITNESS® discrete-event simulation package will be shown. This paper simulates the motion of multiple robots through a mock minefield containing mines and obstacles that limit the motion of the autonomous vehicles.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS.....	ii
ABSTRACT	iii
LIST OF ILLUSTRATIONS.....	viii
LIST OF TABLES.....	xii
Chapter	Page
1. INTRODUCTION	1
1.1 Background and Motivation	1
1.2 Research Objective	3
1.3 Research Approach.....	4
2. LITERATURE REVIEW	10
2.1 Landmine Detection Technologies	10
2.1.1 Sensor Technologies	11
2.1.2 Robot Technologies	13
2.2 Mobile Robot Team Concepts	14
2.2.1 Robotic Behavior Control	14
2.2.2 Searching Applications.....	19
2.3 Vehicle Routing Problem (VRP).....	24
3. MiCAT DESIGN CONCEPT	27

3.1 The Simulated Minefield	27
3.2 The Simulated Mine Detection Vehicle	29
4. MiCAT APPLICATION SCENARIOS	35
4.1 Simulation of a Scout Vehicle	35
4.2 Simulation of a Coordinated Team of Mine Detection Vehicles	43
4.2.1 Military minefield breaching (MMB).....	44
4.2.2 Humanitarian Minefield Reclamation (HMR).....	50
4.3 Point-to-Point Wave	62
5. DISCUSSION OF MODEL GENERALIZATION	69
5.1 Adjustable Grid Scaling and Shifting The Orientation of The Search Grid	69
5.2 The Modeling of Vehicles of Different Sizes.....	75
5.3 Alternative Vehicle Formations.....	77
6. ANALYSIS OF EXPERIMENTAL RESULTS AND DISSCUSSION.....	83
6.1 Examples of MiCAT Use Scenarios.....	86
6.1.1 Analysis of Mine Detection Sensor Effectiveness.....	86
6.1.2 The Impact of Minefield Complexity on Search Performance	89
6.1.3 Impact of the Number of Mine Detection Vehicles Used on Search Time	91
6.2 The Use of MiCAT for HMR and MMB System Design	92
6.3 The Use of MiCAT for the Comparison of HMR and MMB Strategies and Technologies.....	94
6.3.1 Evaluation of Various Point-to-Point (P2P) Wave Deployment Strategies for HMR Operations	94

6.3.2 Evaluation of Various Point-to-Point (P2P) Wave Deployment Strategies for MMB Operations.....	96
6.3.3 Evaluation of the Deployment of a Single Wave of High Capable Vehicle or Using Multiple Waves of Heterogeneous Vehicles	97
6.4 The Use of MiCAT to verify HMR and MMB Simulation Algorithm	98
6.4.1 The Development of Multi-Vehicle Control Algorithm.....	98
6.4.2 The Development of Vehicle Re-Tasking Algorithm Response to Vehicle Loss.....	99
7. CONCLUSIONS AND FUTURE WORK.....	101
7.1 Conclusion	101
7.2 Future Work.....	105
REFERENCES	108
BIOGRAPHICAL INFORMATION.....	113

LIST OF ILLUSTRATIONS

Figure	Page
1.1 Scout Wave (Scout sweeps within the pre-plan path).....	8
1.2 Boustrophedon path.....	8
1.3 Example of Complete Coverage: The number of lanes is divisible by the numbers of vehicles	8
1.4 Example of Complete Coverage: The number of lanes is not divisible by the numbers of vehicles.....	9
1.5 Coordinated Line wave for Breaching: The team will redefine the path to avoid hazard	9
1.6 Point-to-Point Wave: The vehicle moves directly to locations identified by the previous wave	9
2.1 Formation for four robots(a) line, (b) column, (c) diamond, and (d) wedge (Balch and Arkin 1998).....	16
2.2 Zone for the computation of maintain-formation magnitude (Balch and Arkin 1998; Balch and Hybinette 2000).....	17
2.3 The staggered formation of a minesweeping team (Healey 2001; Ludwig 2000)	20
3.1 The Simulated Minefield.....	27
4.1 Breach lane identifying procedure	43
4.2 Simulation of a Scout Vehicle searching for a breach lane (A) three units wide and (B) five units.....	43
4.3 Breach lane identifying procedure of a coordinated team.....	49
4.4 The three vehicle breaching team runs first providing 100% coverage for the breach lane.....	50

4.5	A Turning Point that causes the HMR search area to be decreased.....	51
4.6	An example of HMR's Lane Assignment for four vehicles.....	52
4.7	A Task Assignment procedures for HMR search.....	53
4.8	A HMR search procedure.....	61
4.9	(A) A HMR overlap strategy improves the percentage of the area (B) The survived vehicle supports the un-finished area that is left by the destroyed vehicle.....	62
4.10	(A) The three vehicle breaching team runs first providing 100% coverage for the breach lane, (B) a two-vehicle P2P team starts to work right after the Multi-Vehicle Wave begins its search.	62
4.11	The P2P search flowchart.....	68
5.1	A ZigZag shape minefield area	70
5.2	An illustration of the shifting of minefield orientation	70
5.3	A steep slop produces a sliding displacement or a fracture	74
5.4	The required turning around radius from one piece to another piece of the minefield.....	75
5.5	Two independent vehicles of different sizes (A) three units size and (B) seven units size	77
5.6	A staggered vehicle formation	77
5.7	Setting a virtual obstacle	79
5.8	The vehicle team moves in the same direction to hold a breaching path.....	81
5.9	The vehicle team separates while meeting a landmine	82
5.10	The time control algorithm has made the vehicle able to move back while meeting a landmine.....	82
5.11	The time control algorithm has made the vehicle able to go around the virtual landmine	82

6.1	MiCAT selection interface	84
6.2	Parameters input interface.....	84
6.3	Obstacle (or landmine) Locations Input Interface in Excel	85
6.4	The intended simulated minefield search scenario is loaded.....	85
6.5	Data summary report.....	85
6.6	The search time increases when using lower mine detection sensor	88
6.7	The amount of total steps taken is slight difference.....	88
6.8	Simulation running results at detection probabilities of (A)100%(B) 75%.	89
6.9	Simulation running results of (A) total steps (B) time taken at the different types of mine sensor.	90
6.10	Simulation running results of (A) total steps (B) time taken at the different types of minefield.	90
6.11	Simulation running results for each vehicle when using different types of mine sensor within (A) High (B) Medium complexity minefield.	91
6.12	(A) the amount of total search steps (B) total search time taken, when using different number of vehicles within different type of simulated minefield.....	92
6.13	The trend line of the time taken	93
6.14	(A) The amount of time (B) steps taken, between different multi-wave strategies.	96
6.15	(A) The amount of time (B) steps taken, between different multi-wave strategies of MMB scenario.....	97
6.16	The search time comparison between a single and multiple waves.....	98
6.17	The Virtual (A) and Non- Virtual obstacle (B) strategy	99
6.18	The complete coverage search task at detection probability of 65% (A) Non- Support (B) Reassignment Support.....	100

6.19 The complete coverage search task at detection probability of 60%
(A) Non- Support (B) Reassignment Support..... 100

LIST OF TABLES

Table	Page
3.1 Obstacles Generation Algorithm.....	29
3.2 Eight Moving Directions Of The Vehicle	30
3.3 Obstacle and Mine Detection Algorithm	33
4.1 Scout Scanning Minefield Algorithm	38
4.2 Scout Searching Desired Path Algorithm	39
4.3 Keeping The Scout Stay On Current Route Algorithm.....	41
4.4 Scout Bypass Landmine Algorithms.....	42
4.5 Algorithms To Search The Breaching Lane.....	46
4.6 Algorithms To Stay On The Path For Breaching Team.....	48
4.7 Algorithms To Get HMR's First Lane Assignment.....	54
4.8 Algorithms To Get HMR's Lane Assignments.....	54
4.9 Algorithms To Get Lane Assignments For Each HMR Vehicle.....	55
4.10 HMR Location Relationship	57
4.11 HMR Searching Patterns	57
4.12 Algorithm To Determine A Search Pattern.....	59
4.13 Algorithm To Apply A HMR Search Pattern	59
4.14 HMR Search Algorithms.....	60
4.15 HMR Reassign Search Task Algorithms	60

4.16	Algorithms To Allocate The Potential Landmines	65
4.17	Algorithms To Distribute The Potential Landmines To P2P Vehicles	65
4.18	Point To Point Searching Patterns.....	67
4.19	Algorithms To Obtain A P2P Searching Patterns	67
4.20	Algorithms To Obtain P2P Next Step.....	67
5.1	Algorithms To Reset The Minefield	73
5.2	Algorithms To Resize The Vehicle.....	76
5.3	Algorithms To Search The Entire Grids In Front of A Vehicle.....	76
5.4	Vehicle Movement Control Algorithm	78
5.5	Vehicle Moving Direction Control Algorithm.....	80
6.1	The Amount of Searches and Time taken at Different Probabilities of Mine Sensor Detection.....	87
6.2	The Running Results Comparison at Different Probabilities of Mine Detection Sensor.....	87

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

Clearing minefields is a challenging task to manage. The exorbitant costs, substantial labor, and considerable time it takes to administer removal applications makes this a strenuous project. According to Foreign Affairs Canada (2005) land mines are cheap and easy to use, but are expensive to remove. It costs between \$300 and \$1,000 to safely remove just one mine, because of the skills and equipment required. Additionally, the clearing speed is always slower than new mines being placed. There are 10 new mines being placed for every one mine that is successfully cleared (Gooneratne et al. 2004).

The most affected areas are Asia and Africa. Uncleared landmines pose a huge problem for their economies, destroy their land usage and are dangerous to people who must pass through the landmine explosion environment. Humanitarian organizations are active in helping these countries in clearing landmines, and have established the humanitarian minefield reclamation (HMR) application for the complete removal of all types of mines. HMR operations are typically conducted after military conflict has ceased. The speed with which the mines are detected and removed is not of primary importance. Critical success factors for HMR applications are the complete removal of

all mines, the safety of the demining crews, and the minimization of the overall cost of the demining efforts (Habib 2001).

Another demining application often used is military minefield breaching (MMB). The objectives and philosophy of military demining are different from the HMR application. In MMB applications, the primary objective is speed. Using this method, casualties from mines and other weapons are expected. The military accept the high risk that some of their vehicles and soldiers will still be destroyed and killed during the breaching task. Tactical countermining is comprised of operations that allow an attacking force to penetrate the minefield or avoid mines as it attacks a target or clears an area speedily to sustain specific operations. The primary concern of MMB application is time. A mine clearance or avoidance rate of 80% is generally considered acceptable. (Habib 2001; Rajasekharan and Kambhampati 2003).

As the need for detecting and removing landmines has increased, several researchers proposed the use of robotic systems to support demining operations (Rajasekharan and Kambhampati 2003). Debenest et al. (2003) recommend the automated buggy as a mine detection vehicle for several reasons. This small remotely controlled vehicle has small dimensions and can be used on uneven terrain, narrow roads, inside irrigation channels and urban areas. Also steering, gear changing, accelerating and braking functions can be tele-operated by the controllers at a safe distance from the minefield. Compared with other choices, automated buggy costs are also relatively low (Debenest et al. 2003). The Autonomous Vehicles Laboratory (AVL) at The University of Texas at Arlington (UTA) currently is also developing low-cost,

man-portable autonomous/unmanned ground vehicles (AGV) that are capable of supporting HMR and MMB mine detection applications. Many other organizations are also exploring the use of unmanned or autonomous vehicles for mine detection applications. It is highly unlikely that a universal autonomous mine detection system can be developed for the wide variety and diversity of potential demining environments. As a result, there is a need for a diverse set of technologies and strategies that can be developed to support automated mine detection. This research attempts to develop a tool that can be used to evaluate the merits of using unmanned or autonomous vehicles to match potential mine detection systems and methods to specific demining environments.

1.2 Research Objective

The primary objective of this research is to explore the feasibility of creating a tool that will support the analysis and comparison of various mine detection strategies and technologies. To answer this larger question, we must in turn determine if it is possible to represent or model the minefield environment and processes associated with humanitarian mine detection and removal activities or military minefield breaching applications. These demining processes must be captured or codified within a series of algorithms that describe the behaviors of the various elements within a landmine detection and removal scenario. Once created, these algorithms must, in turn, be embedded within the framework of a generalized analysis tool. The proposed analysis tool must then be populated with information and data that represents the mine detection scenario being studied.

1.3 Research Approach

This paper presents the early effort of the development of a Mine Clearing Analysis Tool (MiCAT) that is designed to support the analysis of various automated mine detection strategies and technologies on both HMR and MMB applications. It is designed to predict the speed, cost, and effectiveness of various automated mine clearing technologies and practices. An immediate use of this type of tool would be to help automated mine clearing technology developers explore and obtain system performance estimates. These estimates would cover a wide variety of system designs early in the design phase in order to identify efficient and cost effective mine clearing solutions. After much development and refinement, this tool can be used as a planning tool for tactical minefield clearing or breaching missions.

A simulator has been used for several decades to analyze and evaluate conceptual ideals before they are implemented into practical applications. The proposed MiCAT will be developed on a Discrete-Event Simulation package named WITNESS®. WITNESS® is a commercial simulation package and has been successfully used to simulate many manufacturing environments. With its high-level construction, flexibility, and graphical representation of independent entity flows between processes, this research can focus on constructing a minefield simulation environment rather than developing a robust simulation engine. WITNESS® also provides a full suite of statistical distribution tools to support multiple simulation experiments, and diversity of system performance reporting tools. We believe that using a Commercial-Off-The-Shelf (COTS) simulation development environment will stream line the transfer of our work

to others in the demining community interested in using or expanding the capabilities of the MiCAT.

Unlike conventional industrial automation that works in a structured environment, autonomous mine detection vehicles have to work within an unstructured and frequently hostile environment. Therefore, MiCAT minefield area is decomposed to a series of lanes with an arbitrary array of cells in each lane based on a geometric structure cellular decomposition (Acar et al. 2003). The cellular decomposition represents a union of non-overlapping adjacency subregions (Acar et al. 2003), each region is a cell (Choset. and Pignon 1997). In MiCAT, each cell represents (X, Y) , the smallest patch of minefield, measured according to the size of the military vehicles. Thus, the simulated vehicle can move from one cell to another. This problem for the coverage within the polygonal and trapezoid environments has been discussed in many papers (Acar et al. 2003; Choset. and Pignon 1997; Hu and Brady 1997; Huang 2000; Kruusmaa 2003; Wong and MacDonald 2003). To limit the scope of this work and concentrate on the analysis of strategies and technologies for the demining applications, we will not explore the decomposition problem.

With MiCAT, we are attempting to investigate the relative merits of using multi-sensor platforms capable of performing the mine detection task in a single pass or using multiple waves of smaller, simpler and less expensive autonomous mine detection vehicles. The multiple-wave scenario means using the information received from the previous wave (e.g. environment information or potential mine locations) for the next wave of searching. In this research there are three general classes of waves: the scout

wave, the coordinated line wave, and the point-to-point wave within the multi-wave scenario. Each wave will be applied one or more times in different mine detection scenarios.

Most of the time the scouts are used in the military reconnaissance and surveillance operations of hostage and survivor rescue missions, illicit drug raids, and responses to chemical or toxic waste spills (Rybski et al. 2000). With the miniature video camera the scout units have the ability to view and scan obstacles within a certain distance. They can also carry sensors to propel themselves away from danger. The scout wave in this research is designed to provide the MMB engineer with a means of inspecting, and if necessary, re-routing the planned path from a location of relative safety (see figure 1. 1).

HMR and MMB applications in this research will apply the different searching algorithms (Rajasekharan and Kambhampati 2003). The objective of the HMR application is the complete removal of all the mines and, thereby recovering the minefield. In order to complete landmine search, a Boustrophedon moving procedure (figure 1.2) is used in the coordinated line of wave of HMR scenario. Boustrophedon path is to make the vehicles able to move along the full length of the minefield in a straight line, turn around, and then traces a new straight line path next to the previous one continuously to complete the search (Choset. and Pignon 1997). The HMR scenario would consist of multiple autonomous mine detection vehicles. The simulated minefield contains a series of lanes. Each vehicle is assigned to a predefined area. A point of concern is how to divide the tasks among the vehicles equally. It is easy to equally

divide the tasks if the number of lanes is divisible by the number of vehicles (e.g. figure 1.3). If not (e.g. figure 1.4), it has to find a better way to share the mine detection tasks.

The goal of MMB application is to clear a narrow path within a minefield to let the military move troops and equipment within this path quietly and quickly. The military does not care about the clearance outside this path. They are only concerned with the clearance inside this narrow path and whether it is wide enough to hold the military troop. The MMB coordinated line wave will utilize a coordinated team of expendable vehicles carrying fast and simple sensors (e.g. conventional metal detectors) to move in formation and to ensure complete coverage of the planned breach path (see figure 1.5). The path width would be dictated by the size of military vehicle the breach path is designed to support. To ensure the search vehicles can completely cover the planned breach path, the reconnaissance result from the previous scout wave can provide the MMB engineer with several paths to choose from rather than only one that fits the exact size of the military vehicle team. Thus, if a mine or obstacle is detected, the vehicle team will have room to attempt to re-define a breach path that will avoid the hazard (see figure 1.5). If the width of the breaching path is not wide enough to keep the vehicle team formation while avoiding the mine or obstacle then the mine will be identified for removal or destruction by subsequent MMB resources.

Once a potential mine location has been identified, it might be beneficial to re-test that location with an alternative mine detection technology. This re-testing can be performed by another set of vehicles referred to as a point-to-point (P2P) wave. One or more vehicles within a point-to-point wave would not attempt to re-search the entire

area but would simply move directly to the high probability mine locations identified by previous waves to either verify or deny the presence of a mine (figure 1.6). A point-to-point wave would also be applicable for resources used to remove or destroy mines detected by other waves.

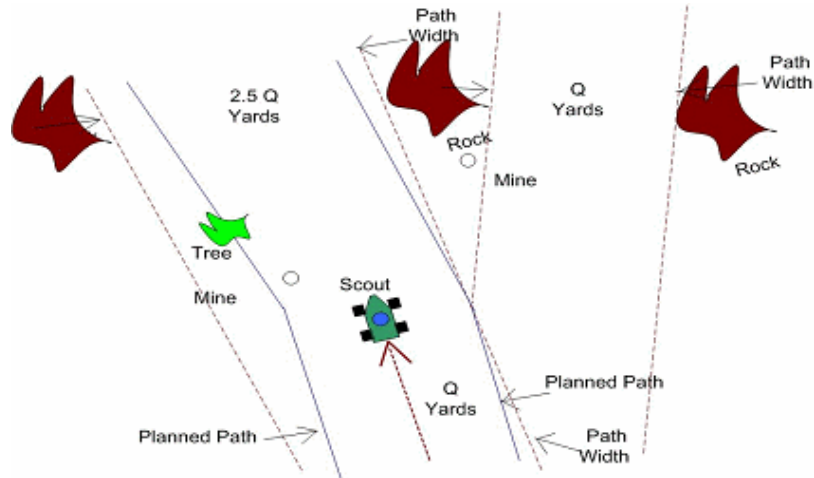


Figure 1.1 Scout Wave (Scout sweeps within the pre-plan path)

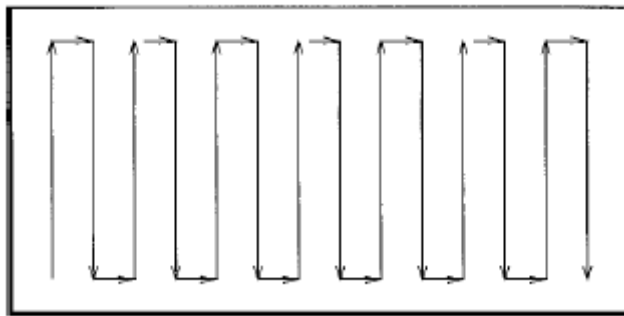


Figure 1.2 Boustrophedon Path

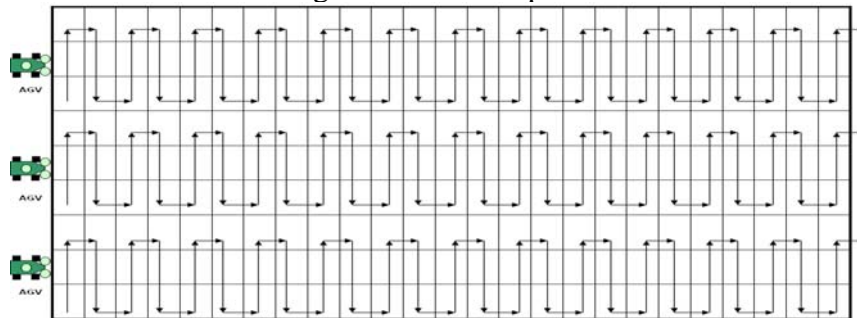


Figure 1.3 Example of Complete Coverage:
The number of lanes is divisible by the numbers of vehicles

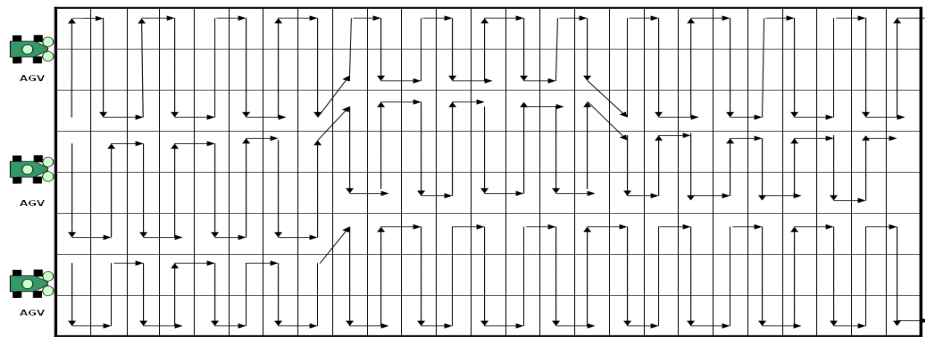


Figure 1.4 Example of Complete Coverage:
The number of lanes is not divisible by the numbers of vehicles

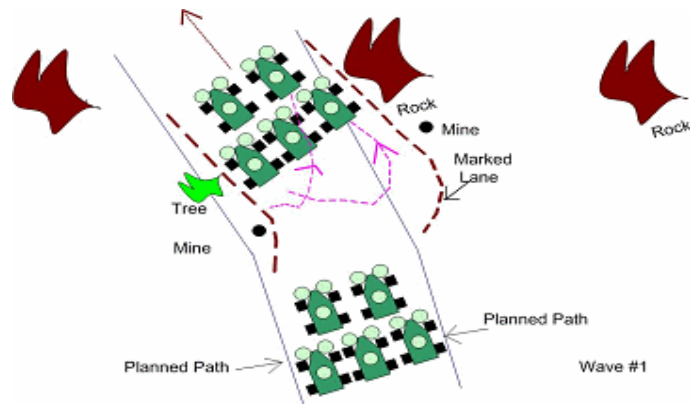


Figure 1.5 Coordinated Line wave for Breaching:
The team will redefine the path to avoid hazard

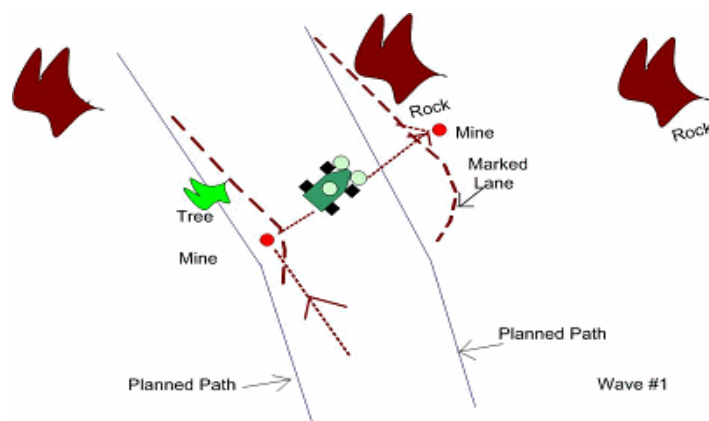


Figure 1.6 Point-to-Point Wave:
The vehicle moves directly to locations identified by the previous wave

CHAPTER 2

LITERATURE REVIEW

This chapter reviews relevant work associated with automated landmine detection and removal technologies and practices. Associated search technologies and optimization technologies are also reviewed. This body of knowledge has helped justify the need and has shaped the design of the MiCAT demining analysis tool presented in subsequent chapters.

2.1 Landmine Detection Technologies

According to Habib (2007), there are about 2000 different types of mines. There are more than 100 million landmines installed in 68 different countries around the world. In addition, two to five million new ones buried each year. Pressure, movement, sound, magnetism, or vibration can trigger a landmine (Habib 2007). Advanced mines can even be triggered by a magnetism without touching it, or by sensing the difference between friendly and enemy types of vehicles by a built-in signature catalogue (Fruergaard-Pedersen 2006). The more advanced the mine technology is, the more difficult the demining task will be. In addition, the type of terrain, soil composition, and climate variables might limit the capability of mine detection technologies. Mine may have also been corroded, waterlogged, impregnated with mud or dirt (Habib 2007). Consequently, modern mine technologies are not the only challenge associated with the demining task.

Environmental conditions can increase the challenge during the demining task (Habib 2007).

2.1.1 Sensor Technologies

The mine clearing challenges include building a reliable mine detection sensor, detection and clearing methods, searching strategies, and other advanced technologies that can support the deminer to remove the landmines. For those challenges, the sensor technology is most essential. Radar (ground penetrating (GPR), wideband, arrays, synthetic aperture radar), infrared and microwave radiometry, explosive vapor sensors, acoustic sensors, Electromagnetic Induction Metal detectors (EMI), magnetometers, Acoustic Imaging, Thermal Neutron Activation (TNA), Photoacoustic Spectroscopy, Nuclear Quadrupole Resonance (NQR), X-Ray Tomography, electrical impedance tomography, Neutron Back-scattering, Biosensors, and Commercial sniffers are some of the techniques which have been investigated to increase the high detection rate (quoted in Habib 2007; Rajasekharan and Kambhampati 2003). As there is no single sensor technology that can attain a good level of detection, using several complementary sensors and doing the sensor data fusion can approach the solution. Several mathematical theories have been proposed to model the sensor data fusion problem to improve mine detection rate, such as fuzzy logic, neural networks, statistical approach, belief functions (BFs), Dempster-Shafer Theory (Habib 2007; Milisavljevic and Bloch 2003; Ramaswamy et al. 2000).

Because the data are highly variable depending on the context and conditions, and there is ambiguity between several types, it is better to collect more than 1000

samples of each type of mine to analyze the mine sensor detection data (Milisavljevic and Bloch 2003). Since it is impossible to model each mine object, the data are not numerous enough to allow reliable statistical learning (quoted in Milisavljevic et al. 2000). The theory of belief functions (BFs) provides a non-Bayesian way of using mathematical probability to make user able to assess probabilities for related questions. The BFs user can then consider the implications of these probabilities for the question they are interested in (Shafer 1990). The ignorance, uncertainty and ambiguity can also be appropriately modeled (Milisavljevic and Bloch 2003). Milisavljevic and Bloch (2003) build the sensor fusion problem based on belief functions (BFs) in the framework of Dempster- Shafer (DS) theory. With BFs theory, Milisavljevic and Bloch (2003) can explore possibilities of modeling the mine detection problem without any statistical information on the data or on noise. The predicted mine sensor fusion result should not be considered the final result. Thus, they provide the deminer as much information as possible and leave the final decision to the deminer. Furthermore, the deminer's analysis also has been included for the modeling and combination process.

Ramaswamy et al. (2000) and Mudigonda et al. (2003) also model the multi-sensor information fusion problem based on the DS theory. In the research of Ramaswamy et al. (2000), they categorize the sensor information and use a supervised feed-forward neural network to learn the causality between the cluster information and the evidence of a given class of the buried object. Therefore, they can use DS evidential reasoning to accumulate different evidence from sensor channels to detect and differentiate between buried landmines and clusters.

Mudigonda et al. (2003) explores the concepts of multi-sensor data fusion based on the Dempster-Shafer (DS) evidential theory. They use a decision-level DS algorithm to combine the evidence from multiple sensors of the landmine detection system. Their research also operates a feature-level DS fusion algorithm on a set of features reported by the ground penetrating radar (GPR) sensor of the system. Their results, based on DS theory, yield that there is a higher probability of detection (Pd) value than other algorithms developed by GD Canada (e.g. heuristic algorithm, Bayesian inference, and Voting fusion concepts).

Therefore, in order to solve the sensor fusion model with less information, using the DS concept can obtain higher reliability than other algorithms.

2.1.2 Robot Technologies

Several robots have been built to support landmine detections to reduce the deminers' difficulties. Depending on the purpose of the robot, they can be divided into: Mine removal robots and Mine detection robots. Generally, the Mine removal robots are used to remove UXO or probable bombs. Depending on their ability, they can be divided into: the remote controlled and autonomous (Fruergaard-Pedersen 2006). Based on the differences of minefield environment, diverse types of robots have been built. The types include: wheeled robot, legged robot, and caterpillar robot. Usually, the wheeled robots have two or three wheels, and can perform a linear motion in any direction relative to its body. Mainly, the minefield environment is an unstructured environment and it makes the wheeled robots perform poorly within this environment in which the robots may face a vertical step or a discontinuous surface. Therefore, the

legged robots are developed to overcome the environmental difficulties (Colon et al. 2007; Rajasekharan and Kambhampati 2003). Debenest et al. (2003) propose automated buggies to perform anti-terrorism operations. Colon et al. (2007) also mention that Belgian Army uses the automated caterpillar EOD (Explosive Ordnance Disposal) platform for anti-terrorism operations. The caterpillar's control system is computer controlled through a micro-controller interface. Additionally, the scanning system, the motion controller, and the visual tracking and location computer have been mounted on the system to scan and acquire the scanned result.

2.2 Mobile Robot Team Concepts

Cooperative mobile robots have been used to solve complex problems that a single robotic system has difficulties accomplishing. There are several important issues concerning cooperative robotics research topic: traffic control, formation control, cooperation, and robotic architecture (Cao et al. 1997). The relevant works associated with robotics searches are reviewed in the following sections.

2.2.1 Robotic Behavior Control

The behavior-based architecture is almost identical to the reactive architecture. It contains the properties or even components of a purely reactive system. However, the computation of the behavior-based systems is beyond the reactive system. It can store various forms of state and can execute different representations (quoted in Dollarhidea and Agah 2003). Hence, by integrating several goal-oriented behaviors, simultaneously, the behavior-based systems can be used to navigate the robot team to waypoints, avoid hazards, and keep formation at the same time (Balch and Arkin 1998). As a result,

behavioral control (or formation control) can be applied in the MiCAT applications to make the UGV able to adapt the command to accomplish their committed tasks. Typically, the behavioral control system is, computationally, in several parts, and the sub-problem of the given assignment can be handled separately (Fruergaard-Pedersen 2006). The behavior consists of several separate components: inter-agent collision avoidance (i.e. traffic control (Cao et al. 1997)), velocity matching and flock centering. With a combination of these separate abilities, the robotic system can perform a specific geometric movement. Several behavior motor schemas: **move-to-goal**, **avoid-static-obstacle**, **avoid-robot** and **maintain-formation** are defined in Arkin and Balch (1998); Balch and Arkin (1995); Balch and Arkin (1998) to make the robot able to maintain a formation and move to a goal location, and protect it from hazard.

In the cooperative robotics discipline, the Formation and Marching Problems are important studies. In general, a specified pattern is needed in order for the group to move together (Cao et al. 1997). For example, the military-type formations can be lines, columns, diamonds, or wedges (see figure 2.1) (Balch and Arkin 1998; Cook et al. 1996; Hsu 2005; Hsu and Liu 2005a; Hsu and Liu 2005b; Hsu and Liu 2004). In order to form a different formation above, each robot has an assigned position based on its own identification number (ID) (Balch and Arkin 1998). Hsu and Liu (2005b) propose a Taxonomy of Formation Control based on the controlled abstraction of formation systems. They categorize the ground-based formation into: control abstraction (i.e. formation shape, reference type, and robotic control), and distinguishing ability (i.e. identification (marked as ID-formations) and anonymous (marked as ANO-formations)).

The formation shape could be one of the formation types mentioned earlier. Choosing a proper reference type depends on the robot's distinguishing ability. If a robot is in an AND-formation, generally the nearest or farthest neighbors are used as reference types, because a robot cannot refer to one or more *specific* robots. The other reference types: the virtual structure, unit-center, leader, neighbor, reference points, friend, directed edge, and queue status are all ID-formation, used as reference types (quoted in Hsu and Liu 2005b). Within these reference types, leader strategy is the most frequently used, for maintaining the mobile robot formation. Each robot can be directed to the correct location within a team, based on leader's or other robots' location, which is depending on which reference type it uses. Generally, the formation control research can be found in the research of wheeled ground-based mobile robots, aircraft, spacecraft, or underwater vehicles to support military missions (quoted in Hsu and Liu 2005b).



Figure 2.1 Formation for four robots (a) line, (b) column, (c) diamond, and (d) wedge (Balch and Arkin 1998)

Balch and Arkin (1998) make a performance comparison between four reference types. If one robot rotates, when using the *unit-center* type, the unit-center has to recalculate. Thus, the other robots have to rotate also, in order to maintain in the edge of the formation. This strategy makes the *unit-center* type has better performance than

the *leader-referenced* type, while the team maintains the *diamond* formation, because its smaller “moment of inertia”.

As mentioned earlier in Arkin and Balch (1998); Balch and Arkin (1995); Balch and Arkin (1998), the robots have different motor schemas to maintain formation and move to a goal location. The maintain-formation motor schema generates a movement vector. Thus, they propose three zones (see figure 2.2) to gain the value of the magnitude of speed and forward direction, and to maintain their motor schema. If the robot is in the **Ballistic zone**, it will be given the maximum gain. On the contrary, if it is in the **Dead zone**, zero magnitude will be given.

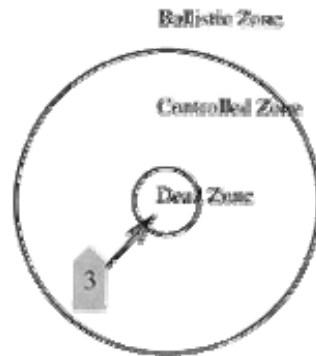


Figure 2.2 Zone for the computation of maintain-formation magnitude (Balch and Arkin 1998; Balch and Hybinette 2000)

A similar algorithm has been proposed in Liu et al. (2006). They use potential-based avoidance and following-wall behavior to make the robot able to avoid the obstacles. There is a difference between Balch and Arkin (1998) and Liu et al. (2006),

Liu et al. define a repel force f_{repel} and a following-wall behavior to go around obstacle.

This repel force is defined as $f_{repel} = \frac{k_1}{d_{obstacle} + \delta_1}$. It makes the robot gain a maximum

repel force when it approaches to the obstacle. If the robot is not on the same side of the target and needs to go around the obstacle, the following-wall behavior is turned on. The repel algorithm, used to control the team's formation and to avoid collisions, has also been discussed in Chen and Luh (1994) and Hsu and Liu (2005a).

Dollarhidea and Agah (2003) also propose a robot control rule to gain the robot movement magnitude. In their research, each robot can sense eight directions, and each of their control rules is made by a bit string of "141 X n" length. Each direction has two distance values and an entity value (a DDE) to decide the score. By matching the gained score with the rule set, the robot's action is decided. In addition, their collision prevention is based on the angle of the detected obstacle. In order to prevent collision with a detected obstacle, a redirection is established. In addition to the formation control in one team's movement, multi-team formation control can be found in Hsu (2005); Hsu and Liu (2005a); Hsu and Liu (2004)

There are several challenges associated with ground-based formation control. Those challenges include communication, stability, scalability, formation establishment, formation switch and multi-team control. Formation stabilization is a big challenge as the number of robots increase. This is usually due to communication problems, such as bandwidth limits, time delays, data losses, and broken links in an unstructured environment. ANO-formation is generally scalable to change the size of the team. Well-designed ID-formation has scalability to adapt to the formation changing. However, considering the cost of formation establishment and operation, ID-formation is larger than the AND-formation. It is due to the fact that the robots with specific IDs are

required to be at their specific position. Without considering the specific location, the AND-formation robot can move to the nearest positions. However, it has multiple configuration problems. Thus, if dynamic ID assignment is available, the cost could be reduced (Hsu 2005). Over and above, it has been observed in most formation control research that knowing the relative position between robots (e.g. the farthest, nearest neighbors, vector, leader) can prevent collisions, point the robot toward the goal, maintain the formation, and achieve the task.

2.2.2 Searching Applications

Researchers have proposed the use of robotic teams with formation control to perform military missions and rescue operations. Healey (2001); Ludwig (2000) propose a team of vehicles that sweep the minefield with a lawnmower search pattern (same as Boustrophedon path in figure 1.2). The vehicles maintain a defined space between each, side by side, to sweep the minefield. Depending on the user's defined percentage of overlap, the spacing could be either positive (i.e. gaps) or negative (i.e. overlap). Their initial idea is that, if with a higher overlap percentage, the minefield sweeping could get a higher coverage rate of the minefield. However, their experiment shows the opposite result. Because the spacing is smaller, the sweeping time will be higher. Also, this increases the required search time. In order to balance the task among the vehicles, they make the team move together in a row to perform the demining. Their strategy is that, if one of the vehicles is killed, the one next to it will replace its position and ID. Their result also shows that the formation could become staggered sometimes,

because communication could be a little bit delayed, after the start of a new formation (see figure 2.3).

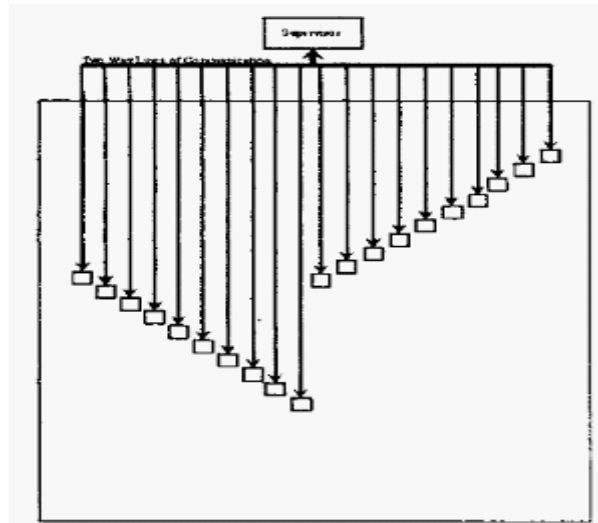


Figure 2.3 The staggered formation of a minesweeping team (Healey 2001; Ludwig 2000)

Other than the specific formation control problems above, swarm formation has been applied in natural heuristics to solve the combinational optimization problems (Bonabeau 1999). It has also been proposed to solve many kinds of landmine detection problems (see Cassinis et al. 1999; Chapman and Sahin 2004; Kumar and Sahin 2003; Munirajan et al. 2004; etc). Swarm intelligence is inspired by a natural behavior. As we know, the social insect societies are autonomous individuals, and they can transmit the information to each other. As a result, by grouping, the sensors information can be combined to maximize the chance of detecting predators or food (Balch and Arkin 1998).

Cassinis et al. (1999) propose strategies for the navigation of robot swarms for landmine detection. They describe several strategies for demining (e.g. Random, relay

clustering, flocking, swarming, formation, and comb movement). Their strategy requires robots capable of avoiding obstacles, finding mines, following a path, and maintaining a formation. The vectorial movements have been used to make the robots capable of these basic behaviors. For example, the vector is used to avoid obstacles, to achieve a goal, to maintain the position in a specific formation, and for maintaining the robot direction. Cassinis et al. (1999) make a comparison concerning the minefield environment and the appropriate demining strategy. A random formation performance is worse than having a specific formation in different minefield situations (contains many obstacles or not). But when many mines are undetected, the uncoordinated strategies are more efficient than the coordinated one. The relay clustering formation strategy can perform more efficient if the mines of the same kind are gathering together in the minefield.

Fruergaard-Pedersen (2006) also proposes a landmine detection strategy, using a swarm of low-cost robotic devices. In this research, a large robot, containing many features, is used to locate and clear mines. One of the features is a dynamic map of the area. This feature makes it able to distribute the tasks among the small robots. This research also proposes an alternateBot strategy to determine if possible mine locations contain mine or not. Further, there are two types of mission: area demining and road demining in this research. In general, the robot will not meet obstructions during the road demining task because they work on a certain road. Thus, the overlap is different between these two missions.

Likewise, the ant motion theory is also inspired by the natural behavior. It has been applied to many optimization problems. Munirajan et al. (2004) apply ant-colony foraging behavior on the mine detection problem. Their mine detection algorithm is a combination of stochastic, which is used in the foraging stage, and deterministic methods. After an ant enters a scent area, during the foraging period, it moves toward a randomly generated point in a deterministic manner. The ant's strategy is to follow the route along which the scent increases. Supposedly, the ant can find the mine position at the peak scent. The system can be more effective through inter robot communication. However, some difficulties may exist in implementing the simulated algorithm to the real world because the complexity of the situation requires a high degree of precision and resources.

Other than using behavior control to perform search applications, probabilistic approaches have also been proposed. Probabilistic search has been adopted by Acar et al. (2003); Gelenbe and Cao (1998); Zhang (2004); Zhang et al. (2001); Zhang et al. (2002). Probabilistic search in an unknown environment leads to two types of problems: one concerns the efficient construction of a mine distribution map, and the other concerns the use of this distribution map to generate an optimal search path (Acar et al. 2003). As the spatial distributions of mines will impact on how to deploy and move the mine sensors and detectors in the minefield, Gelenbe and Cao (1998) study directing autonomous searches using spatio-temporal distributions in order to utilize sensors and autonomous agents in the field efficiently. Therefore, they store the type, location and likelihood density of mines, and the type and location of agents in the "*Scenes*" as the

information units. These “*Scenes*” are “typically multi-sensory representations of the area in which mines are being sought and localized”. Therefore, the next move can be learned in an on-line manner with the prior information of the “*Scenes*” and the local information sensed by the individual robot. In order to optimize “the rate at which potential mine locations are being visited”, the rate into the “infinite horizon” at each step is calculated with *Simplified Infinite Horizon Optimization (SIHO)* algorithm. The *SIHO* algorithm is also used to avoid unnecessary repeated visits of the robot to the same points. This algorithm computes the transition rate $\{\mu^i(x, y, u, v)\}$ at each i th step in order to find the long run probability (after j th step). It makes the probability of a robot visit to a point in a search area match closely to the probability of finding a mine at that point. A Markovian field is estimated with the *SIHO* algorithm to direct the robot to the locations that probably contain mines. Compared to the work of Gelenbe and Cao (1998), and Zhang (2004), Zhang focuses on building a probabilistic map of mine locations. The probabilistic map is built based on two types of dispersion pattern models of the minefield. One is regular and the other one is scatterable patterns. Zhang builds a three level hierarchical spatial statistics model to show that the actual mine position differs from the intended mine position. He reproduces regular pattern by six scale and rotation parameters. His concern is how to use the collected information, inside the covered region, efficiently, to estimate the pattern parameters. A Bayesian approach is adopted in order to calculate the posterior distribution of the pattern parameters. After that, a model-partitioning algorithm, identifying all of the possible local maxima, is used to avoid the estimation becoming trapped in a local maximum or accepting

proposed parameters with near zero probability. With this algorithm, a global maximum is guaranteed. The Metropolis-Hastings algorithm is able to use local maximum likelihood information and limited computing resources to estimate global parameters of a regular pattern minefield. Zhang constructs a five-level hierarchical spatial point process model, for scatterable patterns, in order to model the location of unexploded submunitions dispersed from cluster bombs, fragments of the exploded submunitions, and the scattered metal particles unrelated to the cluster bombs. Having these probabilistic maps, a robot is able to implement full coverage of a sample of the target area.

2.3 Vehicle Routing Problem (VRP)

There is an abundant literature dealing with vehicle routing problems (VRP), such as: Alvarenga et al. (2007); Bräysy (2001); Bräysy and Gendreau (2002); Choi and Tcha (2007); Cordeau et al. (2001); Gribkovskaia et al. (2007); Li et al. (2006); Potvin and Bengio (1996); Potvin et al. (1996); Prins (2004); Shaw (1998); Tavakkoli-Moghaddam et al. (2007). The vehicle routing problem (VRP) can be stated as follows: using a set of vehicles to visit a set of customers exactly once with a prescribed constraint. “The goal is to produce a low cost routing plan specifying for each vehicle, the order of the customer visits they make” (Shaw 1998). This problem is an extension of the shortest path problem and the traveling salesman problem. In general, within VRP, each vehicle has a capacity constraint. Capacitated VRP (CVRP) is the base of the VRP problem (Russell and Lamont 2005). Shaw (1998) uses Large Neighborhood Search (LNS) to solve VRP problem. The VRP problem is defined as open VRP

(OVRP) if the constraint is that a vehicle does not return to the depot after servicing the customers (Li et al. 2006). Other extensions of VRP are: Vehicle Routing Problems with Time Windows (VRPTWs), the Periodic Vehicle Routing Problem with Time Windows (PVRPTW), and the Multi-Depot Vehicle Routing Problem with Time Windows (MDVRPTW) (Cordeau et al. 2001). According to the literature, several algorithms can be used to solve the VRP problem: Cluster first, route second (CFRS), Tabu search algorithm (TSA), Adaptive memory-based tabu search (BR), Backtracking adaptive threshold accepting (BATA), List-based threshold accepting (LBTA), Tabu search heuristic (TS), Adaptive large neighborhood search (ALNS), Evolutionary Strategies, parallel approach, fuzzy logic, branch-and-bound techniques, problem simplification, Genetic algorithm (GA), ant algorithms and simulated annealing (SA) (quoted in Li et al. (2006); Prins (2004) and Russell and Lamont (2005)).

Tabu search starts from an initial given solution. Next, each Tabu search iteration moves from the current solution to the best one in its neighborhood. In order to avoid cycling, the forbidden solutions of the neighborhood are used. In addition, several features are applied to enhance intensification and diversification mechanisms to explore a broad portion of the solution space (Cordeau et al. 2001). With these special characteristics, it has been proven that Tabu search (TS) outperforms other meta-heuristic approaches. Prins (2004) presents a simple but effective hybrid GA which can compete with TS. Besides the procedures of crossover or mutation, used to weaken the genetic transmission of information from parents to children, a special splitting

procedure is used to convert a chromosome into an optimal VRP solution at any time in his research.




The graph modeling formulation of VRP definition is the base of many combinatorial problems that run in real-time, like the Unmanned Aerial Vehicle (UAV) routing problem (quoted in Russell and Lamont (2005)). The UAV routing problem has been considered as VRP problem in Russell and Lamont (2005), Berger et al. (2007); Brown (2001); Harder (2000); Kinney (2000); O'Rourke et al. (1999); Russell and Lamont (2005); Ryan (1998). Berger et al. (2007) use a hybrid genetic approach. Kinney (2000) applies a Hybrid Jump Search and Tabu Search. O'Rourke et al. (1999) and Ryan (1998) apply Reactive Tabu search. Russell and Lamont (2005) apply a Genetic Algorithm to model their UAV problem. As a result, finding minimized vehicle traveling costs and time, the point to point (P2P) wave as proposed in Chapter 1 can be thought of as a VRP problem, by substituting the highly developed mine detection vehicle for delivery vehicles, and the potential landmine locations for depots.

CHAPTER 3

MiCAT DESIGN CONCEPT

This chapter is going to discuss the implementation of Mine Clearing Analysis Tool (MiCAT) within WITNESS®. The abilities of major variables, attributes, functions that will be used to build the MiCAT will be explained in this chapter.

3.1 The Simulated Minefield

A series of “Machines” as lanes, each lane having arbitrary lengths, has been modeled to represent a discrete area or patch of a two-dimensional minefield. The number and length of the lane depends on the width of the minefield. Each machine element (cell) represents the smallest patch of the minefield. The lanes are ordered from bottom to top as the Y coordinates and cells are ordered from left to right as X coordinates (shown in figure 3.1). Different icons are added to the simulated minefield to represent the landmine detection vehicles, obstacles, or landmines. The black square  represents a stone, icon  represents a tree, icon  represents an undetected landmine.

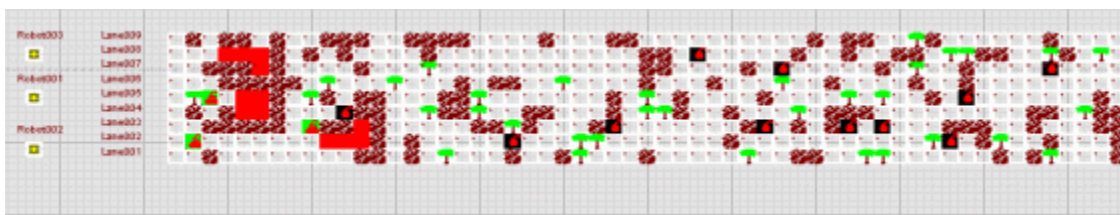


Figure 3.1 The Simulated Minefield

The MiCAT interface provides two methods for placing mines and obstacles in the simulated minefield. The first technique allows the tool to automatically place a specified number of mines or obstacles in the model using random distributions to determine their location. The second method allows the user to explicitly define the location of obstacles or mines within the minefield prior to each simulation run. This ability to design the minefield layout gives the analyst the opportunity to test a given mine detection resource scenario against a particular simulated landform. The MiCAT interface will also allow users to simultaneously use both methods of specifying the locations of mines and obstacles. In order to represent the landforms in MiCAT, “Machine” elements will carry the attributes (*Obj_Trap_Mine_Metal (i)*) that specify the actual state of the simulated minefield element. These attributes are categorized into four types of status: obstacle (stone or tree), trap, mine (detected or under detected), and metal. Letter “i” represents the type of object: obstacles (could either be a stone or tree, where $i \leftarrow 1$), trap (where $i \leftarrow 2$), mine (detected or under detected, where $i \leftarrow 3$), or metal (where $i \leftarrow 4$). Three algorithms have been created to specify the state of the simulated minefield element when the minefield model is initialized. One is to randomly define the state of an element as an obstacle. One is to randomly specify the state of an element as a landmine. The last one can explicitly define the certain elements as the obstacles (or landmine). The specific obstacle (or landmine) locations, type of the simulated object, mine number, obstacle number, sensor process time, transition point, turning degree, random seed (i.e. *X_OBSTACLE*, *Y_OBSTACLE*, *X_MINE*, or *Y_MINE*), are asked to input MiCAT Excel interface at the beginning of the model. These data are

loaded from EXCEL via the function *XLReadArray* (a WITNESS® built-in function) and stored in the variable *Minefeild_Data*. According to the random seeds, the simulated landmine and obstacle can be placed in a specific (either randomly generated or specific) location (X,Y). “**WHILE LOOP**” or “**FOR LOOP**” logic is used to distribute the desired number of mines (represented with the variable *NUM_MINE*) or obstacles (represented with variable the *NUM_OBSTACLES*) into the model. An excerpt of the code that expresses the obstacle (or landmine) generation algorithm is shown in Table 3.1.

Table 3.1 Obstacles Generation Algorithm

Algorithm: *RandObjDistribution ()*

```

for i←1 to maximum_lane do {
  for j←1 to 100 do {
    if MatrixMap(i,j) =1 then
      MatrixMap(i,j) at 0:Obj_Trap_Mine_Metal (i)←1
      SET ICON of MatrixMap(i,j) to 88 /*Represented by a black box*/
    elseif MatrixMap(i,j) =2 then
      MatrixMap(i,j) at 0:Obj_Trap_Mine_Metal (i)←1
      SET ICON of MatrixMap(i,j) to 87 /*Represented by a tree icon*/
  }
}









```

3.2 The Simulated Mine Detection Vehicle

In order to visualize the moving action of the mine detection vehicle, it is represented as a simulation entity referred to as a Part within WITNESS® simulation package. Within the fixed grid of simulated minefield elements, each vehicle has eight potential moves within the minefield grid: a) forward in its current lane, b) diagonally forward and into the lane to the vehicle's right, c) diagonally forward and into the lane to the vehicle's left, d) laterally to the lane on the right, e) laterally to the lane on the left,

f) backwards and into the lane on the right, g) backwards and into the lane on the left, and h) retreating backwards within its current lane. In order to visualize its moving action, eight symbols are used to represent these directions (shown in 2nd line of each row in the Table 3.2).


Table 3.2 Eight Moving Directions Of The Vehicle

(7) (-1,1) (g) 	(8) (0,1) (e) 	(1) (1,1) (c) 
(6) (-1,0) (h) 		(2) (1,0) (a) 
(5) (-1,-1) (f) 	(4) (0,-1) (d) 	(3) (1,-1) (b) 

Each vehicle carries attributes to indicate its identification number, types of mine detection sensor that are onboard, assigned sweeping area or lane, and the performance data that will be sent back to the control center. The “Part” in WITNESS® has limited power; therefore the “Machine” performs the functionality of the mine detection sensor(s) and decision processing. When the vehicle “Parts” progress through the minefield element machines, the machine decision processor reads the attributes associated with the vehicles to populate the “Machine” cycle time and the stochastic mine detection logic.

It is difficult to find a single sensor for mine detection that can reach the necessarily high detection rate in all possible scenarios. For that reason, this paper simulates the ability of a vehicle to carry multiple complementary sensors to support the single sensor (Milisavljevic and Bloch 2003). The “Machine” element is modeled as a five-cycle activity in order to represent multiple mine sensor detection activities. In one of the cycles, the simulated vehicle searches the adjacent minefield elements prior to

moving into that potential location (*NTA & LTA*) (it represents the coordinate of intended moving element) with the search logics. One function is defined to receive the values of the potential move (i.e. parameter *pos* and are shown in 1st line of each row in the Table 3.2) in order to earn two values: *Lane_Assignment_Index* is defined as a movement at Y-axis direction (1: makes upward movement (e.g. +y), -1: makes downward movement (e.g. -y)) and *N_Index* is defined as a movement along X-axis direction (1: makes forward movement (e.g. +x), -1: makes backward movement (e.g. -x)). Attribute *N* and *Lane_Number* are defined as current X and Y respectively. Therefore, ($N+N_Index$, $Lane_Number + Lane_Assignment_Index$) represents the coordinate of each potential intended moving element (i.e. (*NTA & LTA*)). Mine detection search logic and simple obstacle detection logic (this paper assumes that the simulated vehicle carries a 100% perfect obstacle sensor) are also defined (an excerpt of the code that is used to express the mine (obstacle) detection rule is shown in Table 3.3). The mine detection vehicle will not be allowed to enter a minefield element, which is defined as an obstacle (or landmine). It has to search another adjacent area. The other four cycles represent the mine detection sensor(s) carried by vehicle. If a specific sensor dwell time is longer or occurs after the vehicle motion has concluded, then additional time is used in the processing time for that cycle. If a sensor can perform its task in parallel with the vehicle motion, then no additional time is added for the current cycle. In addition to representing processing time, stochastic logic associated with these cycle constructs can determine, based on a given stochastic function, whether the sensor will fail to detect a mine, if present, or will conclude that a mine is present when it, in

fact, is not. If multiple sensors are present on the simulated vehicle, and because the reliability and detection ability of any sensor is scenario-dependent (Milisavljevic and Bloch 2003). A simple sensor fusion mechanism is used in this research to access the vehicle's belief that a mine is, or is not, present. Much more sophisticated sensor fusion mechanisms are presented in the literature (Milisavljevic and Bloch 2003). The structure of the MiCAT mine detection logic will support the integration of more sophisticated sensor fusion mechanisms as needed in the future. The current MiCAT implementation, each simulated mine detection sensor can be assigned a probability value that indicates that the sensor will detect a mine if it is present in the simulated minefield location. A similar probability value can also be assigned to determine if the sensor will indicate that a mine is present, when in fact it is not present in a given minefield location. During the execution of the simulation, random numbers will be generated and compared with these positive and false positive detection performance values for each sensor carried by the vehicle at each searched minefield location. This process generates a belief value for each sensor, indicating the presence of a mine. The sensor fusion mechanism will obtain a combination of the belief values of all sensors as the vehicle's belief. If this value is greater than a specified threshold, the vehicle will believe there is a mine in that location and will continue searching adjacent cells to go around. The *Obj_Trap_Mine_Metal* (3) attribute will be updated from 1 to 2 to represent that the vehicle believes there is a mine at this location. At the same time and the background of the machine element icon will be changed from black to green (). If a belief value is less than the threshold, the vehicle believes there is no mine. If a



mine is present and the vehicle fails to detect a mine that is present, the machine logic will use a random distribution to determine if the search vehicle will be destroyed by the mine or whether it will be left undetected in a patch of area marked as clear. If the vehicle is destroyed, the mine detection logic marks that mine location as clear and removes the vehicle from the minefield (symbolized as  at that location). If it has been defined as a non-mine element and the belief value is greater than the threshold, it will cause a false positive alarm, and the vehicle will still be safe. The reason for simulating the mine sensor's detection process is that if the vehicle fails to detect a mine and is destroyed by the mine, it will affect the subsequent mine detection activities. Other minefield attributes have been defined to allow a mine detection vehicle to detect and avoid cyclic search patterns that will entrap the vehicle. The *Obj_Trap_Mine_Metal* (2) attribute of a minefield location element is assigned a value of "1", signifying a Trap, if it is revisited during a cyclic search. If the vehicle's decision processor thinks the current cell is a trap, that cell will be marked as a "Trap" (). This "Trap" cell will be treated as a kind of obstacle in subsequent searches of that simulated minefield location.

Table 3.3 Obstacle and Mine Detection Algorithm

Algorithm: *GetMaxMinLnEach*

```

if MatrixMap(LTA,NTA) at 0:Obj_Trap_Mine_Metal (1)=1
  Obstcale_Flag  $\leftarrow$  1
elseif MatrixMap(LTA,NTA) at 0:Obj_Trap_Mine_Metal (2)=1
  Obstcale_Flag  $\leftarrow$  1
elseif MatrixMap(LTA,NTA) at 0:Obj_Trap_Mine_Metal (3)=2
  Obstcale_Flag  $\leftarrow$  1
elseif MatrixMap(LTA,NTA) at 0:Obj_Trap_Mine_Metal (3)=1
  Minebelief  $\leftarrow$  0
  if Random(1) $\leq$ 0.9 then Minebelief  $\leftarrow$  Minebelief+1
  if Random(2) $\leq$ 0.9 then Minebelief  $\leftarrow$  Minebelief+1
  if Random(3) $\leq$ 0.9 then Minebelief  $\leftarrow$  Minebelief+1

```

Table 3.3 - Continued

```
if Random(4) <= 0.9 then Minebelief ← Minebelief + 1
if Minebelief >= 2 then
  Obstcale_Flag ← 1
  MatrixMap(LTA,NTA) at 0:Obj_Trap_Mine_Metal (3)=2
  SET ICON of MatrixMap(i,j) to 104 /*Represented by a red box*/
else
  Explode ← 1
  MatrixMap(LTA,NTA) at 0:Obj_Trap_Mine_Metal (3)=0
  /*Used to erase the vehicle from the minefield after it has explode*/
  SET ICON of MatrixMap(i,j) to 33
```

CHAPTER 4

MiCAT APPLICATION SCENARIOS

Chapter 3 has explained how to build a simulated minefield and a mine detection vehicle within WITNESS®. This chapter will discuss the search algorithms for each application.

4.1 Simulation of a Scout Vehicle

The role of the Scout Vehicle Wave within the Military minefield breaching (MMB) scenario is to verify the feasibility of the breach path assigned by the MMB engineer. The job of the scout vehicle is to make sure that the breach lane searched by the autonomous vehicles is wide enough to support the size of troop formations or vehicles that will be expected to pass. The Scout Vehicle Wave model assumes that the scout vehicle will carry mine detection sensors capable of searching the ground directly in front of itself. It also assumes that the scout vehicle carries obstacle detection sensors, enabling it to identify and avoid obstacles out to the maximum desired breach lane width.

Because the breach lane must accommodate vehicles that are much larger than the scout vehicle, the obstacle detection algorithms must support a parametric breach lane width and accommodate non-zero turning radius vehicles. To accomplish this, the Scout Vehicle Wave search algorithm must support the searching of multiple minefield

cells prior to each simulated vehicle move. Two searches are conducted for each move of the simulated scout vehicle. First, the next cell to be occupied by the vehicle will be searched for a mine. Second, a group of cells within the proposed breach path will be searched to see if they contain obstacles that would block the path of the breach formation.

In order to support this concept of a parametric breach lane width within the MiCAT system, the tool's user interface must allow user to specify the desired width of the breach path. Within the current MiCAT prototype, the lane width is defined relative to the size of the simulated scout vehicle. The current algorithms implemented within the tool will allow a minimum lane width that is three times the width of the simulated scout. The value entered by the user is used to determine the number of "scout vehicle widths" that must be searched on either side of the simulated scout vehicle to ensure that the desired breach lane width is obtained. The simulation logic variable "*int_CoveredL*" has been defined to record this value. The width of the desired breach path is then equal to " $2*int_CoveredL+1$ ". In order to define a simulated breach path that could accommodate non-turning radius vehicles we needed to search for obstacles far enough ahead of the simulated scout vehicle so that the resulting searched area was large enough to provide vehicles room to maneuver. In the current MiCAT implementation this value is equal to " $2*int_CoveredN$ ", where the variable "*int_CoveredN*" is equal in size to the variable " $2*int_CoveredL$ ".

The simulated scout vehicle has the ability to project a search area that is " $2*int_CoveredL+1$ " wide and " $2*int_CoveredN$ " long in one of three potential

directions within the simulated minefield grid. The search area can be projected in front of the scout vehicle. The search area can also be projected at an angle of 45° to the left of the scout vehicle or at an angle of 45° to the right of the scout vehicle. The direction selected by the search algorithm is determined by the current location of the scout vehicle with respect to its assigned breach path and the presence or absence of mines or obstacles during previous search cycles.

The actual search algorithm implemented within MiCAT uses “*FOR LOOP*” logic to check the state of all of the simulated minefield cells. The total number of cells searched is equal to “ $2*int_CoveredL+1$ ” X “ $2*int_CoveredN$ ”. If the scout is on its desired search path or does not need to go around an obstacle or mine, the searched area forms a rectangle extending in front of the scout vehicle. If the scout needs to move out of its current lane, the search area will form a parallelogram projected at an angle of 45° to the left or right of the scout’s current location. The variable “*int_L_Diff*” is used to project a diagonal line from the scout’s current position into the search space. If the value of “*in_L_Diff*” is 1 then the search area will be projected at an angle of 45° to the left of the scout. If the value of “*in_L_Diff*” is -1 then the search area will be projected at an angle of 45° to the right of the scout.

For every cell within the search area, the algorithm will check the state of the simulated minefield to see if it contains an obstacle. To accomplish this, the algorithm will index through each of the cells in the search area using the “*int_Ary_RX*” and the “*int_Ary_RY*” variables to define a relative grid location with respect to the scout’s current position. If an obstacle is detected, the algorithm will next determine if it is

closer to the left boundary or the right boundary of the search space. The algorithm will use this information to determine if it will shift its current search direction to either the left or the right in an effort to find a clear path around the obstacle. The logic used within this scanning algorithm is summarized in Table 4.1.

Table 4.1 Scout Scanning Minefield Algorithm

Algorithm: *ScanningAround()*

```

for i ← 1 to (2 * int_CoveredN) do {
  for j ← 1 to (2 * int_CoveredL + 1) do {
    x ← int_Ary_RY - j + int_L_Diff /*Relative x*/
    y ← i - int_Ary_RX + int_N_Diff /*Relative y*/
    if LocationDetector(x,y,1) = 1 then
      ObjDetecAry_R1 (i, j) ← 1
      Obj_Target_v (1,int_Target) ← LTA
      Obj_Target_v (2,int_Target) ← NTA
      int_Target ← int_Target + 1
    }
  }
if int_Target > 1 then
  upSpace ← Lane_Number + int_CoveredL - Obj_Target_v (1,1)
  downSpace ← Obj_Target_v (1,int_Target-1) - (Lane_Number - int_CoveredL)
  ObjDetecAry_R1_Space (i,1) ← upSpace
  ObjDetecAry_R1_Space (i,2) ← downSpace
else /*Nothing occupied, means maintaining a required space */
  ObjDetecAry_R1_Space (i,1) ← (2 * int_CoveredL + 1)
  ObjDetecAry_R1_Space (i,2) ← (2 * int_CoveredL + 1)
if flag_int_L_Diff=0 then /*On the desired lane*/
  /*Checking if it needs to scan diagonally to go around*/
  if int_go_up = 1 then
    int_L_Diff ← int_L_Diff + 1
  if int_go_down = 1 then
    int_L_Diff ← int_L_Diff - 1
  else if flag_int_L_Diff > 0 then /*Above the desired lane*/
    int_L_Diff ← int_L_Diff - 1
  else /*Below the desired lane*/
    int_L_Diff ← int_L_Diff + 1

```

The Scout vehicle is being asked to find the lane that will allow it to move to the target most quickly. Therefore, it is assumed there is a desired search direction or path.

This desired path would be determined prior to the beginning of the breaching maneuvers by an MMB engineer. If the above searches fail to detect obstacles or a mine, the scout vehicle can occupy the searched minefield grid. If the searches detect an obstacle or mine, it means the path width is less than the required breaching lane width. Therefore the next most favorable potential motion will be chosen, and the Scout will diverge from the desired search path. To accomplish this, each simulated vehicle entity will carry a set of attributes to represent the current Y_i , and desired path, etc. The equation: “ $Lane_Number$ (i.e. Current Y_i) - $Lane_Assignment$ (i.e. desired path)” is used to determine its relative position to see how far the vehicle is above or below its desired path. If the Scout is beyond its desired path, it has to apply a search criterion to move back. Five search criteria have been classified in Table 4.2 to make the Scout vehicle able to stay on its desired path, or go around the obstacle. At the same time, an attribute L_Diff is defined to make the Scout able to scan a diagonal line at 45° to the left or right, if it has to find an alternative path.

Table 4.2 Scout Searching Desired Path Algorithm

Algorithm: *PerformSearch()*

*/*pattern A- above the desired lane, perform downward algorithm */*

if Current $Y_i >$ its desired lane **and** flag attributes are zero **then**

$L_Diff \leftarrow 0$

$flag_L_Diff \leftarrow Lane_Number - Lane_Assignment$

ScanningAround () to check downward path

$flag_L_Diff \leftarrow 0$

if it is clear **then**

moves downward (\searrow) to return to its original lane

else

MoveForward() to go around

*/*pattern B- below the desired lane, perform toward up algorithm */*

else if Current $Y_i <$ its desired lane **and** flag attributes are zero **then**

$L_Diff \leftarrow 0$

Table 4.2 - *Continued*

```

    flag_L_Diff ← Lane_Number - Lane_Assignment
    ScanningAround () to check upward path
    flag_L_Diff ← 0
    if it is clear then
        moves upward (↗) to return to its original lane
    else
        MoveForward() to go around
/*pattern C-prior move is beyond the desired lane and not able to go
around the obstacle */
    else if flag_ObjDetec_up=1 then
        make upward movement (↗) to go around the obstacle
/*pattern D-prior move is beyond the desired lane and not able to
go around the obstacle */
    else if flag_ObjDetec_down=1 then
        make downward movement (↘) to go around the obstacle
/*pattern E- maintain on current path to stay or find an alternative
path to bypass the obstacle*/
    else then
        MoveForward()

```

When the Scout has taken a new path to avoid an obstacle, it should stay on that path, until it is able to return to the desired path. In Table 4.2, a search pattern is defined to determine if the Scout has to stay on its current route, return to the desired path, or go around an obstacle. The algorithm uses “**FOR LOOP**” logic to index from the nearest cell to farthest cell to determine if the breach lane, scanned by the autonomous vehicles, is wide enough to support the size of troop formations, in which, attributes *flag_ObjDetec_up* and *flag_ObjDetec_down* are triggered to make the Scout bypass a detected obstacle. The assignment *flag_ObjDetec_up* ← 1 makes the scout take an alternative route in the upward direction. The assignment *flag_ObjDetec_down* ← 1 makes the scout move downward to go around the obstacles. Both attributes will be reset to zero after the Scout has passed the obstacle location.

Table 4.3 Keeping The Scout Stay On Current Route Algorithm

Algorithm : *MoveForward ()*

```

ScanningAround()
Ary_i ← 2 * int_CoveredN
Ary_j ← 2 * int_CoveredL + 1
int_space_availble ← 1 /*Checking flag*/
for i ← 1 to Ary_i do {
    if ObjDetecAry_R1_Space (i,1) < (Ary_j - i + 2) or ObjDetecAry_R1_Space (i,2)
    < (Ary_j - i + 2) then
        /*Has to re-define the path, if one of the front columns has no enough space*/
        int_space_availble ← 0
        /*Checking if upper path is wider or lower path is wider */
        if ObjDetecAry_R1_Space (i,1) > ObjDetecAry_R1_Space (i,2) then
            int_go_up = 1
            int_go_down = 0
        else
            int_go_up = 0
            int_go_down = 1
    }
if int_space_availble = 1 then /*Moving path maintains a required space*/
    get cell information & move forward (→)
else /*Re-define the path and check if it is available*/
    ScanningAround()
    if int_go_up = 1 then
        if int_ObjDetec_Ary_Sum = 0 then
            flag_ObjDetec_up ← 1
            flag_ObjDetec_down ← 0
            get cell information & move upward (↗)
        else
            int_go_up ← 0
            int_go_down ← 1
            ScanningAround()
            if int_ObjDetec_Ary_Sum = 0 then
                flag_ObjDetec_down ← 1
                flag_ObjDetec_up ← 0
                get cell information & move downward (↘)
            else if int_go_down = 1 then
                /*Depends on how far the Scout needs to scan*/

```

.....

It is assumed that the scout vehicle can only detect mines in its immediate

vicinity, as a result only the next cell to be occupied by the vehicle will be searched for

a mine. The Scout is not able to maintain a required turning radius prior to moving forward to go around the detected mine. In order to enable the Scout vehicle go around the landmine efficiently, the Scout Vehicle Wave simulation engine marks that detected landmine as a kind of obstacle. Thus, the Scout can perform the obstacle algorithm to go around the detected landmines. If the Scout detects a mine, the value of attribute *POS_Flag* is set to “1” to ask the Scout to retreat. The value of attribute *POS_Flag* is set to “0”, until the scout has retreated to a large enough distance to bypass the detected landmines. The logic of the bypass algorithm is summarized in the Table 4.4.

Table 4.4 Scout Bypass Landmine Algorithms

Algorithm: *ScanningWay* ()

```

if POS_Flag=0 then
    PerformSearch
else
    int_BackSteps  $\leftarrow$  int_BackSteps + N_Index /*Retreat space indicator*/
    if int_BackSteps>0 then
        /* Scout has to keep moving backward to retain a turning radius*/
        POS_Flag  $\leftarrow$  1
    else
        POS_Flag  $\leftarrow$  0

```

Within this simulation, the vehicle can negotiate the mines and obstacles, and identify a breach path free of obstacles of a given width. The obstacle search algorithms consider turning radius constraints in their search for a feasible path. Figure 4.1 shows the above search procedures.

The simulation results have shown the behavior and performance of a single vehicle moving through a minefield while attempting to locate a path, wide enough to support a given breach formation (see figure 4.2).

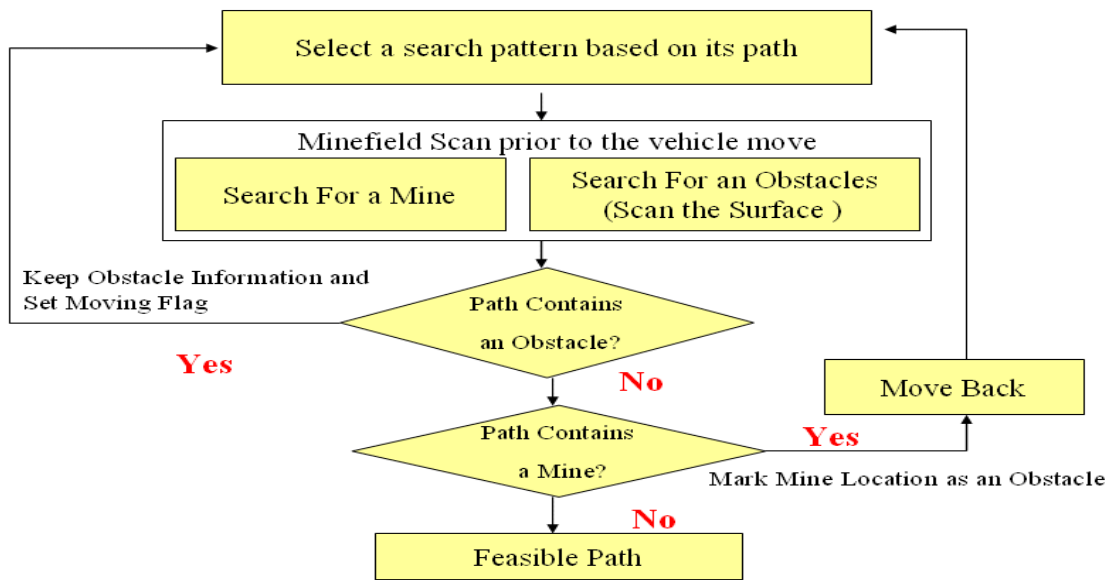


Figure 4.1 Breach lane identifying procedure for a Scout

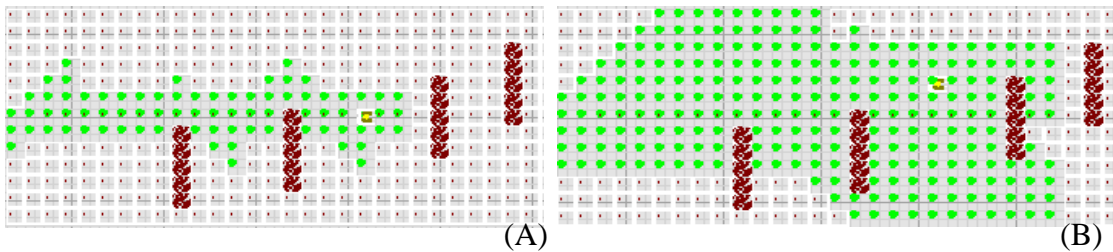


Figure 4.2 Simulation of a Scout Vehicle searching for a breach lane (A) three units wide and (B) five units wide

4.2 Simulation of a Coordinated Team of Mine Detection Vehicles

Robot or unmanned vehicle formation control has been an active area of research for several years. Behavior-based techniques that simultaneously integrate several goal-oriented behaviors have demonstrated the ability to navigate a team of cooperative autonomous vehicles through a series of waypoints while maintaining formation and avoiding obstacles (Balch and Arkin 1998). Examples of atomic goal-oriented behaviors that can be integrated into more comprehensive behavior-based

control mechanisms include inter-agent collision avoidance, velocity matching, and flock centering behaviors (Balch and Arkin 1998). With a combination of these separate behaviors, the individual robots within a team can be made to perform a specific geometric movement. Several behavior motor schemas: move-to-goal, avoid-static-obstacle, avoid-robot, and maintain formation are defined in (Arkin and Balch 1998; Balch and Arkin 1995; Balch and Arkin 1998) to make a team of robots maintain formation, move to waypoints, and negotiate obstacles encountered along the team's path. Formation control strategies that define the relative positions, or roles of vehicles within a team have also been investigated (Hsu and Liu 2005b). The potential for using multi-vehicle behavior-based formation control strategies for mine detection has also been identified (Fruergaard-Pedersen 2006). This section will discuss two multi-vehicle mine detection strategies: humanitarian minefield reclamation (HMR) and military minefield breaching (MMB) operations.

4.2.1 Military minefield breaching (MMB)

Within our multi-wave MMB scenario, once the Scout Vehicle Wave has found a feasible path wide enough to support the breach formation, the breach path must be searched across its entire width for mines. A coordinated team of mine detection vehicles works in a coordinated fashion in order to ensure that the entire breach path is searched. The motion of the mine detection vehicle formation is coordinated within the pre-plan lane to ensure that no gaps in the search area are created.

In order to maintain a line formation and make the vehicle able to determine its relative formation position to the leader vehicle, vehicle identification numbers (ID)

(Balch and Arkin 1998) are used. In addition, each robot can be directed to the correct location within a team, based on leader's or other robots' location (Hsu and Liu 2005b). In the simulation, an algorithm is used to make each vehicle check its surrounding minefield status when it enters a new location. Each vehicle in the formation searches for mines directly in front of itself. The vehicles on the end of the line are also required to search the simulated minefield grid locations laterally beside and diagonally in front of and away from the end of the line.

With the above algorithm, each cell status is posted to an array variable “*Status(i)*” (“*Status(i)*” $\leftarrow 0$ represents that there is no mine or obstacle in that location, and “*Status(i)*” $\leftarrow 1$ is the opposite). Letter *i* represents the lane relative position (where *i* represents the vehicle's front cells from top to bottom). The central vehicle in the coordinated team will perform an algorithm to pole the search results of all the members of the team (i.e. *Status(i)*). To simulate this multi-vehicle search technique, “**FOR LOOP**” logic is run for “ $2 * \text{int_CoveredL} + 1$ ” times (depending on the size of the team) to earn the value of *StatusSum(i)* (where $i \leftarrow “1”$ represents the direction \nearrow , “2” represents the direction \rightarrow , and “3” represents the direction \searrow). Based on the search results obtained from the team members and the relative location of the search team compared to its desired path (by checking the available status of *StatusSum(i)*), the center vehicle will move to the next location. An equation “*Lane_Number - Lane_Assignment*” is defined in Table 4.5 to check the relative location of the search team in order to determine if the team has moved beyond the pre-planned path or not. When the landmines or obstacles block the path (if all the directions of \rightarrow , \nearrow , and \searrow are

not available), the center vehicle re-defines the breaching path by changing direction or considering the turning radius to move several steps back to a place where the center vehicle is able to lead the team to go around the obstacles. Then, the other vehicles can move to the positions relative to the center vehicle to maintain a line formation. By subtracting or adding to the center vehicle's lane number (based on the $RI_Position(i,j)$ information in the command center), the intended moving location of the other vehicles are obtained.

Table 4.5 Algorithms To Search The Breaching Path

Algorithm: *ScanningWay* ()

```

if BackSteps_Count <= 0 then
  for i ← 1 to 2* int_CoveredL+1 do {
    j ← i+2* int_CoveredL
    for Status_i ← i to j do {
      StatusSum (i) = StatusSum (i) + Status (Status_i)
    }
  }
  /*If Current Yi > its desired lane, it is above the desired lane, and so on*/
  /*It represents "Lane_Number - Lane_Assignment >0"*/
  if Current Yi < its desired lane and StatusSum (1)=0
    ToPosAssigned (1) (i.e. move to 315 degree direction)
  elseif Current Yi > its desired lane and StatusSum (3)=0
    ToPosAssigned (3) (i.e. move to 45 degree direction)
  else
    MoveForward ()
else
  /*Have to move several steps back to go around the obstacles*/
  if Back_Alt_way =1 then
    direction_chk ← 1
  else
    NTA ← Steps (2,BackSteps_Count)
    LTA ← Steps (1,BackSteps_Count)

```

If the team does not need to move back to its desired path, an algorithm *MoveForward*(defined in Table 4.6) is used to make the team move straight forward or

to retreat to go around the obstacles or mines. In order to prevent the team from moving in a back and forth cycle while moving backward (Dollarhidea and Agah 2003), the team could move straight backward or follow the way it came from. If it needs to move straight backward and does not follow the way it came from, “*Back_Alt_way*” is set to “1”. At the time the team has to move backward to go around the obstacles, a distance variable *BackSteps* will be posted to the command center to represent a moving backward request. This distance is a function of the required width of the breach lane and the turning radius capabilities of the breach formation. By checking the value of *BackSteps* whenever the vehicle moves to the new location, the center vehicle can determine how many grid locations the formation of vehicles must go back. The more obstacles or mines occupying the front locations, the bigger the variable *BackSteps* is. A two dimensional integer array named “*Steps(i,j)*” is used to store the last four locations that the lead vehicle has just passed through. This two by four integer array is used to represent the most recently swept steps. Letter “i” of *Steps(i,j)* has two values: 1 and 2. “1” represents “Lane number (i.e. the Y-coordinate)” and “2” represents “N (i.e the X-coordinate)”. Letter “j” of *Steps(i,j)* represents the index in *Steps(i,j)*. The *Steps(i,j)* array is used by the lead vehicle to guide the search formation backwards through a series of the team previous moves. The variable “*BackSteps_Count*” is used to indicate the number of steps the search team has retreated. In addition, the *BackSteps_Count* variable increases by one whenever the team moves backwards one step. After that, if “*BackSteps = BackSteps_Count*” and “*Best_N > N*”, the center vehicle knows that the whole team has enough space to go around the obstacles. At the same time, the value of

“BackSteps” and “BackSteps_Count” are reset. This means that whole team does not need to move backward and the team can start to re-define the breaching path to go around the obstacles.

The attributes “UP_FLAG” and “DOWN_FLAG” have been used as flags to make the team able to scan the entire pre-planned path (at the same X-location). Whenever the team has reached its lower boundary of the pre-planned path and is not able to move further than current X, “UP_FLAG” is triggered. On the other hand, if the vehicle searches in the other direction and is not able to move further, “DOWN_FLAG” is triggered. By checking these two attributes, the team can skip moving downward (or upward) to look for a feasible path without getting stuck.

Table 4.6 Algorithms To Stay On The Path For Breaching Team

Algorithm: *MoveForward ()*

```

if StatusSum (2) = 0 and N >= Best_N then
    ToPosAssigned (2)
    ToCell ()
else
    /* Memorize the obstacle locations*/
    R1_Position_Obj (1,1) ← R1_Position (1,1)
    R1_Position_Obj (2,1) ← R1_Position (2,1) + 1
    BackSteps_Count ← 0
    BackSteps ← 0
    if StatusSum (3) = 0 and UP_FLAG <> 1 then
        ToPosAssigned (3) /*move downward (↘)*/
        ToCell ()
    elseif StatusSum (1) = 0 and DOWN_FLAG <> 1
        ToPosAssigned (1) /*move upward (↗)*/
        ToCell ()
    else /*Retreated to hold a turning space*/
        if StatusSum (2) = 1 then
            if Status (3) = 1 then
                BackSteps ← BackSteps_Count + 1
            else
                BackSteps ← BackSteps_Count + 2

```

Table 4.6 - Continued

```

/*Flag triggered when reach the boundary*/
if Lane_Number = MinLane + 1 then
    DOWN_FLAG = 0
    UP_FLAG = 1
elseif Lane_Number = MaxLane - 1
    DOWN_FLAG = 1
    UP_FLAG = 0
elseif StatusSum (2) = 2 then
    BackSteps ← BackSteps_Count + 1
elseif StatusSum (2) = 3 then
    BackSteps ← BackSteps_Count + 2
    BackSteps_Count ← BackSteps_Count + 1
    LTA = Steps (1,BackSteps_Count)
    NTA = Steps (2,BackSteps_Count)
if LTA <> Lane_Number then
    direction_chk ← 1

```

The above breach lane identifying procedure of a coordinated team is shown in figure 4.3. The simulation result (see figure 4.4) demonstrates that the above algorithms and flag configurations of the team of mine detection vehicles enable them to maintain an acceptable line width and turning radius of the search path, if an obstacle or mine is encountered.

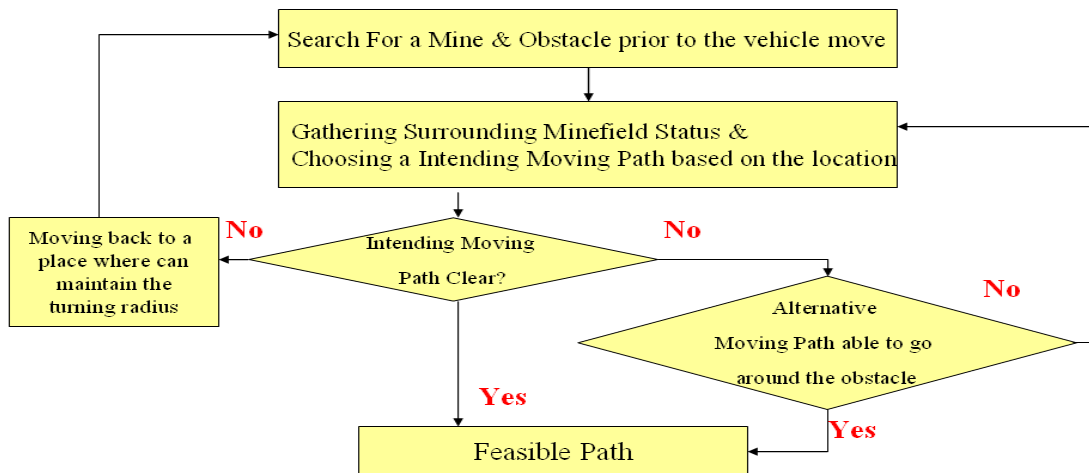


Figure 4.3 Breach lane identifying procedure of a coordinated team

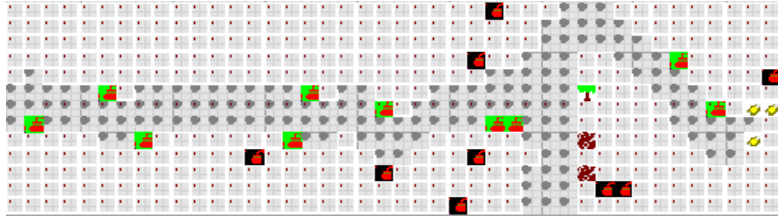


Figure 4.4 The three vehicle breaching team runs first providing 100% coverage for the breach lane

4.2.2 Humanitarian Minefield Reclamation (HMR)

The objective of the HMR application is to remove 99% of all mines within a given area. In order to search the entire minefield, the HMR scenario applies the Boustrophedon path (mentioned in Section 1.3) to complete the mine detection operation.

The HMR scenario assumes the HMR engineer has enough landform information beforehand to define each transition point of the minefield in MiCAT. A transition point is the location (highlighted in figure 4.5) where the size of search area is changed. With that information this model is able to distribute the area search task equally among the vehicles. In addition, the mine detection vehicle can change its direction to fulfill its search task in its re-assigned search area. The landform information is a list of the transition point information. The transition information includes the number of transition points (“ $N_Transition$ ”), the value of X-axis of the right most transition point (“ $Last_Transition$ ”), and a list of transition points (“ $TransitionN(i,j)$ ”. Letter “ i ” represents the transition points place in the order). Each i^{th} transition point lists the values of X-axis of i^{th} transition point ($TransitionN(i,1)$), and the values of the bottom-most ($TransitionN(i,2)$) and the up-most ($TransitionN(i,3)$) lane of

that point, which are used to make each mine detection vehicle able to change its direction toward its own search area. In addition, the numbers of mine detection vehicles (represented by the variable “*TotalRob*”) that will be used to sweep the minefield are also needed. This information can be defined in the EXCEL interface.

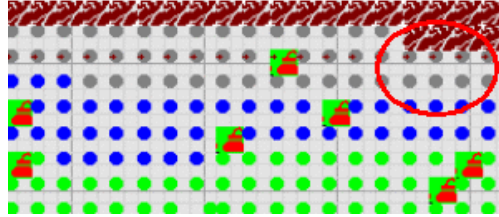


Figure 4.5 A Turning Point that causes the HMR search area to be decreased

Each vehicle’s task is assigned at each transition point, along with the size between two transition points (e.g. the first transition area is between *TransitionN(1,1)* and *TransitionN(2,1)* and is defined as a transition area). Each transition area is divided into “*TotalRob*” sections, with a width of “*TransN*” in each section. In order to obtain “*TransN*”, “*Trans_Cnt*” is used to represent the index of the current transition point. Equation (4.2) is used to obtain “*TransN*” when the mine detection has reached the right most transition point. Otherwise, Equation (4.1) is used.

$$TransN=(TransitionN(Trans_Cnt+1,1)-TransitionN(Trans_Cnt,1))/TotalRob \quad (4.1)$$

or

$$TransN=(Last_Transition-TransitionN(Trans_Cnt,1))/TotalRob \quad (4.2)$$

We first use equations (4.3-5) to obtain the total number of lanes (“*TotalLane*”) in a transition area, the quotient (“*Ln_Rob*”), and the remainder (“*Ln_Rob_Remain*”). Then, a list of bottom most and up-most lane numbers (represented by the array variables:

$TransionList_Min$ and $TransionList_Max$) of each vehicle's transition area, is obtained by whichever vehicle reaches a transition point first.

$$TotalLane = TransitionN(i,3) - TransitionN(i,2) + 1 \quad (4.3)$$

$$Ln_Rob = TotalLane / TotalRob \quad (4.4)$$

$$Ln_Rob_Remain = MOD(TotalLane, TotalRob) \quad (4.5)$$

In order to equally divide the vehicles' lane assignments, an algorithm is used to assign at least " Ln_Rob " lanes and additionally portions of " Ln_Rob_Remain " lanes to each vehicle in each transition area. This means that in each transition section, Ln_Rob_Remain vehicles will scan " Ln_Rob+1 " lanes. For example (see figure 4.6), if four vehicles are used to completely search a transition area with thirteen lanes, this transition area will be divided into four transition sections. In addition, each section is divided into four parts, perpendicularly. Thus, each vehicle can scan three lanes in each section and four lanes in one of the sections. This task procedure is shown in figure 4.7.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
13	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
12	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
11	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
10	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4
9	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
8	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
7	2	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3
6	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
5	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
4	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 4.6 An example of HMR's Lane Assignment for four vehicles

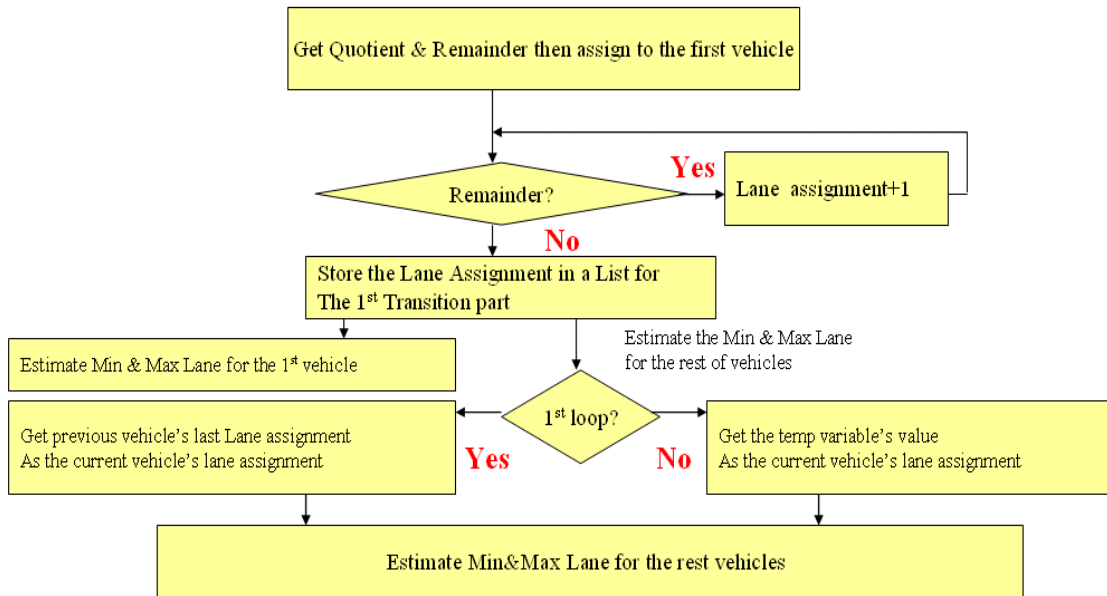


Figure 4.7 A Task Assignment procedures for HMR search

Table 4.7 and Table 4.8 (are defined in algorithm *GetMaxMinLn*) express the logic that is used to obtain a list of lanes that bind the vehicle's search in an area for each transition section. Table 4.7 uses "**FOR LOOP**" logic to run "*TotalRob*" times to obtain the lane assignment list, the bottom most and up-most lane numbers of the first section: "*TransitionLnList*", "*TransionList_Min (1,i)*" and "*TransionList_Max(1,i)*" ("1" means the first section, and letter "i" represents section order). "*Ln_Rob_Remain*" is used to count the remainder of the lanes. Thus, the vehicle can be assigned one more lane when the value of "*Ln_Rob_Remain*" does not run out during the assignment. This algorithm is able to balance the task among the vehicles, by distributing the remainder of the search lanes to different sections, even if the number of lanes is not divisible by "*TotalRob*". Next afterward, a bubble sort algorithm is used in Table 4.8 to rotate the value of "*TransitionLnList (i)*" "*TotalRob-1*" times to distribute the lane assignment to

the remaining sections. For example, the first rotation rotates the 2nd vehicle's search lanes as the 1st vehicle's search lanes, and so on. Therefore, this algorithm can generate a list of each transition area's turning points.

Table 4.7 Algorithms To Get HMR's First Lane Assignment

Algorithm: *GetMaxMinLn-1*

```

/*Ln_Rob_Remain_A as attribute of the temporary remainder holder*/
/*Ln_Rob_A stores the temporary accumulative lane assignment*/
/*Ln_Rob_B as the number of lane assignment */
/*TransionSecList used to store a list of turning points in a transition area*/
/*TransitionN (Trans_Cnt,2) this transition point's bottom most lane*/
Ln_Rob_A ← 0
Ln_Rob_Remain_A ← MOD (TotalLane / TotalRob)
Ln_Rob ← TotalLane / TotalRob
TransionSecList(i) ← TransitionN (Trans_Cnt,1)
for i ← 1 to TotalRob do {
    L_Cnt ← 0 /*Used to determine the remainder's value*/
    if Ln_Rob_Remain_A > 0 then
        L_Cnt ← 1
        Ln_Rob_Remain_A ← Ln_Rob_Remain_A -1
        Ln_Rob_B ← Ln_Rob + L_Cnt
        MinLane ← TransitionN (Trans_Cnt,2) + Ln_Rob_A
        MaxLane ← MinLane + Ln_Rob_B -1
        Ln_Rob_A ← Ln_Rob_A + Ln_Rob_B
        TransionLnList (i) ← Ln_Rob_B
        TransionList_Min (1,i) ← MinLane
        TransionList_Max (1,i) ← MaxLane
    }

```

Table 4.8 Algorithms To Get HMR's Lane Assignments

Algorithm: *GetMaxMinLn-2*

```

element_tmp ← 0
if Trans_Cnt = index of the last transition point then
    TransN ← (TransitionN(Trans_Cnt+1,1) - TransitionN(Trans_Cnt,1)) / TotalRob
else
    TransN ← (Last_Transition - TransitionN(Trans_Cnt,1)) / TotalRob
for i ← 2 to TotalRob do {
    Ln_Rob_A ← 0
    TransionSecList(i) ← TransitionN (Trans_Cnt,1) + TransN * (i - 1)
    for j ← 1 to TotalRob do {

```

Table 4.8 - Continued

```

if j=1 then
    Ln_Rob_B ← TransitionLnList (TotalRob)
    /*Used to get the last vehicle's lane assignment in 1st section*/
else
    Ln_Rob_B ← element_tmp
    MinLane ← TransitionN (Trans_Cnt,2)+ Ln_Rob_A
    MaxLane ← MinLane + Ln_Rob_B -1
    Ln_Rob_A ← Ln_Rob_A + Ln_Rob_B
    element_tmp ← TransitionLnList (j) /*Making a rotation*/
    TransitionLnList (j) ← Ln_Rob_B
    TransitionList_Min (i,j) ← MinLane
    TransitionList_Max (i,j) ← MaxLane
}
}

```

In order to reduce the occurrence of unsearched areas, an algorithm “**GetMaxMinLnEach** (defined in Table 4.9)” is used to overlap the overlooked cells. Overlooked cells usually occur at a turning point, when the current section has less covered lanes than the previous section. Thus, the “**GetMaxMinLnEach**” algorithm is used to obtain a new list of turning points (“*TransitionList_Partial*”), and the bottom-most (“*TransitionList_Min_A*”) and up-most (“*TransitionList_Max_A*”) lane numbers to make the original turning point move one step further (if there are more covered lanes than in the previous section, then move one step backward) to the overlap the ignored area.

Table 4.9 Algorithms To Get Lane Assignments For Each HMR Vehicle

Algorithm: *GetMaxMinLnEach*

```

for i ← 1 to TotalRob do { /*i is vehicle's order */
    if Robot_No=i then /*Robot_No used to represent a vehicle's identification */
        for j ← 1 to TotalRob do /*j is the order of each section*/
            TransitionList_Partial (j) ← TransitionSecList (j)
            if j >= 2 then
                if TransitionList_Max(j - 1,i) > TransitionList_Max (j,i)
                    then TransitionList_Partial (j) ← TransitionSecList (j) + 1
                elseif TransitionList_Max (j,i) > TransitionList_Max (j - 1,i)

```

Table 4.9 - *Continued*

$then \ TransitionList_Partial(j) \leftarrow TransitionSecList(j) - 1$ $TransitionList_Min_A(j) \leftarrow TransitionList_Min(j,i)$ $TransitionList_Max_A(j) \leftarrow TransitionList_Max(j,i)$ $\}$

After performing the above algorithms, a list of the bottom-most and up-most lane of each section within a transition area is obtained. By following a Boustrophedon path, the mine detection vehicles start from the up-most lane, moving downward (\downarrow) to the bottom-most lane, then moves forward (\rightarrow) to next location, and moves upward (\uparrow) to its up-most lane, continuously. They do this until they reach the end of the transition section, and then change their moving direction to the next up-most (or bottom-most) lane, finishing their task inside a transition area.

Depending on the value of X-axis, the search patterns have been categorized into two parts: $D_Lane=1$ and $D_Lane=0$. The attribute “ D_Lane ” is used to represent the status of the X-axis. Equation (4.6) is used to obtain the value of “ D_Lane ”.

$$D_Lane \leftarrow MOD(N,2) \tag{4.6}$$

“ D_Lane ” is either “1 (meaning ODD N)” or “0 (meaning EVEN N)”. Moving direction “ \downarrow ” is the first search priority, when N is ODD (or \uparrow when N is EVEN). The relative position between the current moving lane and their bottom-most and up-most lane are divided into five relationships (see Table 4.10). Whenever a vehicle goes around the obstacle (or mine), it has to move beyond (or behind) the current “N”. Under this condition, a vehicle may have some cells go un-searched at current “N”, if the current lane is not its boundary. Hence, the attribute $Go_Further$ is defined to let a vehicle know that it has not finished its task in order to prevent the problem of overlooked cells.

As a result, each search pattern above is classified into three subdirectories, based on the value of *Go_Further*. With this setting, whenever a vehicle moves beyond the (defined as $Go_Further > 0$ (or behind ($Go_Further < 0$))) current “N” to skip the current “N” search, it knows it has to move back to the original “N” to continue its task. The relationship between location and moving status (*Go_Further*) is defined in Table 4.10.

Table 4.10 HMR Location Relationship

<i>Lane_Number</i>	<i>D Lane=1</i>			<i>D Lane=0</i>		
	<i>Go_Further</i>			<i>Go_Further</i>		
	= 0	< 0	> 0	= 0	< 0	> 0
Equal to <i>MaxLane</i>	(1)	(2)	(3)	(12)	(12)	(13)
Between <i>MaxLane</i> and <i>MinLane</i>	(1)	(15)	(5)	(14)	(4)	(11)
Equal to <i>MinLane</i>	(6)	(7)	(8)	(14)	(15)	(11)
Higher than	(9)	(2)	(3)	(4)	(4)	(16)
Lower than	(10)	(10)	(17)	(14)	(15)	(11)

The number defined for each relationship in Table 4.10 represents a searching pattern.

These search patterns are listed in Table 4.11.

Table 4.11 HMR Searching Patterns

Priority\Pattern	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	↓	↘	←	↘	←	→	→	↖	↓	↗	←	→	←	↑	↗	↙	↖
2	↘	→	↙	→	↖	↗	↗	←	↘	↑	↙	↘	↙	↗	→	←	←
3	↙	↗	↖	↓	↑	↑	↑	↑	↙	→	↖	↗	↖	↖	↑	↓	↙
4	→	↓	↓	↗	↙	↘	↘	↙	↑	↗	↑	↓	↓	→	↘	↖	↑
5	←	↑	↑	↓	↓	↖	↓	↓	→	↓	↓	↑	↑	←	↓	↑	↓
6	↗	↙	↘	↙	↗	←	↖	→	↗	↖	↗	↙	→	↘	↖	↘	↗
7	↖	←	→	←	→	↓	←	↗	←	←	→	←	↘	↙	←	→	→
8	↑	↖	↗	↖	↘	↙	↖	↘	↙	↘	↘	↖	↗	↓	↙	↘	↘
9	↙	↙	↙	↙	←	↖	↖	←	↙	↖	↙	↙	↙	↖	↖	←	←
10	←	←	←	←	↖	←	←	↙	←	←	↖	←	↖	←	←	↖	↙
11	↖	↖		↖		↙	↙		↖	↙		↖		↙	↙		

Algorithm *ScanningWayNoChg*(defined in Table 4.12) is defined to determine the location relationship, in order to apply a search pattern to the vehicle. Algorithm *SearchPattern*(defined in Table 4.13) is defined to obtain an available moving direction for a vehicle. If the first direction of the search pattern contains an obstacle (or mine), it scans the next direction to find a clear path to move further. Usually, if an obstacle (or mine) is found when moving in \downarrow direction, it scans either \swarrow or \searrow to go around it. It scans \nwarrow or \nearrow , if it is moving \uparrow . If a vehicle does not reach its target boundary, but has to go around an obstacle (or mine) to next the “N”, the attribute “*Go_Further*” is triggered. However, if this obstacle (or mine) is on its searching target boundary, (attributes “*Min_Reached*” and “*Max_Reached*” are used to determine if the vehicle has reached its bottom-most (or up-most) lane and obstacle (or mine) is also on this boundary), the value of attribute “*Go_Further*” will be reset to zero. This situation means the vehicle has finished its search task at current “N” and is able to move further to the next “N”. Therefore, attribute “*Go_Further_Chg=1*” is triggered (in search algorithm *ScanningWayNoChg*) to make the vehicle re-apply the search procedure and move further (see the rules in Table 4.14).

Therefore, the vehicle can skip scanning backward directions, such as \leftarrow , \swarrow and \nwarrow , to the next available moving direction. Several directions have been highlighted in red in Table 4.11 to indicate that they could be skipped in *SearchPattern*. However, these directions have also been appended into Table 4.11(have been highlighted in blue) in order to prevent the vehicle from becoming trapped inside the current cell, if this skipped direction is the only way out of the current location.

Table 4.12 Algorithm To Determine A HMR Search Pattern

Algorithm: *ScanningWayNoChg*

```

/* MinLane ← TransitionList_Min_A (Index_N_Sec)*/
/* MaxLane ← TransitionList_Max_A (Index_N_Sec)*/
if D_Lane = 1 then
    if Lane_Number =MaxLane then
        if Go_Further = 0
            then SearchPattern (1)
        else if Go_Further < 0
            then SearchPattern (2)
        else if .....
    else if Lane_Number < MaxLane AND Lane_Number > MinLane then
        if Go_Further = 0
            then SearchPattern (1)
        .....
    else if Lane_Number =MinLane then
        .....
    else if Lane_Number >MaxLane then
        .....
    else if Lane_Number <MinLane then
        .....
else then
    if Lane_Number =MaxLane then
        if Go_Further = 0
            then SearchPattern (12)
        else if Go_Further < 0
            then SearchPattern (12)
        else if
            .....

```

Table 4.13 Algorithm To Apply A HMR Search Pattern

Algorithm: *SearchPattern (Pattern)*

```

if Pattern= 1/*The order for each pattern is defined in Table 4.11*/
    for a← 1 to Search_No do {
        if a=1 /*pos is the direction value that is defined in Table 3.1*/
            then ToPosAssigned (pos)
        else if a=2
            .....
            .....
        else if a=3
            .....

```

Table 4.13 - *Continued*

```

    MineSearching ()
    returnPoint (NTA,LTA)
    elseif Pattern=2
        .....
}

```

Table 4.14 HMR Search Algorithms

Algorithm: *ScanningWay*

```

Go_Further_Chg ← 0
ScanningWayNoChg()
if Go_Further_Chg = 1 then
    if (D_Lane = 1 and Max_Reached = 1) or (D_Lane = 0 and Min_Reached = 1)
    then
        ScanningWayNoChg()
    Go_Further_Chg = 0

```

Beyond the search algorithms above, the following algorithms are defined to cover the remaining task of a destroyed vehicle. The solution is to reassign their task, if they are notified that one of the vehicles has been destroyed, by making the other vehicles move to this location to complete the search. The lane reassignment procedure (similar to *GetMaxMinLn*) is triggered to reassign the task, whenever one of the vehicles is destroyed. At that location “N” where the vehicle is destroyed, *GetMaxMinLnEach* procedure is performed to get a new assignment for all of the surviving vehicles. Table 4.15 summarizes the reassigning procedure.

Table 4.15 HMR Reassign Search Task Algorithms

Algorithm: *ReAssignSearchingArea*

```

/* KeepRobNo is used to store # of vehicle before one of the vehicle destroyed*/
/* TotalRob is used to store # of vehicle before after one of the vehicle destroyed*/
if KeepRobNo <> TotalRob then
    if Robot_No > Explode_NO then
        Robot_No ← Robot_No - 1
        /*Used to get a new ID if its ID is bigger than the destroyed one's, in order to
        get an appropriate assignment*/
        GetMaxMinLnEach ()

```

Table 4.15 - Continued

```

Index_N_Sec ← 1
if N ≥ TransitionList_Partial (Index_N_Sec) then
/*Here is used to determine if it is ahead than the vehicle destroyed location. If
it is ahead, then it has to move back to support the task*/
Back_Explode ← 1
MinLane ← TransitionList_Min_A (Index_N_Sec)
MaxLane ← TransitionList_Max_A (Index_N_Sec)
Index_N_Sec ← Index_N_Sec + 1

```

The above HMR search procedure for a Coordinated Team is shown in figure 4.8. The simulation result (see figure 4.9) shows that the above overlapping and reassignment algorithms are effective in solving the un-searched problems and also handling the allocation of multiple search vehicles to balance the search task of each vehicle.

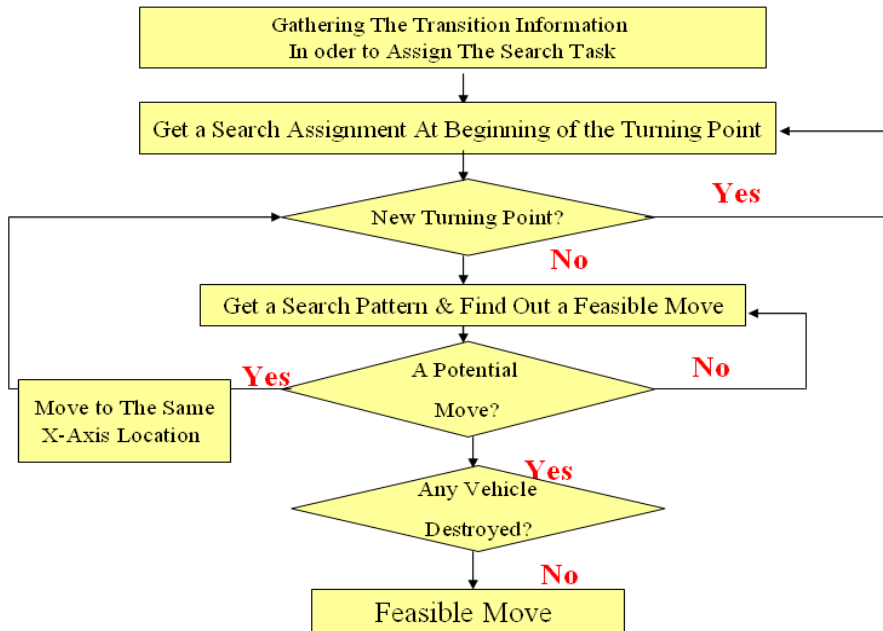


Figure 4.8 A HMR search procedure

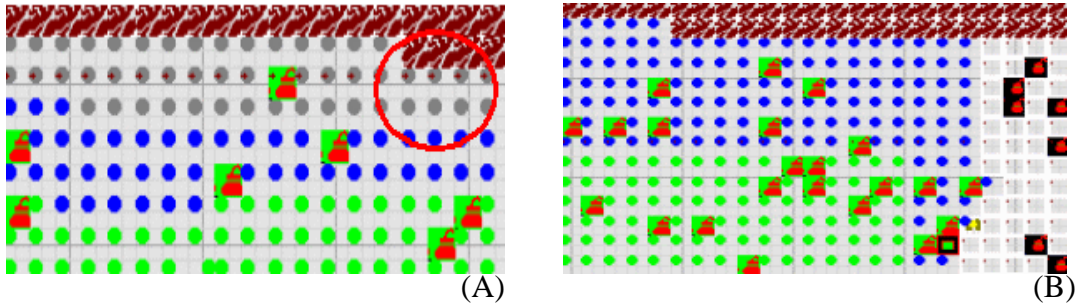


Figure 4.9 (A) A HMR overlap strategy improves the percentage of the area (B) The survived vehicle supports the un-finished area that is left by the destroyed vehicle

4.3 Point-to-Point Wave

The concept of a point-to-point (P2P) sweep was to incorporate the use of slower but more discriminating mine sensor technologies that might be too slow to support 100% coverage in MMB or HMR scenarios, but would be highly effective at a specific detection task, like the elimination of false positive mine detections.

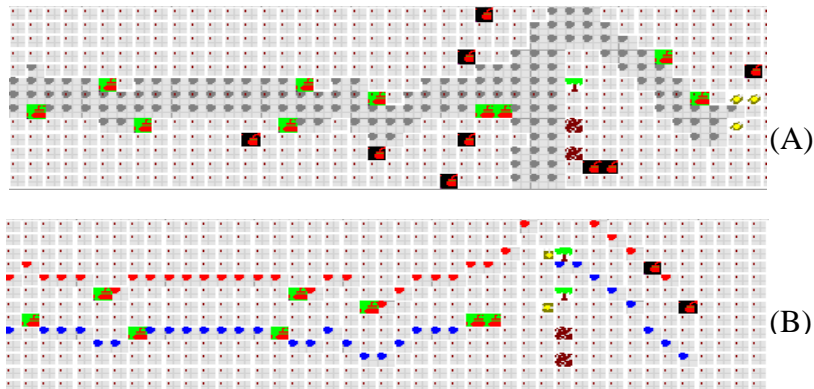


Figure 4.10 (A) The three vehicle breaching team runs first providing 100% coverage for the breach lane, (B) a two-vehicle P2P team starts to work right after the Multi-Vehicle Wave begins its search.

The Multi-Vehicle Wave simulation (e.g. figure 4.10(A)) will determine the locations that have a high probability of containing a mine. These locations will be recorded within the larger mine detection system (this model is recorded in the variable

Table *MineTable*). The P2P wave can then extract these locations from the system to determine where more discriminating sensor reading should be taken.

In addition, the Multi-Vehicle Wave simulation will generate the boundary information to make the Point-To-Point simulation restrict the motion of the point-to-point vehicle(s) to within the marked boundary of the searched lane (figure 4.10(B)). It will do this without crossing the boundary, even if there is a shorter path between two points. A boundary table, *BunTable*, contains all the points of the boundaries that are generated. In above waves, the top-most vehicle's path will determine the upper boundary, and bottom-most vehicle's path will determine the lower boundary (e.g. $BunTable(1,N) \leftarrow$ bottom-most vehicle's lane number and $BunTable(3,N) \leftarrow$ top-most vehicle's path). Whenever the P2P vehicle moves, it can check *BunTable* to confirm if the next moving location is beyond the boundary or not. If the vehicle tries to move across the boundary, a warning will be given to ask it try another location.

The P2P wave can be modeled as a Vehicle Routing Problem (VRP) in which a set of vehicles visit customers exactly once per service cycle (Shaw 1998). Within the MiCAT tool, this research has modeled a number of vehicle/mine location allocation strategies. The first one is to require that the previous mine detection waves complete their searches before making the P2P vehicle assignments. A number of conventional VRP analytical optimization techniques assume that the set of route locations have been fully defined prior to the start of the assignment of these locations to the vehicle set. The Variable *Transation_Achieved_Rot* is defined in both the Multi-Vehicle Wave and the Point-To-Point wave. Whenever the previous mine detection waves complete their

searches, *Transation_Achieved_Rot* is set to “one” and the P2P vehicle team is allowed to start its searching.

The other strategy is to allow the P2P vehicle team to start working right after the Multi-Vehicle Wave begins its search. Under this strategy, the P2P vehicle team will only be a few steps behind the previous wave. Simple load balancing techniques that handle the dynamic definition of route locations are defined in this strategy. When the vehicles are allowed to start their search before the previous wave has completed, the *Transation_Achieved_Rot* variable is set to “two” in order to release the vehicles in the P2P wave. This setting will be made even if there is no potential mine found in the prior wave. If the previous wave vehicles have found one potential mine, they reset *Transation_Achieved_Rot* to “one” to make the P2P vehicle move further to the potential landmine location, even if the distance between both waves is less than required. If the P2P vehicle finds that its location is close enough to the previous wave vehicle, the value of *Transation_Achieved_Rot* is set to “zero” and it stops.

If the P2P vehicle assignments have to be fully defined prior to the P2P vehicle team is allowed to start its searching, the possible landmine locations, detected by the previous wave, can be sorted in sequence by WITNESS® built-in function *SortVar* (see Table 4.16). In order to equally allocate the possible landmine locations to *Rob_MineTable_i* (a list of the next indented moving target for vehicle *i*), Table 4.16 divided the total number of vehicles to get an equal number of tasks. In order to obtain the vehicle’s assignment, Table 4.16 assigns the vehicle’s identification number to *MineTable*. If the remainder is greater than zero, the remaining landmines will be

assigned to the preceding vehicles. After that, Table 4.17 uses this ID to allocate the detection task to each P2P vehicle. It is used to check if that detected landmine is assigned to that vehicle or not.

Table 4.16 Algorithms To Allocate The Potential Landmines

Algorithm: *DivideArea_a()*

```

SortVar (MineTable,2,1)
Mine_ind ← total number of potential landmines
TotalRob ← total number of P2P vehicles
Mine_Assignment ← ((Mine_ind - 1) / TotalRob) /*Get an equal detection task*/
Mine_Assignment01 ← Mine_Assignment
Mine_Assignment_Remain ← MOD (Mine_ind - 1, TotalRob) /*the remainder*/
Mine_ind01 ← 1 /* Initialized by 1st vehicle's ID*/
for K ← 1 to Mine_ind do {
    if K < Mine_Assignment01 then
        MineTable01 (3,K) ← Mine_ind01 /*To sort the landmine*/
        K ← K+1
    else if K = Mine_Assignment01 then
        MineTable01 (3,K) ← Mine_ind01
        K ← K+1
        if Mine_Assignment_Remain > 0
            MineTable01 (3,K) ← Mine_ind01
            K = K + 1
            Mine_Assignment_Remain ← Mine_Assignment_Remain - 1
            MineAsigRem_Ind ← 1
        if MineAsigRem_Ind = 1
            Mine_Assignment01 ← Mine_Assignment01 + Mine_Assignment + 1
        else
            Mine_Assignment01 = Mine_Assignment01 + Mine_Assignment
        /*Start to assign the task to the next vehicle*/
        Mine_ind01 ← Mine_ind01 + 1
}

```

Table 4.17 Algorithms To Distribute The Potential Landmines To P2P Vehicles

Algorithm: *DivideArea_b()*

```

Mine_ind01 ← 1 /* Initialized by 1st vehicle's ID*/
QW1 ← 1 /* Initialized by 1st mine for 1st vehicle, etc....*/
.....
for K ← 1 to Mine_ind do {
    if Mine_ind01 = 1

```

Table 4.17 - Continued

```

Rob_MineTable_1 (1,QW1) ← MineTable01 (1,K)
Rob_MineTable_1 (2,QW1) ← MineTable01 (2,K)
if MineTable01 (3,K) <> MineTable01 (3,K + 1)
    /* The task for the next vehicle*/
    Mine_ind01 ← Mine_ind01 + 1
else QW1 = QW1 + 1
elseif Mine_ind01 = 2
    .....
K ← K + 1
}

```

Depending on the value of the X-axis of the vehicle's location, the search patterns have been categorized into three parts: A, B, and C. Each part is categorized into three parts based on the value of the Y-axis of the vehicle's location. The relationship of the vehicle's location is defined in Table 4.18. Each relationship is defined as a searching pattern. An algorithm is used to get a relationship between the current location and the next target location (i.e. *Rob_MineTable_i*). In Table 4.19, an appropriate searching pattern is also applied, depending on the obtained relationship (e.g. A-1, A-2,...etc.). The algorithm in Table 4.20 is used to obtain the next intended moving step and is applied whenever the vehicle has been assigned a searching pattern. It tries the first direction to see if it is available to move there. If the first location of the search pattern contains an obstacle (or mine), it scans the next location to find a clear path to move further. If it is clear, (NTA,LTA) will be returned as the moving location. While the vehicle performs the mine or obstacle algorithm, it also checks to see if it reaches its target or not. If it already reaches its current target, the next potential landmine location in *Rob_MineTable_i* is assigned as its target. The above P2P mine allocation and search procedures are defined in the flowchart (see figure 4.11).

Table 4.18 Point To Point Searching Patterns

Interval	A			B		C		
	1	2	3	1	2	1	2	3
1	↘	↗	→	↓	↑	↙	↖	←
2	→	→	↗	↙	↖	←	↙	↖
3	↓	↑	↘	←	←	↓	←	↙
4	↗	↘	↑	↖	↙	↖	↑	↑
5	↑	↓	↓	↑	↓	↑	↓	↓
6	↙	↖	↖	↘	↗	↘	↗	↗
7	↖	←	↙	↗	↘	→	→	↘
8	←	↙	←	→	→	↗	↘	→

Table 4.19 Algorithms To Obtain A P2P Searching Patterns

Algorithm: *GetSearchPattern* ()

```

if Current  $X_i$  < target  $X_j$  then
    if Current  $Y_i$  is > target  $Y_j$  then
        pattern ← A-1
    else if Current  $Y_i$  is < target  $Y_j$  then
        pattern ← A-2
    else if Current  $Y_i$  is = target  $Y_j$  then
        pattern ← A-3
else if Current  $X_i$  = target  $X_j$  then
    if Current  $Y_i$  is > target  $Y_j$  then
        pattern ← B-1
    else if Current  $Y_i$  is < target  $Y_j$  then
        pattern ← B-2
else if Current  $X_i$  > target  $X_j$  then
    if Current  $Y_i$  is > target  $Y_j$  then
        pattern ← C-1
    else if Current  $Y_i$  is < target  $Y_j$  then
        pattern ← C-2
    else if Current  $Y_i$  is = target  $Y_j$  then
        pattern ← C-3
return (pattern)

```

Table 4.20 Algorithms To Obtain P2P Next Step

Algorithm: *performSearchPattern* (*searchPattern*)

*/*The order for each pattern is defined in Table 4.18*/*

```

if searchPattern=A-1
    for a ← 1 to 8 do {

```

Table 4.20 - *Continued*

```

if a=1 then
    ToPosAssigned (pos)
else if a=2
    ....
    ....
else if a=3
    ....
    ....
MineSearching_P2P ()
returnPoint (NTA,LTA)
}
elseif searchPattern=A-2
    .....

```

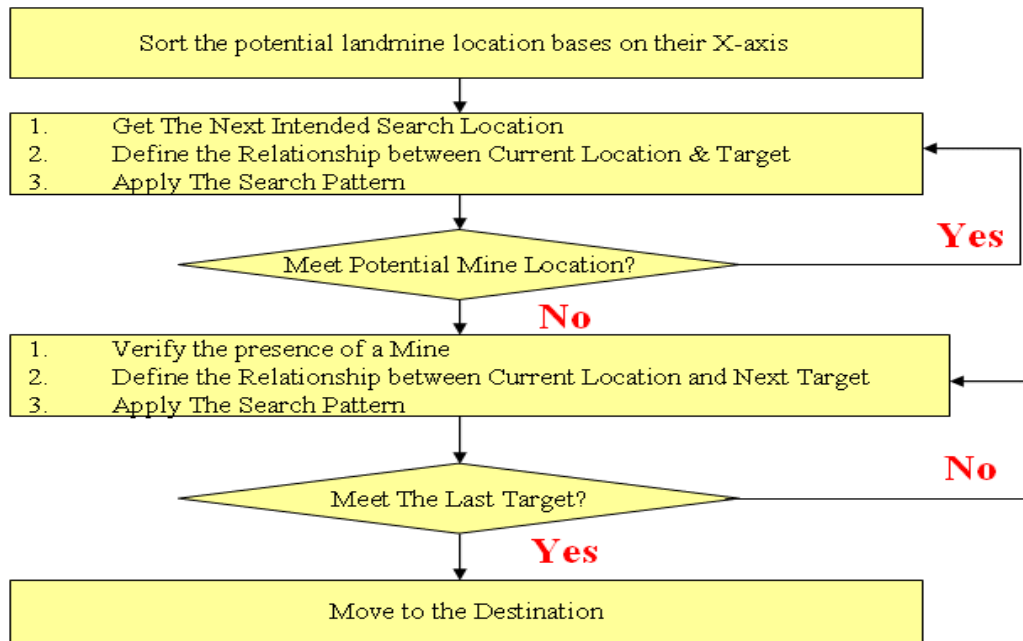


Figure 4.11 The P2P search flowchart

CHAPTER 5

DISCUSSION OF MODEL GENERALIZATION

Up to this point, all the simulation models have been designed for a pre-determined minefield size and specific number of search vehicles. The minefield itself has been modeled as a pre-defined grid with the assumption that the search vehicle occupies only one grid location at a given time. This chapter will evaluate the potential of developing parameterized algorithms that will allow the MiCAT user to dynamically specify the characteristics of a mine detection problem without having to modify the underlying simulation code that will drive the MiCAT system. This paper will discuss the relative feasibility of developing these parametric algorithms.

5.1 Adjustable Grid Scaling and Shifting The Orientation of The Search Grid

This section investigates the potential of supporting adjustable grid scaling and the orientation shifting of the search grid to increase the flexibility of MiCAT system. These features would enable better modeling of non-rectangular search areas and greater flexibility in breach path direction control. The idea to model a non-rectangular search area and to make the vehicle able to move in more than just a straightforward direction is to make the adjustment of minefield orientation become more flexible. As this paper mentioned earlier, this study assumes a non-rectangular search area that can

be decomposed into lanes and cells. Therefore, if the minefield is adjusted with different degree turns, it will look like a zigzag lane (see figure 5.1).

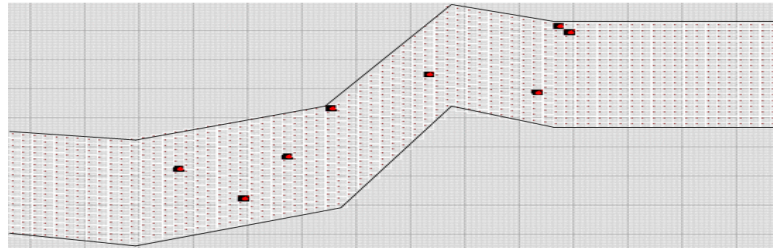


Figure 5.1 A ZigZag shape minefield area

In order to support the capability of adjustable grid scaling and the orientation shifting of the search grid, this study provides the users a input function to place the minefield elements to the desired X-Y coordinate. MiCAT applies the SETPOSN function of WITNESS® to specify the X and Y co-ordinates of the minefield elements. The SET Icon function is then used to adjust the minefield element to its appropriate size. Finally, the SET ICON action is used to reset the model. Therefore, MiCAT is able to shift the orientation of the search grid automatically to correspond to the user's minefield specification.

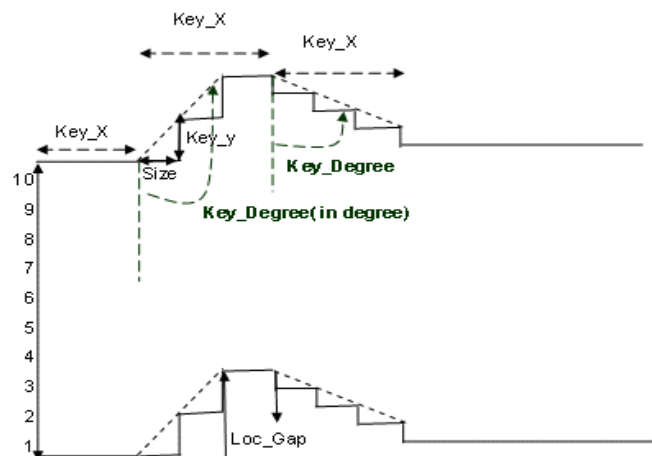


Figure 5.2 An illustration of the shifting of minefield orientation

Figure 5.2 illustrates how to shift the minefield elements to represent the orientation of the search grid at each minefield transition. In figure 5.2, the width of the cell is defined as “*Size*” and “*Size*” is the same height of the lane. The degree shifting of the orientation is defined as “*Key_Degree*”. In order to create a “*Key_Degree*”, a distance of “*Key_y*” units is shifted along the Y-coordinates for one unit shifts in the X-coordinate (*Key_y_prev* is defined to express this distance in the previous section). If *Key_Degree*>90, the orientation shifts upward simultaneously. If *Key_Degree*<90, the orientation shifts downward. Because the Y-coordinates in WITNESS® increases from top to bottom and is opposite to the X-Y coordinates, the shifting direction is opposite to “*Key_y*” (e.g. *Key_Degree*>90, *Key_y* <0). Because *Size* in the X-coordinate direction is fixed, MiCAT utilizes it to estimate *Key_y*. Tangent and *Key_Degree* are also used. Where WITNESS® does not support tangent, SIN and COS are used instead (see equation 5.1). In addition, the specified angle of SIN and COS in WITNESS® must be specified in radians (see equation 5.2). The SETPOSN function has to receive integer X and Y parameters, thus the TRUNC function is used to truncate the real number to an integer (see equation 5.3).

$$\text{Tan} = \text{SIN} (3.14157 * \text{Key_Degree} / 180) / \text{COS} (3.14157 * \text{Key_Degree} / 180) \quad (5.1)$$

$$\text{Key_y_real} = \text{Size} / \text{Tan} \quad (5.2)$$

$$\text{Key_y} = \text{TRUNC} (\text{Key_y_real}) \quad (5.3)$$

Different transitions’ orientations shifted make different inclinations and gaps between transition points along the Y-coordinate (defined as *Loc_Gap*, see figure 5.2). The Y-coordinate of a cell also depends on the order of its lane. In order to estimate the

Y-coordinate, (Pos_X, Pos_Y) is used to represent the starting point of the minefield coordinates. Loc_Y_df is then used to indicate the order of a lane, and Key_x is defined to express the distance between the current and the last transition points (Key_x_prev is defined to express this distance in the previous section). There are three possible cases for the shifting orientations.

- Case 1: The previous transition is horizontal. In this case, there is no gap between the current and the last transition points. Thus, the lane Y-coordinate can be expressed as equation 5.4.
- Case 2: The previous transition is not horizontal but the current transition is horizontal, therefore, a gap occurs. The gap depends on the width of the last transition. Equation 5.5 expresses this gap. Thus, the Y-coordinate is estimated by adding the gap height to the original Y-coordinate (see the expression in equation 5.6).
- Case 3: Both the previous and current transitions are not horizontal. The specific Y-coordinate within a gradient transition depends on how far out it goes on the X-coordinate. $QW1$ is used to indicate the cell order counting from the first cell of a transition along the X-coordinate. Therefore, the Y-coordinate can be expressed in equation 5.7.

$$Loc_Y = Pos_Y + Size * Loc_Y_df \quad (5.4)$$

$$Loc_Gap = Loc_Gap + Key_y_Prev * (Key_x - Key_x_Prev) \quad (5.5)$$

$$Loc_Y = Pos_Y + Size * Loc_Y_df + Loc_Gap \quad (5.6)$$

$$Loc_Y = Pos_Y + Size * Loc_Y_df + Key_y * QW1 + Loc_Gap \quad (5.7)$$

Algorithm *MapBuilder* summarizes the above logic and is used to adjust the grid scaling as well as the orientation shifting of the search grid. Algorithm *MapResting* then places the minefield element to the specific location by using the functions *SETPOSN* and *SET ICON*.

Table 5.1 Algorithms To Reset The Minefield

Algorithm: *MapBuilder(Pos_X, Pos_Y,Size)*

```

Loc_Y_df = 0
Loc_Lan = MaxLane
for i ← 1 to MaxLane do {
/*The map is built from the further most lane (top one) to nearest lane, because the
Y-coordinate is from top to bottom*/
Key_x ← 1
ind ← 1
Key_x_Prev ← 0
Key_Degree_Prev ← 0
Key_QW_Prev ← 0
Loc_Gap ← 0
While Key_x > 0 /*This is used to stop the procedure*/{
Key_x ← Transition (ind,1)
Key_Degree ← Transition (ind,2)
if Key_Degree <> 0
Tan ← SIN(3.14*Key_Degree/180) / COS (3.14 * Key_Degree/180)
Key_y_real ← Size / Value_Tan
Key_y ← TRUNC (Key_y_real)
Key_QW ← Key_x
if Key_Degree_Prev <> 0
if Key_ind = 2
Loc_Gap ← Loc_Gap + Key_y_Prev * Key_x_Prev
else
Loc_Gap ← Loc_Gap + Key_y_Prev * (Key_x_Prev - Transition(ind-2, 1))
for QW ← 1 to Key_QW do {
/*This loop is used to place the cells in an order*/
Loc_X ← Pos_X + Size * QW
if Key_Degree_Prev <> Key_Degree and Key_Degree <> 0
/*Used to distinguish the transition degree between different transition*/
QW1 ← QW - Key_QW_Prev
Loc_Y ← Pos_Y + Size * Loc_Y_df + Key_y * QW1 + Loc_Gap
else
Loc_Y ← Pos_Y + Size * Loc_Y_df + Loc_Gap

```

Table 5.1 - Continued

```

MapResting (Loc_Lan,QW,Loc_X,Loc_Y,20)
  QW  $\leftarrow$  QW + 1
}
/*Save the current transition information for the comparison*/
Key_x_Prev  $\leftarrow$  Key_x
Key_y_Prev  $\leftarrow$  Key_y
Key_QW_Prev  $\leftarrow$  Key_QW
Key_Degree_Prev  $\leftarrow$  Key_Degree
ind  $\leftarrow$  ind + 1
}
Loc_Lan  $\leftarrow$  Loc_Lan - 1
Loc_Y_df  $\leftarrow$  Loc_Y_df + 1
}

```

Above algorithms can represent the transition’s angle change for a small piece of a simulated minefield. However, these algorithms still need improvements. For example, when a transition slope gets steeper, the minefield representation becomes less accurate and a sliding displacement (or fracture) will be produced (figure 5.3). The Boustrophedon path search in the above search algorithms did not consider a turning around radius that is able to expand the minefield search to a larger area, whenever the vehicles finish one piece of small minefield and need to go around one piece to another (see figure 5.5). The formal development and prototype implementation of simulation models based on these problems are left as future work.

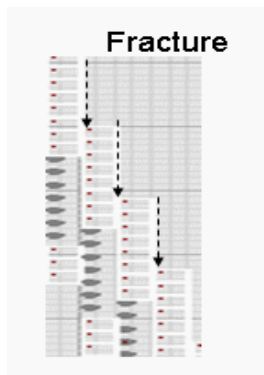


Figure 5.3 A steep slop produces a sliding displacement or a fracture

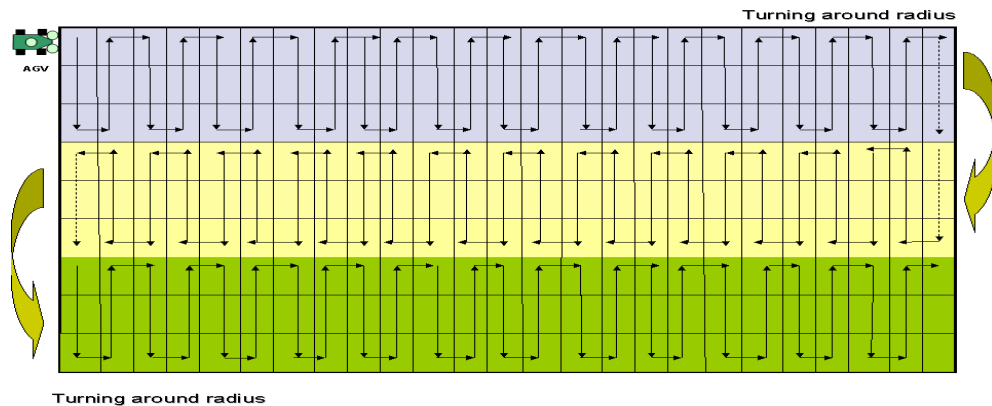


Figure 5.4 The required turning around radius from going around one piece of the minefield to get another

5.2 The Modeling of Vehicles of Different Sizes

This section will investigate the modeling of vehicles of different sizes to increase the flexibility of mine detection tasks. This section uses the SET ICON action of WITNESS® as a virtual part of vehicle to resize the vehicle. At the beginning of the model, Variable “*int_CoveredL*” is defined as the size measured from center to one edge of the vehicle. With this information whenever the vehicle element moves to a new minefield grid location, it draws “*int_CoveredL*” units on the minefield grid locations laterally besides itself (Table 5.2 represents this drawing procedure).

In addition, it has to verify its current minefield grid and the grids laterally beside the center of the vehicle for an obstacle and a mine. The obstacle detection algorithm that has been discussed in the previous section is able to detect an obstacle out to the maximum permissible lane width. However, mine detection sensors only search the ground directly in front of itself. An algorithm *MineSearchingLoc* is defined to run *MineSearching* for $(2*int_CoveredL + 1)$ times within the model, in order to

verify a clear area in front of the vehicle itself. Furthermore, whenever the vehicle element moves toward the next location, these virtual vehicle parts will be erased from the simulated minefield. The ICON is reset to “one” and the matrix of minefield of the virtual vehicle part locations are reset to “zero”. The above algorithms have appended to the Scout model to make it able to specify a flexible size of a vehicle. Figure 5.5 shows two independent vehicles of different size running on the minefield.

Table 5.2 Algorithms To Resize The Vehicle

Algorithm: *DrawVirtualElements*

```

Ary_j ← 2 * int_CoveredL + 1
/*The draw procedure depends on vehicle's size which is 2*int_CoveredL+1*/
int_Ary_RY ← int_CoveredL + 1
/*Define the location of vehicle's center part */
for i ← 1 to Ary_j do {
    LTA_C ← Lane_Number + int_Ary_RY - i
    /*LTA_C is to represent the relative location of vehicle's center part */
    NTA_C ← N
    if LTA_C > 0 AND NTA_C > 0
        SET ICON of MatrixMap (LTA_C, NTA_C) to ICON
        MatrixMap (LTA_C, NTA_C) AT 0:Obj_Trap_Mine_Rob_VO_M_B (4) = 1
}

```

Table 5.3 Algorithms To Search The Entire Grids In Front of A Vehicle

Algorithm: *MineSearchingLoc()*

```

Count ← 0
Ary_j ← 2 * int_CoveredL + 1
/*The vehicle has to scan all of the minefield elements*/
int_Ary_RY ← int_CoveredL + 1
for i ← 1 to Ary_j do {
    MineSearchCntTmp ← MineSearching (int_Ary_RY - i)
    Count ← MineSearchCntTmp + Count
}
Return Count

```

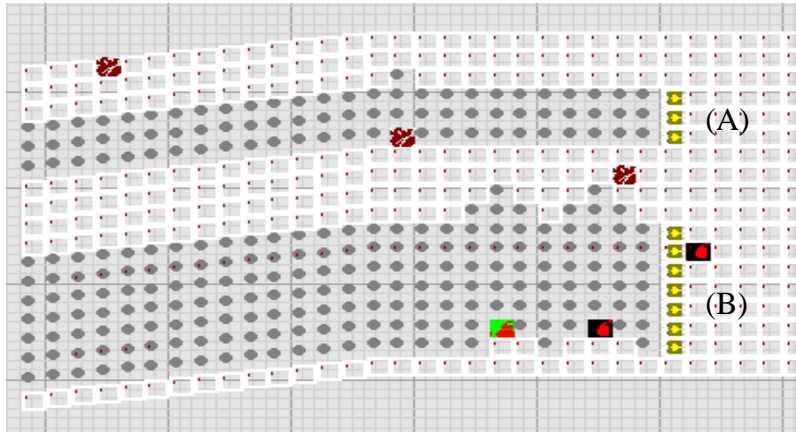


Figure 5.5 Two independent vehicles of different sizes (A) three units size and (B) seven units size

5.3 Alternative Vehicle Formations

In reality, two vehicles traveling side-by-side in a non-contact formation leave an unsearched gap between themselves. In order to cover the gap, this section investigates an alternative staggered vehicle formation for military minefield breaching (MMB). This staggered vehicle formation is to start the first two vehicles together followed by a third one that moves behind on a path between these two vehicles (see figure 5.6). Thus, the rear one can cover the unsearched gap.

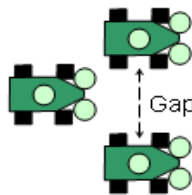


Figure 5.6 A staggered vehicle formation

Two types of vehicles are investigated in this section: independent and dependent. The dependent vehicle team has to maintain the same moving direction to

hold a breaching path, but the independent vehicle team will not need to follow the other vehicles' moving direction.

Although the independent vehicle team does not to move in same direction, the team's formation is maintained by controlling the steps difference between the front vehicles in one step and the steps differences between front and rear in less than five steps. Variable *MoveStatus_i* (*i* stands for the vehicle's number) is used to control WITNESS®'s PUSH or WAIT action to make the vehicles perform the appropriate movement (see Table 5.4). Whenever the vehicle moves apart from the others more than the steps allowed, they will wait for the others until the ones behind have moved ahead.

Table 5.4 Staggered Formation Movement Control Algorithms

Algorithm: *VehicleMovementControl* in "To" Action

i ← 1 to Number of Vehicles
if *RobotMoveStatus_i*(1)= 1
 /*The vehicle has to wait if *MoveStatus* =1*/
STOP ← *RobotNameMoveStatus_i* (2)
if (*N* ≥ *STOP* and *i* < *last vehicle*) or (*N* ≥ *STOP* + 5 and *i* = *last vehicle*)
 /*Reset *MoveStatus_i*(1) to zero if it has moved ahead the one waiting for it*/
RobotNameMoveStatus_i(2) ← 0

The complexities increase for the dependent vehicle team to maintain the same moving direction to hold a breaching path. This section reduces the complexities by appending the virtual obstacle (or landmine) strategy to the above independent vehicle obstacles (or landmine) avoiding algorithms. In order to express the virtual obstacle (or landmine) in the system, attribute *Virtual_No* is used. It is represented as the vehicle's second id. Its value is equal to the vehicle's assigned lane. When one of the vehicles

find an obstacle (or landmine), it marks the locations that are related to the found obstacle position as a virtual obstacle (or landmine) and updates their value with a *Virtual_No*. For example, if the vehicle found an obstacle (or landmine) at (X,Y) , it marks and updates the values of location $(X,Y-2)$ and $(X,Y-4)$ with *Virtual_No* as virtual obstacle locations (figure 5.7).

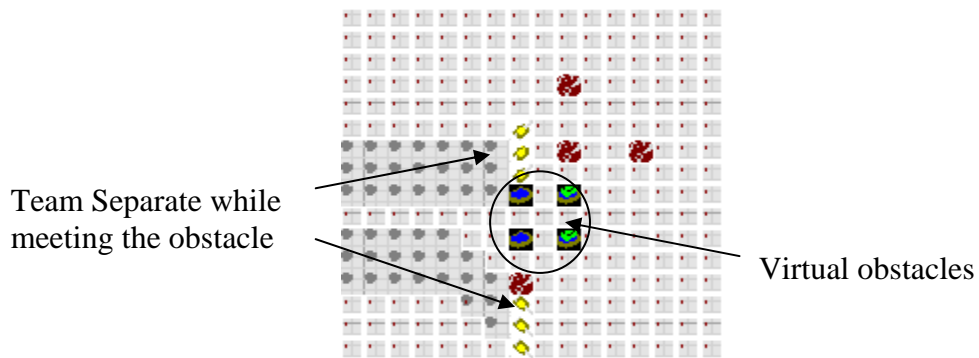


Figure 5.7 Setting a virtual obstacle

Whenever the obstacle blocks the path between the vehicle team, the vehicle team members can make a turn at the virtual obstacle locations. However, the team members shift in different directions (figure 5.7). The purpose of this section is to maintain a path and cover that unsearched gap. The solution is to use various attributes (or variables) to control the vehicle to move in the same direction. This section uses the variable *MidLaneAssign* to let the vehicle determine its position that related to the center vehicle. Whenever the vehicle finds a virtual obstacle on its path, it gets its virtual obstacle value and posts this value to variable *Obj_Target_v*. Therefore, the algorithms in Table 5.5 utilize this information to make the front vehicles go around the obstacle on the same side of the obstacle. The rear one follows the same direction as the

one that found the obstacle. The result is shown in figure 5.8 and it shows that the team can hold a breaching path while avoiding an obstacle.

Table 5.5 Staggered Formation Moving Direction Control Algorithms

Algorithm: *MovingDirectionControl*

```

if Lane_Assignment > MidLaneAssign then
  if ObjDetecAry_R1_Space (i,1) < ObjDetecAry_R1_Space (i,2) then
    int_go_up ← 0
    int_go_down ← 1
  else
    int_go_up ← 1
    int_go_down ← 0
elseif Lane_Assignment < MidLaneAssign then
  if ObjDetecAry_R1_Space (i,1) > ObjDetecAry_R1_Space (i,2) then
    int_go_up ← 1
    int_go_down ← 0
  else
    int_go_up ← 0
    int_go_down ← 1
else
  if Obj_Target_v < Virtual_No then

```

*/*If the vehicle detects there is a virtual obstacle on its path and is smaller than its Virtual_No, it means the virtual obstacle is post by the lower vehicle. In this case, the vehicle will follow the same as the lower vehicle's direction*/*

```

  if ObjDetecAry_R1_Space (i,1) > ObjDetecAry_R1_Space (i,2) then
    int_go_up = 1
    int_go_down = 0
  else
    int_go_up = 0
    int_go_down = 1
else
  if ObjDetecAry_R1_Space (i,1) < ObjDetecAry_R1_Space (i,2) then
    int_go_up = 0
    int_go_down = 1
  else
    int_go_up = 1
    int_go_down = 0

```

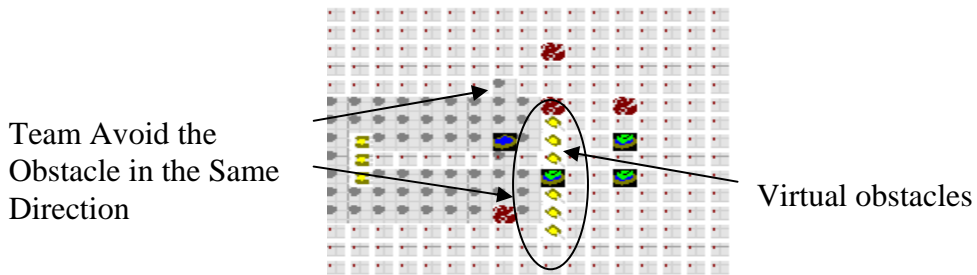


Figure 5.8 The vehicle team moves in the same direction to hold a breaching path

Other than the above separating problems, the preliminary simulation result shows that the team is not able to maintain a breaching path whenever the team found a landmine (figure 5.9). The previous chapter has developed an algorithm that the center vehicle will consider the turning radius to move several steps back to a place where it is able to lead the team to go around the landmine. Whenever the vehicle needs to go around the landmine it has to move several steps back. Thus, if the vehicle does not know it has to go around a virtual landmine before the other one has found it, it will not make a backward movement in the same direction as the one that found the landmine.

This section has made the vehicle perform the detection procedure earlier than the moving direction decision procedure. The previous *ScanningWay* algorithm in section 4.2.1 has been expanded to two cycles, *ScanningWay_cy1* and *ScanningWay_cy4*, with different times. The first cycle makes the vehicle scan the minefield. The second cycle makes the vehicle decides the movement. In first cycle, the one that found a landmine will also post the virtual landmine information to the center. Thus, the other vehicles have enough time and virtual landmine information to decide to move back (figure 5.10) to go around the landmine in the second cycle. The simulation

results (figure 5.11) shows the virtual obstacle (landmine) and time control algorithms have made the vehicle team, retaining a staggered formation, able to maintain a breaching path, and go around the virtual obstacle (or landmine).

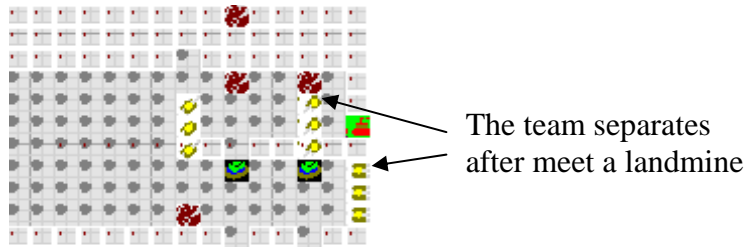


Figure 5.9 The vehicle team separates while meeting a landmine

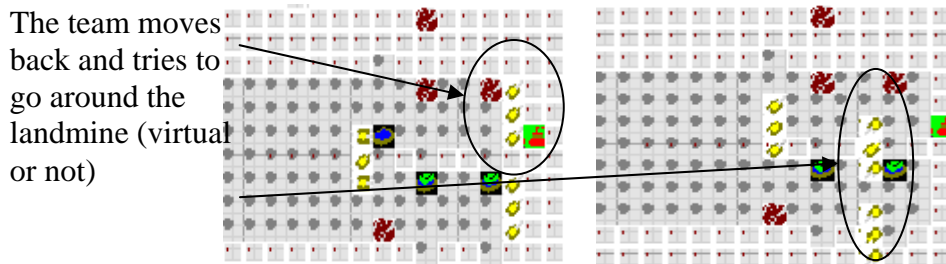


Figure 5.10 The time control algorithm has made the vehicle able to move back while meeting a landmine

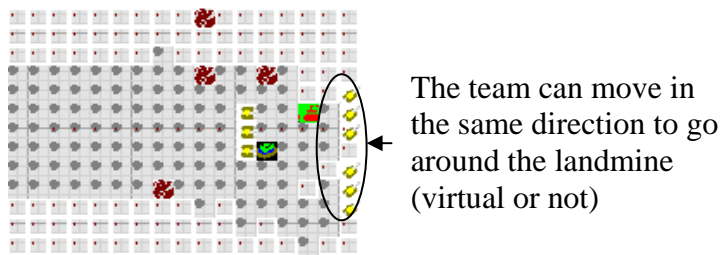


Figure 5.11 The time control algorithm has made the vehicle able to go around the virtual landmine

CHAPTER 6

ANALYSIS OF EXPERIMENTAL RESULTS AND DISCUSSION

This chapter will represent the experimental results of different scenarios of Humanitarian Minefield Reclamation (HMR) and Military minefield breaching (MMB). In particular, this chapter will illustrate how a MiCAT system is used to conduct a meaningful analysis of the scenarios. All search scenarios that are mentioned in previous chapters are integrated into an Excel application to provide MiCAT users a friendly interface to select their intended minefield search scenario (figure 6.1).

The users can input the number of simulated landmines or obstacles to simulate in order to obtain their desired complexity of the minefield. The other information like number of lanes, number of simulated vehicles, or landform transition information can also be specified in the Excel interface to a desired simulated minefield size (figure 6.2). In addition, the users can also choose to design a simulated minefield with randomly assigned landmine and obstacle locations by inputting the parameters of random seeds. Alternatively, they can specify the value of the desired simulated object at the absolute location of Excel interface (figure 6.3). Each desired simulated object is represented by different values. The value of “1” represents a stone, “2” represents the tree, and “3” represents a landmine. This interface also allows the users to generate a random and desired simulated landform at the specified locations by selecting “Randomly” and “Manually” options at the same time. In order to build a specified landform, the user

may also need to specify a clear spot at an absolute position. The value of “5” is to makes sure that a position has no landmine or obstacle. The information and data that represents the intended simulated minefield search scenario will be loaded through Excel VBA via WITNESS®WCL commands without having to modify the underlying simulation code. Then the intended simulated minefield search scenario will be loaded in WITNESS® for the user to simulate (figure 6.4). The simulation results are also generated in Excel to produce the summary reports (figure 6.5). The findings of the experiments will be analyzed and discussed in the following sections.

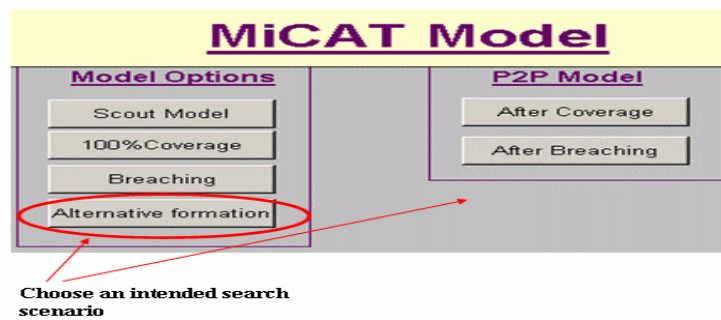


Figure 6.1 MiCAT selection interface

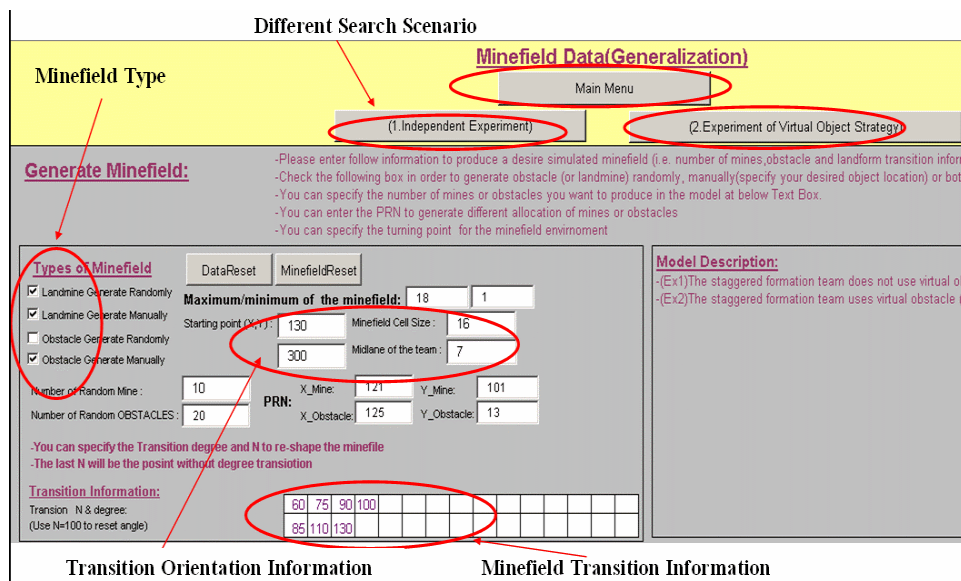


Figure 6.2 Parameters input interface

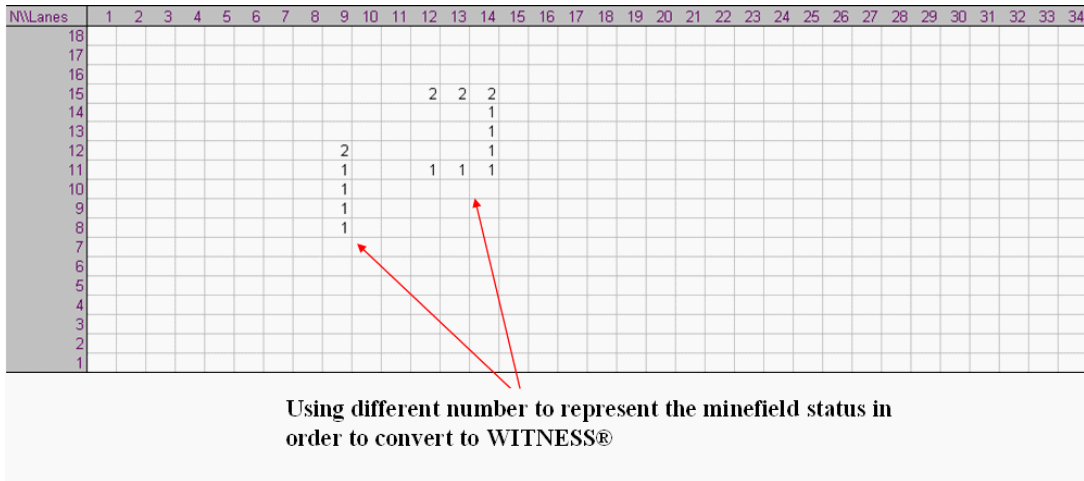


Figure 6.3 Obstacle (or landmine) Locations Input Interface in Excel

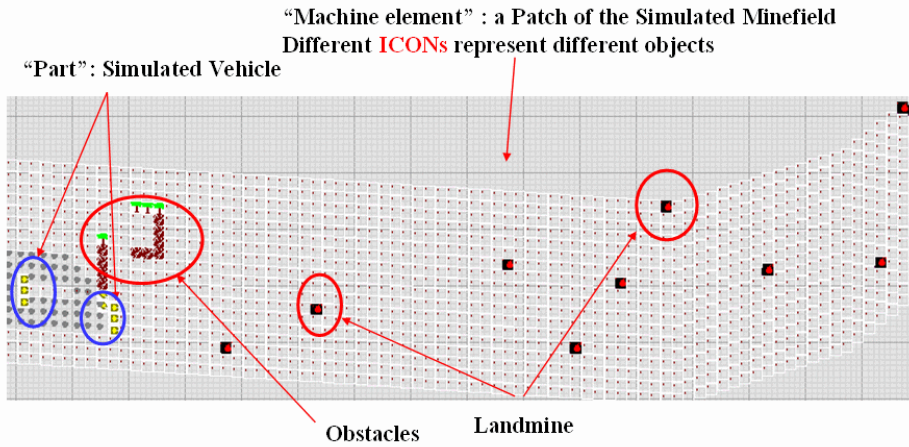


Figure 6.4 The intended simulated minefield search scenario is loaded

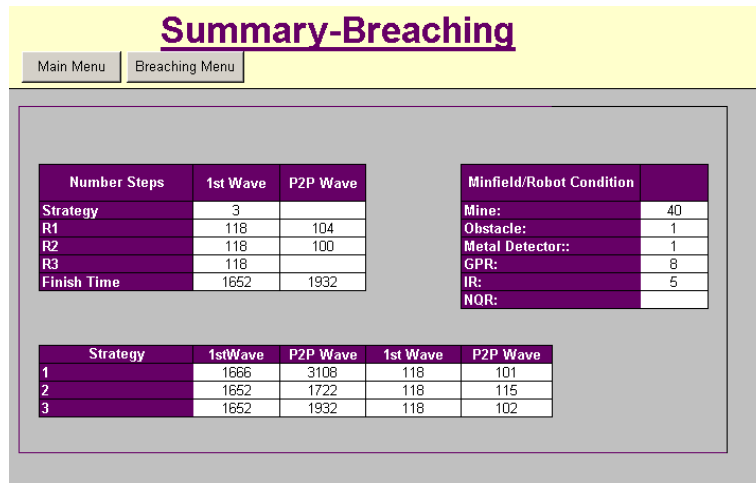


Figure 6.5 Data summary report

6.1 Examples of MiCAT Use Scenarios

This section will describe the simulation results by giving a specific configuration of mine detection resources to determine the search time, number of mine detection vehicles destroyed, and the amount of steps taken (includes the amount of steps that each vehicle has taken and the amount of total searches taken) in the search area. A series of HMR simulation cases are tested within a simulated minefield of 14 lanes with 100 cells in length.

6.1.1 Analysis of Mine Detection Sensor Effectiveness

Three UGVs are assigned to perform the mine detection task. A high complexity minefield is simulated. This experiment assumes a high complexity minefield contains about 10% of the obstacles and landmines, and 70% of them are landmines. This experiment assumes that the total process of time to move from one cell to the next is 0.5 minutes. The vehicle process time for each additional location search is 0.5 minutes. This experiment also assumes the vehicle will use metal detector, GPR, IR, and NQR sensors if needed. The process time of a metal detector is 0.5 minutes. The process time of a GPR sensor is 1 minutes. The process time of an IR sensor is 2 minutes. The process time of a NQR sensor is 3 minutes. The mine sensor detection probabilities of 85%, 90%, and 100% are then tested separately. The results found when running the simulations based on these assumptions above are shown in Table 6.1.

The comparison between the detection probability and the number of searches taken is shown in Table 6.2. These experiments show that Robot2 is destroyed at the

detection probability of 90%, and Robot1 and Robot2 are destroyed at the detection probability of 85%. Because the surviving vehicles in the HMR model are designed to assume the tasks of those that have been destroyed, the simulation results demonstrate that the proportion of work required to complete the HMR task shifts to the remaining vehicles. The mine detection tasks of Robot1 and Robot2 increase about 30% from the detection probability of 100% to 90%. The mine detection tasks for Robot1 increase about 50% and increases about 11% for Robot2 from the detection probability of 100% to 85%. These experiments demonstrate that the total search time depends on how many tasks that the survival vehicles have to perform. They show the total search time increases 30% from the detection probability of 100% to 90%, and 50% from the detection probability of 100% to 85%.

Table 6.1 The Amount of Searches and Time taken at Different Probabilities of Mine Sensor Detection

Detection Probability	100%	90%	85%
Number of Destroyed Robot	0	1	2
Number of Undetected Mine	0	2	3
Robot Name \ Researches			
R1	521	713	990
R2	515	715	447
R3	514	220	220
Total Searches	1550	1648	1657
Search Time	658	925	1267

Table 6.2 The Running Results Comparison at Different Probabilities of Mine Detection Sensor

Rob \ Probability	From 100% to 90%	From 100% to 85%
R1 Searches	Increases 30%	Increases 50%
R2 Searches	Increases 28%	Increases 11%
R3 Searches	Reduces 47%	Reduces 47%
Total Searches	Increases 4%	Increases 5%
Finish Time	Increases 30%	Increases 50%

Based on the above simulation results, figure 6.6 shows the search time has significantly increases when the probability of mine sensor detection is low. As the total mine detection task is fixed, these experiments show that the amount of total searches taken using different detection probabilities of mine sensor has some differences (figure 6.7). The simulation representation results are shown in figure 6.8 (the different colors represents the portions of the minefield searched by different vehicles). If the vehicles have a detection probability of 100%, the mine detection vehicles will search equal areas (figure 6.8(A)). If the detection probability is lower, the vehicles have a higher probability of being destroyed earlier. The search tasks are then reassigned after one is destroyed (figure 6.8(B)).

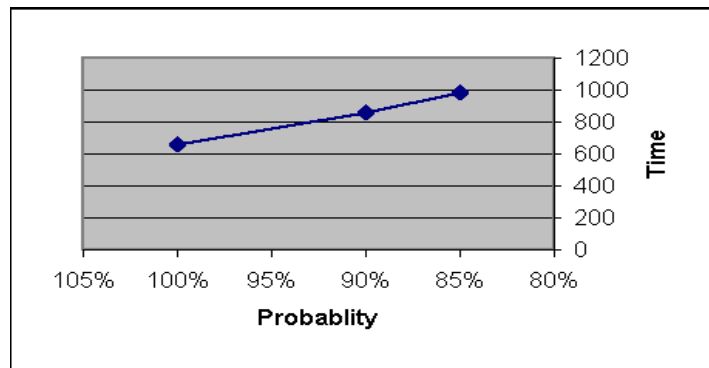


Figure 6.6 The search time increases when using lower mine detection sensor

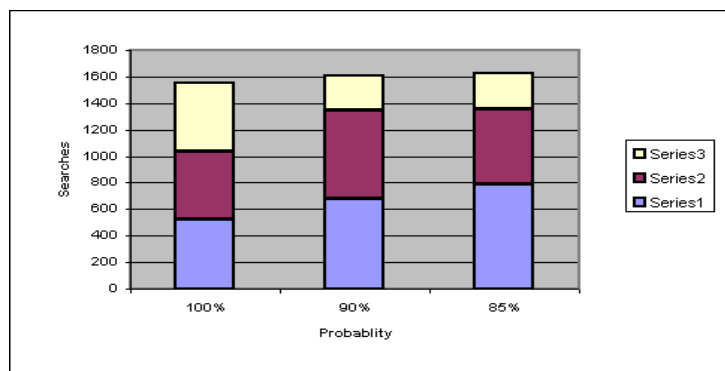


Figure 6.7 The amount of total searches taken is a slight difference

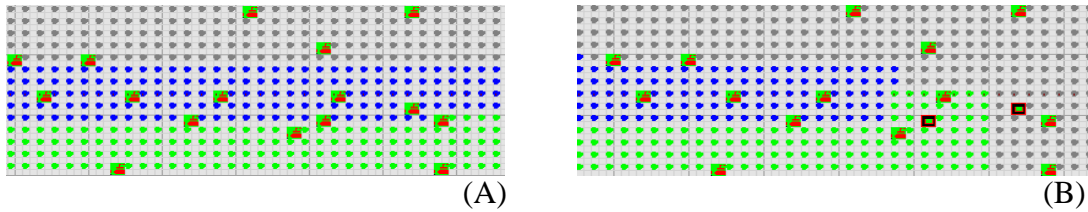


Figure 6.8 Simulation running results at detection probabilities of (A) 100% (B) 90%

6.1.2 The Impact of Minefield Complexity on Search Performance

This section also assigns three UGVs to perform the mine detection task. This experiment assumes the vehicles have all the same configurations as shown in section 6.1.1. Different minefield complexities are tested: **Normal**, **Medium**, and **Complex**. This experiment also assumes that a high complexity minefield contains about 3% obstacles and 7% landmines. Medium complexity minefields contain 3% obstacles and 3% landmines. While Low complexity minefields contain about 2.4% obstacles and 0.6% landmines. The sensor detection probability of 85%, 90%, and 100% are tested separately in the different types of minefields. The following graphs demonstrate the simulation running results based on the assumptions above. Figure 6.9 shows the comparison between using different probabilities of mine detection sensors to search the same type of simulated minefield. Figure 6.10 shows the comparison results of using the same mine detection sensor to search the different types of simulated minefields. Figure 6.9 (A) and (B) shows that there is little difference in the amount of total searches and time taken with different mine detection probabilities sensors within a low complexity minefield. Figure 6.9 (A) shows the amount of total searches taken increases from the detection probability of 100% to 85% when searching in all levels of minefield complexity. This is because the surviving vehicles will move to the place where a

vehicle is destroyed. This extra activity also causes the average search time to increase, especially when the search is taken in the high complexity minefields (figure 6.9(B) and 6.10 (B)). This is because there are more landmines in the higher complexity minefields. A vehicle can be destroyed quickly leaving a greater unsearched area. Figure 6.11 shows that as the sensor detection probability falls, the disparity in the amount of work accomplished by each vehicle increases.

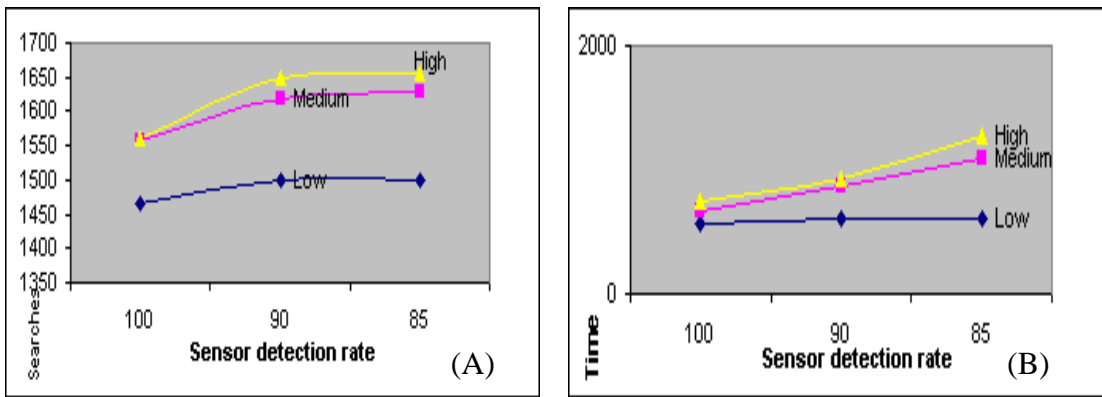


Figure 6.9 Simulation running results of (A) total searches (B) time taken at the different types of mine sensor

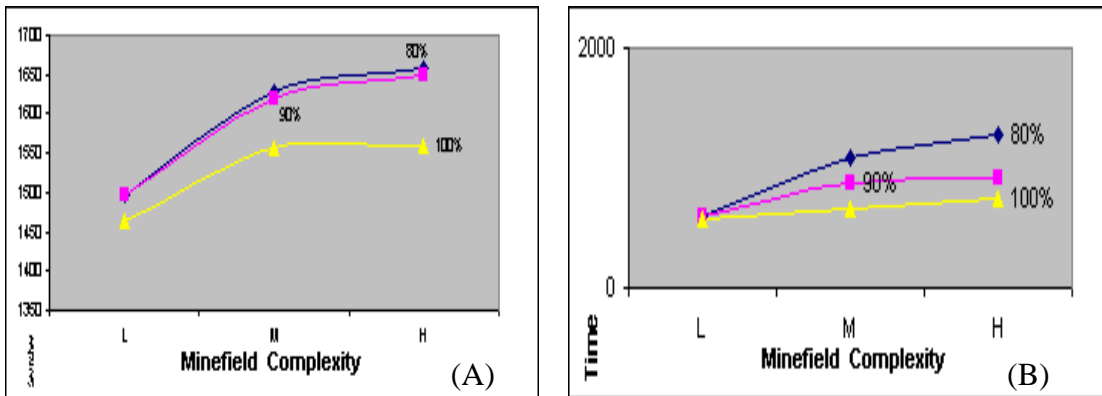


Figure 6.10 Simulation running results of (A) total searches (B) time taken at the different types of minefield

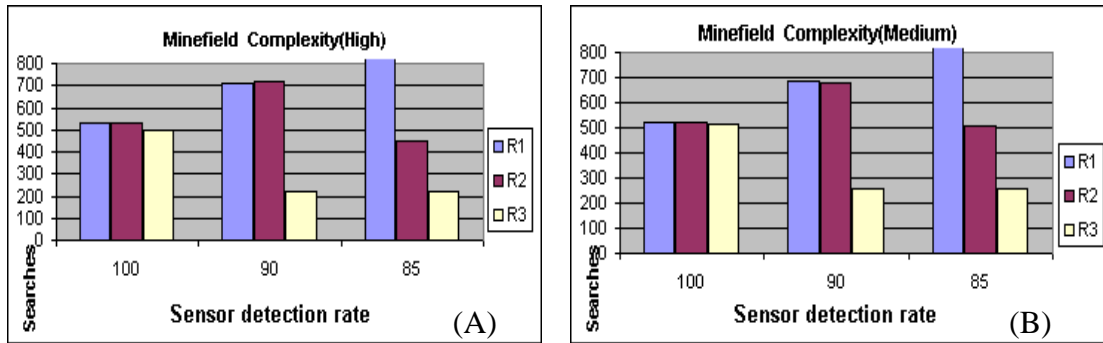


Figure 6.11 Simulation running results for each vehicle when using different type of mine sensors within (A) High (B) Medium complexity minefield

6.1.3 Impact of the Number of Mine Detection Vehicles Used on Search Time

This section investigates the effects of different numbers of mine detection vehicles have on the total time required to perform HMR activity. In order to reduce the variation, the probability of detection is fixed at 100%. At the beginning of the simulation, the desired number of simulated vehicles is specified in the Excel interface. MiCAT can assign the desired number of simulated vehicles to the minefield automatically. The results show that the difference in the number of total searches taken in using a different number of mine detection vehicles on the mine detection task has some differences (figure 6.12(A)). The total processing time taken is much less when using more vehicles on the mine detection tasks (figure 6.12(B)). There is a small increase in the total number of searches taken when using the same number of vehicles to perform the search tasks in a higher complexity minefield. The reason is that the simulated vehicle will need to search more times in order to go around more obstacles and landmines in a higher complexity minefield than in a lower complexity minefield.

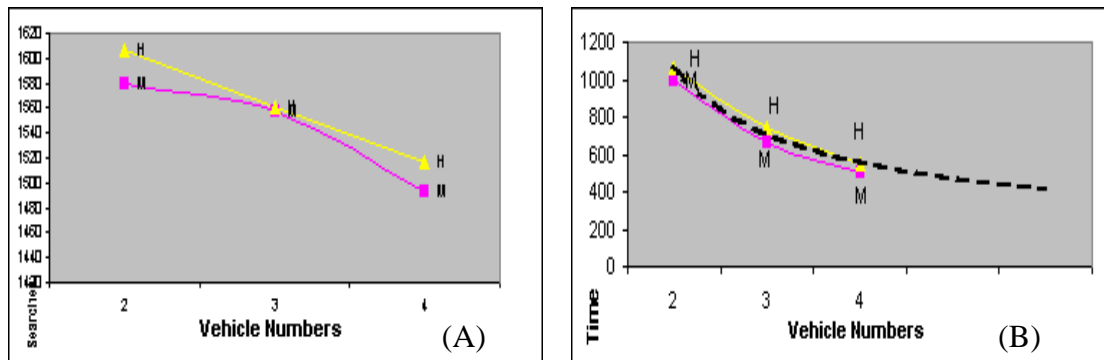


Figure 6.12 (A) the amount of total searches (B) total search time taken, when using different number of vehicles within different type of simulated minefield

6.2 The Use of MiCAT for HMR and MMB System Design

The purpose of this MiCAT application example is to demonstrate the use of the tool to support HMR and MMB System Designs. This will demonstrate the potential for this class of tool to determine the appropriate mine detection resources required to accomplish a specific HMR or MMB task with a specific period of time.

For the purpose of this experiment we assume the mine detection vehicles are operating with 100% accurate mine detection sensors within a medium complexity minefield of 14 lanes; each lane is 100 cells in length. The process time to move from one cell to the next is 0.5 minutes. The vehicle process time for each search is 0.5 minutes. This experiment also assumes the vehicle will use a metal detector, GPR, IR, and NQR sensors. The process time of a metal detector is 0.5 minutes. The process time of a GPR sensor is 1 minute. The process time of an IR sensor is 2 minutes. The process time of a NQR sensor is 3 minutes. The total sensor process time is, therefore, 6.5 minutes if all of the sensors are used for each search. If only one vehicle is assigned to the search, the estimated time taken is about 9100 (=14*100*6.5) minutes. Therefore, if

three vehicles are assigned to search the minefield, the estimation of the time taken would be about 3033 minutes. The simulation results show that the amount of total searches and time taken are 1555 searches and 3882 minutes when three vehicles are assigned.

The above simulation search time is greater than that provided by our simple numeric estimation. This is because the vehicle will need to take more time to search if it finds any obstacle or landmine. This discrepancy illustrates the potential limitations of using simple calculations to predict the amount of resources required to accomplish a given demining task within a specified time and the potential value of a simulation tool like MiCAT to provide higher fidelity estimates of the resources required.

Since three vehicles were assigned in the previous test, total process time taken when using only one vehicle is about 11649 ($=3882*3$). Therefore, if the search task has to be finished in 2500 minutes, a prediction of the number of required mine detection vehicles will be five vehicles ($11649/2500=4.6$). The simulation result (figure 6.13) corresponds to this prediction and shows that the actual process time is 2221 minutes when using five vehicles.

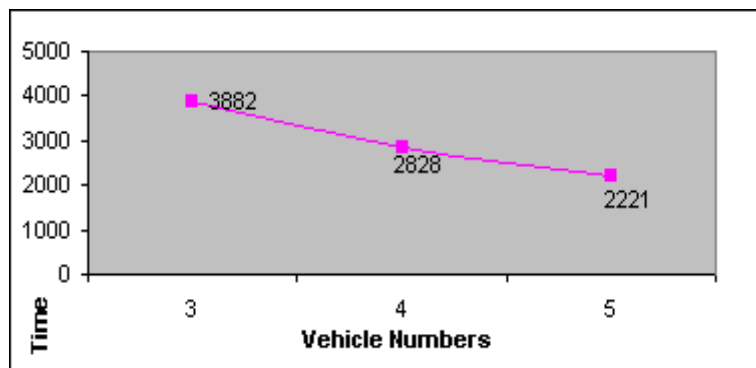


Figure 6.13 The trend line of the time taken

6.3 The Use of MiCAT for the Comparison of HMR and MMB Strategies and Technologies

6.3.1 Evaluation of Various Point-to-Point (P2P) Wave Deployment Strategies for HMR Operations

This section will compare two multi-wave strategies for the Humanitarian Minefield Reclamation (HMR) scenario that are mentioned in section 4.3. The two strategies are listed below:

1. Wait for the Multi-Vehicle Wave to complete its search activities
2. Start working right after the Multi-Vehicle Wave begins its search

This section assumes the use of two separate waves of mine detection resources. The first wave would consist of multiple mine detection resources performing an entire sweep of the search area. It is envisioned that this first wave of mine detection resources would be fast and relatively inexpensive. An example of this type of sensor might be the conventional metal detector. When limited to the location of high metal content mines, conventional metal detectors have good detection probabilities. These sensors, however, suffer from a high probability of false-positive errors due to the fact that they are not able to distinguish between a stray piece of metal debris and the metal contained in a mine. To reduce the impact of these false-positive errors, a second wave of more discriminating mine detection resources can be deployed to determine if the location identified by the metal detector actually contains a mine or a random piece of metal. This second wave of mine detection resources might utilize sensors that are slower and are more costly. This second wave would not attempt to completely re-search the minefield but would perform a point-to-point search, returning to the locations

identified by the first wave of sensors and accessing whether or not the location actually contains a mine.

The study presented below assumes the vehicles in Multi-Vehicle Wave only have a metal detector. The process time to move from one cell to the next is *0.5 minutes* in the Multi-Vehicle Wave. The process time of a metal detector is assumed to be *0.5 minutes*. The P2P Wave vehicles will carry highly capable mine detectors. The process time to move from one cell to the next is also *0.5 minutes*, if the P2P Wave vehicle does not need to use the highly capable detectors. Otherwise, the total process time is *6.5 minutes*. In order to reduce the variation, 100% probability of detection sensor is assumed in these experiments. A medium complexity minefield is also assumed. The simulation results are generated to figure 6.14 (A) and (B) based on the above assumptions. The finish time between the first and second strategies shows a great difference. These preliminary experiments run within MiCAT indicate that the time lost due to waiting for the completion of the previous wave more than consumes any timesaving, by using optimal vehicle assignments that require a predefined set of route locations. However, the simple load balancing strategies that handle the dynamic definition of route locations permit the P2P wave to begin right after the previous wave starts its search can save the waiting time than using an optimal vehicle assignment. However, figure 6.14(B) shows that the number of search steps by both methods is very similar because the minefield size is fixed.

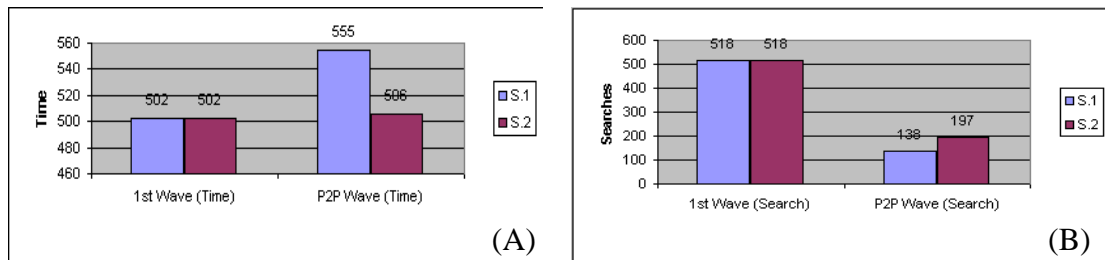


Figure 6.14 (A) The amount of time (B) searches taken, between different multi-wave strategies

6.3.2 Evaluation of Various Point-to-Point (P2P) Wave Deployment Strategies for MMB Operations

Section 6.3.1 has compared two multi-wave strategies for the Humanitarian Minefield Reclamation (HMR) scenario. This section will make a comparison between different strategies for the Military minefield breaching (MMB) scenario. This section assumes only a metal detector is carried on the vehicles in the Multi-Vehicle Wave of MMB scenario. The total processing time to move from one cell to the next is assumed to be *0.5 minutes*. This section also assumes the total process time to move from one cell to the next is *0.5 minutes* when the P2P Wave vehicle only uses a metal detector. If the P2P Wave vehicle needs to use the highly capable detectors, the process time is *6.5 minutes*. Two strategies are listed below:

1. Wait for a breaching path is to be completely determined
2. Start working right after the Multi-Vehicle Wave starts its search

The results show that there is a significant difference in the search time (figure 6.15 (A)) between the first and second strategies but not much difference in the number of searches performed (figure 6.15 (B)). It also proves that the time consumed on the

searching task can be reduced if the Point-to-Point Wave and the Multi-Vehicle Wave can work simultaneously.

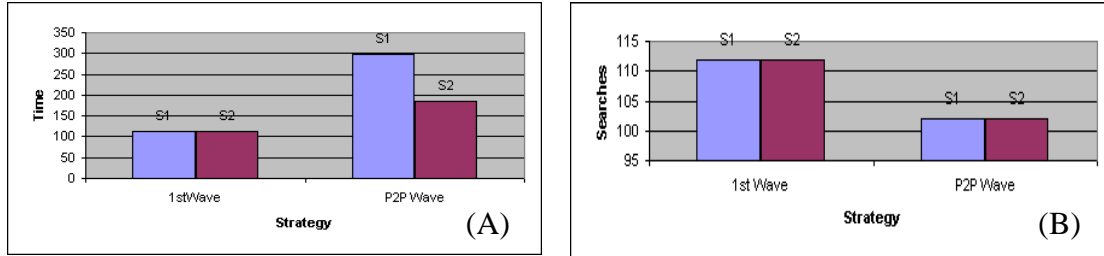


Figure 6.15 (A) The amount of time (B) searches taken, between different multi-wave strategies of MMB scenario

6.3.3 Evaluation of the Deployment of a Single Wave of High Capable Vehicle or Using Multiple Waves of Heterogeneous Vehicles

Single wave and multiple waves are addressed in this paper. The single wave only uses highly developed vehicles to search the minefield. If the multiple-wave strategy is used, the first wave will use simpler mine detection vehicles, and the second wave will use highly developed vehicles to search for possible landmines. A simpler mine detection vehicle only has a metal detector. This experiment assumes that each highly capable mine detection vehicle has a metal detector, a GPR sensor, an IR detector, and a NQR sensor. Different experiments are simulated for the Humanitarian Minefield Reclamation (HMR) scenario. The process of time for each vehicle depends on its carried sensors. It assumes the total time from moving one cell to the next cell is *1 minute* when the vehicle uses only a metal detector. This section also assumes the process time of these sensors is the same as mentioned in the section 6.1.1. Thus, if the vehicle needs to use the highly capable detectors, the process time of one type of vehicle is *6.5 minutes*.

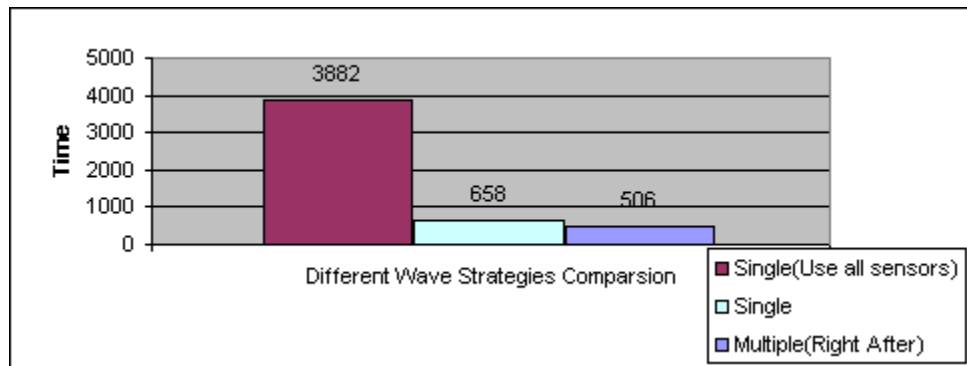


Figure 6.16 The search time comparison between a single and multiple waves

The simulation results show that using the multiple-wave strategy is better than using a single-wave strategy. This P2P strategy assumes that the vehicle starts working right after the Multi-Vehicle Wave. A conclusion can be made from these observations; the search time is saved because a multiple-wave strategy can distribute the workload more efficiently.

6.4 The Use of MiCAT to verify HMR and MMB Simulation Algorithm

6.4.1 The Development of Multi-Vehicle Control Algorithm

This paper has mentioned two types of strategies to maintain the staggered formation team: virtual and non-virtual. Virtual landmine (or obstacle) strategy will make the dependent vehicle team follow the other team's movement in order to maintain a breaching path with no gaps. The Non-Virtual landmine strategy is used by an independent vehicle team. They are independent and will not interact with other vehicles.

Figure 6.17 (A) shows the simulation results for a vehicle team that uses a virtual object strategy. It can hold a path width because the rear vehicle can cover the

gap between front vehicles. However, without the coordination between the vehicle members, the vehicles do not know where they have to cover the gap or move to the same direction to maintain the breaching path width (figure 6.17 (B)). If the military uses the Non-Virtual object strategy and expects to find a breaching path concerning the path width, it will fail to meet the expectation. On the other hand, the independent vehicle team can only carry a simpler and more cost effective arithmetic unit than the dependent vehicle team because they do not need to coordinate with each other while performing the obstacle avoiding behavior.

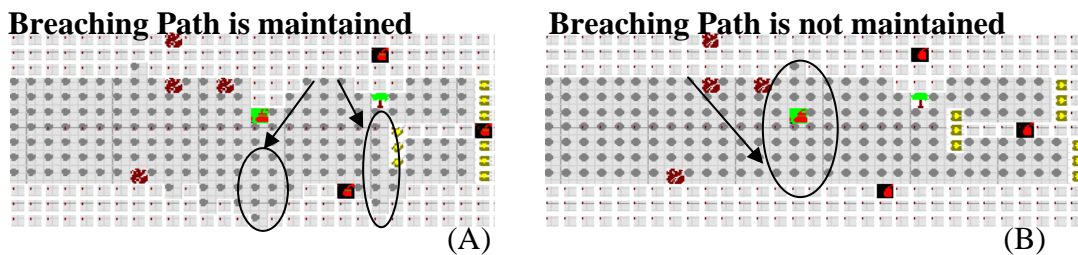


Figure 6.17 The Virtual (A) and Non- Virtual obstacle (B) strategy

6.4.2 The Development of Vehicle Re-Tasking Algorithm Response to Vehicle Loss

This research has developed an algorithm that is used to reassign the search task whenever a vehicle is lost. This section proves that the supporting strategy increases the percent of area coverage. The mine detection probability of 65% and 60% are assumed in this section. The results show that the reassign task strategy could cover almost 100% of the search task (figure 6.18 & 6.19 (B)). If the reassign task strategy is not used, there are still 28% uncovered (figure 6.18(A)) at probability of 65% and 45% uncovered (figure 6.19(A)) at probability of 60%. From these results, this paper found that when the vehicle carries less capable detection sensors, it will be destroyed more quickly.

Therefore, there is a significant difference in the coverage rate when using lower mine detection probability.

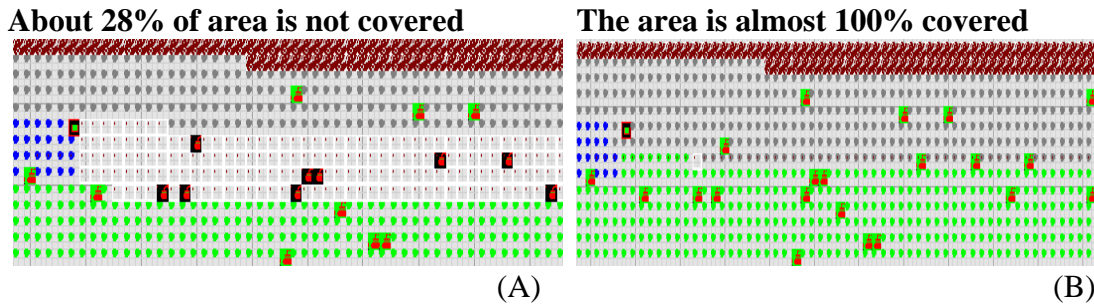


Figure 6.18 The complete coverage search task at detection probability of 65% (A) Non- Support (B) Reassignment Support

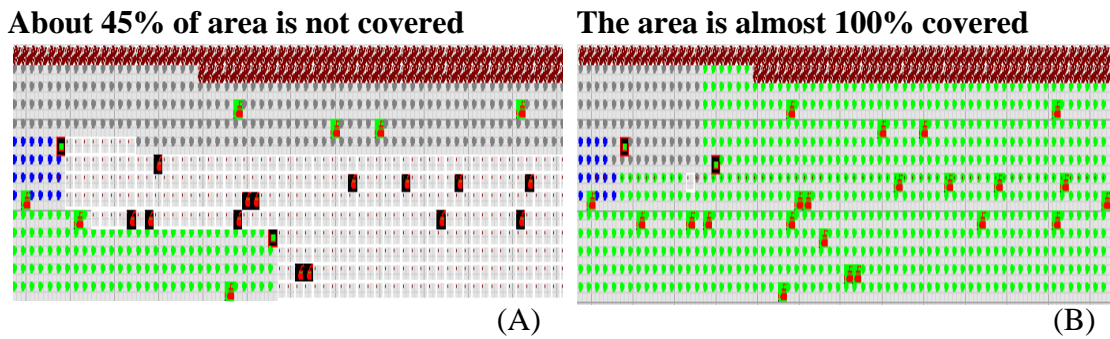


Figure 6.19 The complete coverage search task at detection probability of 60% (A) Non- Support (B) Reassignment Support

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

This dissertation has presented several demining strategies. A Mine Clearing Analysis Tool (MiCAT) is used to analyze of various automated mine clearing technologies. The preliminary work on MiCAT, presented above, has allowed users to dynamically specify the characteristics of the mine detection problem. Therefore, through the EXCEL interface, one can use MiCAT to evaluate the potential of developing parameterized algorithms.

The major approaches to demining strategies are:

- Breach Path Verification with a Scout Vehicle - The simulation prototype of a scout vehicle can determine if a proposed breaching path is viable. This resource can scan the surface of the minefield to detect physical obstacles and determine if a breach path of the required width can be found. The breach path verification's search algorithm has been expanded to include the ability to identify simulated surface obstacles out to the limits of the required breach lane width to support a given breach formation.
- Humanitarian Minefield Reclamation (HMR) – As a part of this research, HMR modeling and vehicle control algorithms have been developed that allow simulated

search vehicles to behave in a way that is consistent with how we would like the proposed autonomous mine detection vehicles to behave. The simulated vehicles will now move back and search the areas of the simulated minefield that were originally blocked by large formations of mines or obstacles. The HMR simulation algorithms are expanded to handle the allocation of multiple search vehicles in a manner that balances the search area assigned to each vehicle. The algorithms will also re-balance the work load of the vehicle team if a vehicle in the team is destroyed or malfunctions.

- Military Minefield Breaching (MMB) – Algorithms for the MMB simulation prototype have been developed that model a team of search vehicles that work together side-by-side, to search a breach lane of a specified width. We assume that the vehicles that will be expected to follow this breach lane will not have a zero-turn radius and consequently will not be able to turn instantaneously. To accommodate this limitation, the MMB simulation logic was enhanced to increase the search area evaluated by the team of robotic mine detection sensors when the breach path changes directions.
- Point-to-Point (P2P) Multi-Pass Mine Detection Strategies - One of the objectives of this research was to produce a tool that could evaluate different demining strategies and technologies. One of the strategies of interest was the use of multiple waves of mine detection resources that possess different capabilities. In this scenario a second wave of vehicles would reevaluate locations that a previous wave of vehicles has identified as having a significant probability of containing a

mine. To support this scenario, this research has developed simulation logic that models Point-to-Point search strategies. Different types of algorithms were developed to allocate multiple mine detection vehicles to search a series of high-probability mine locations. This work evaluated the application of optimized search location assignment strategies that require a prior knowledge of the set of desired search locations. This work also investigated the use of heuristics-based assignment strategies that operate without a prior knowledge of the required search locations. The implementation of the algorithms that model these two competing Point-to-Point search location strategies within the MiCAT has also allowed the performance of various mine detection systems using these two strategies to be compared.

- A key feature of any simulation tool is its ability to be quickly modified to meet the characteristics and constraints of multiple system scenarios. Over the course of this work, considerable effort has been made to ensure the generality of the proposed MiCAT system. The MiCAT prototype provides a form-based operator interface that allows the user to select and parameterize the specific demining scenario they wish to investigate without directly modifying any of the simulation code. The MiCAT interface also allows the placement of mines and obstacles in the simulated minefield to be determined randomly or to be specified explicitly by the user. This work has also tackled additional problems that in the beginning of this effort appeared to limit the generality of this work. In our earliest attempts to simulate minefield demining operations, there was an implicit assumption that the

smallest area of land that could be searched was the same size of the search vehicle. This assumption greatly simplified the early simulation logic development effort but would preclude our ability to compare systems that offered different physical sizes of sensor arrays. We have expanded our algorithms to allow our search vehicles to be a whole number of minefield “pixels” wide. Our early simulation models assumed that our minefields were rectangular in shape that could be easily represented as a matrix of minefield cells or pixels. The MiCAT prototype can now vary the shape of the search area by shifting the location of the minefield cells within the minefield display. This work has also investigated the modeling of alternative search team configurations. Earlier work simply modeled a team of search vehicles as a line of vehicles moving side-by-side across a search area. In reality, however, vehicles searching in this manner will most likely leave a small gap of unsearched ground between them due to their simple desire not to collide into each other during search operations. An alternative search wave formation would consist of two staggered lines of search vehicles, with the second line of vehicles positioned in such a way as to cover the gaps between adjacent vehicles in the first lane. Simple search vehicle formations like the staggered multi-line formation described above have been simulated in MiCAT. This has been challenging due to the coordination required of all the members of the simulated search team. Team coordination strategies such as the “Virtual Obstacle” strategy have been developed to ensure that the simulated search vehicles move as a coordinated team.

The primary objective of this research was to explore the feasibility of creating a discrete-event simulation-based tool that could support the analysis and comparison of various mine detection strategies and technologies. The MiCAT prototype system is offered as an existence proof that this class of tool can be created and that the tool can be flexible enough to support the meaningful analysis of alternate demining strategies and technologies.

7.2 Future Work

This dissertation has developed a Mine Clearing Analysis Tool (MiCAT) to determine if it is possible to represent or model the proposed demining strategies for humanitarian minefield reclamation or military minefield breaching applications. MiCAT has also supported the analysis of various mine detection strategies and technologies. This research will continue to explore different implementations in mine detection strategies and technologies.

In addition to exploring various strategy possibilities, some of the developing parametric algorithms have to be optimized. The possibilities are listed below:

- The current HMR search algorithms have reduced the occurrence of unsearched areas, not only at the time meets large obstacles (or clusters of mines) but also the time one of the vehicles is destroyed. A parameter tuning of the HMR search algorithms will be explored to make the vehicles able to achieve 100% search in different minefield environments.

- The current Scout and MMB search algorithms are able to support breach troop formation. In order to maintain the minimum turning radius, the Scout search algorithms have defined a scan distance parameter to support the searching of multiple minefield cells prior to each vehicle move. In the MMB search algorithm, a distance parameter is defined to make the vehicles go back to maintain the minimum turning radius if a landmine or obstacle blocks the path. However, the search algorithms limit the lane width parameter to an odd number of lanes. The future work will explore the feasibility of developing an algorithm that allows a flexible lane width parameter and an optimum distance parameter.
- In order to provide 100% coverage of the planned breach path, the current Scout and MMB search algorithms will make the coordinated team of vehicles to re-define a breach path if a mine or obstacle is detected. However, if it is not able to re-define a breach path, the current Scout and MMB search algorithm will make the vehicles stop to search. The future work will make the vehicles able to explore an alternative breach path if a detected mine can be identified for removal or destruction by subsequent MMB resources.
- The current P2P search algorithms are capable of allocating multiple mine detection vehicles to the high-probability locations within the simulated minefield. At this time, the P2P vehicles move within a limited area. The more complex work assignment of vehicle routing optimization techniques will be investigated, in order to optimize the problem of the increasing number of lanes of the simulated minefield and mine detection vehicles.

- This research has only represented a small patch of the simulated minefield. The minefield extensions of a steeper slope of a minefield problem and a turning around radius whenever the vehicle following the boustrophedon path from one piece of the minefield to another will continue to be explored in future work.
- Currently, MiCAT uses simple sensor detection logic. More advanced sensor fusion mechanisms (e.g. belief function or Dempster-Shafer theory) will be investigated to determine how real-world mine detection activities affect the minefield search.
- To identify and work with experts in the HMR community and the MMB community to obtain realistic performance information of actual mine detection equipment, so that we can validate the MiCAT tool.
- Plan to apply MiCAT of other applications, like: Military Convoy Risk Assessments for Improvised Explosive Devices.

REFERENCES

- Acar, E. U., Choset, H., Zhang, Y., et al. (2003). "Path Planning for Robotic De-mining: Robust Sensor-based Coverage of Unstructured Environments and Probabilistic Methods." *The International Journal of Robotics Research*, 22(7-8), 441-466.
- Alvarenga, G. B., Mateus, G. R., and de Tomi, G. (2007). "A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows." *Computers & Operations Research*, 34(6), 1561-1584.
- Arkin, R. C., and Balch, T. (1998). "Cooperative Multiagent Robotic Systems." *Artificial Intelligence and Mobile Robots*, D.Kortenkamp, R.P. Bonasso, and R. Murphy, eds., MIT/AAAI Press, Cambridge,MA.
- Balch, T., and Arkin, R. C. (1995). "Motor Schema-based Formation Control for Multiagent Robot Teams." *Proceedings of the First International Conference on Multiagent Systems*, San Francisco.
- Balch, T., and Arkin, R. C. (1998). "Behavior-based formation control for multirobot teams." *Robotics and Automation, IEEE Transactions on*, 14(6), 926-939.
- Balch, T., and Hybinette, M. (2000). "Social potentials for scalable multi-robot formations." *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, San Francisco, CA, 1, 73-80.
- Berger, J., Barkaoui, M., and Boukhtouta, A. (2007). "A hybrid genetic approach for airborne sensor vehicle routing in real-time reconnaissance missions." *Aerospace Science and Technology*, 11, 317-326.
- Bonabeau, E. (1999). *Swarm intelligence : from natural to artificial systems* / Eric Bonabeau, Marco Dorigo, Guy Theraulaz., New York ; Oxford : Oxford University Press.
- Bräysy, O. (2001). "Genetic Algorithms for the Vehicle Routing Problem with Time Windows." Internal Report STF42 A01021, SINTEF Applied Mathematics, Department of Optimization, Norway.
- Bräysy, O., and Gendreau, M. (2002). "Tabu Search heuristics for the Vehicle Routing Problem with Time Windows." *Sociedad de Estadística e Investigación Operativa, TOP*, 10(2), 211-237.
- Brown, D. T. (2001). "Routing unmanned aerial vehicles while considering general restricted operating zones," Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio.
- Canada, F. A. (2005). "Canadian Landmine Fund Annual Report 2003-2004", Canada's Mine Action Archived Publications.
- Cao, Y. U., Fukunaga, A. S., and Kahng, A. (1997). "Cooperative mobile robotics:Antecedents and directions." *Autonomous Robots*, 4, 7-27.

- Cassinis, R., Bianco, G., Cavagnini, A., et al. (1999). "Strategies for navigation of robot swarms to be used in landmines detection." *Advanced Mobile Robots, 1999. (Eurobot '99) 1999 Third European Workshop on*, 211-218.
- Chapman, E., and Sahin, F. (2004). "Application of swarm intelligence to the mine detection problem." *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, 6, 5429-5434.
- Chen, Q., and Luh, J. Y. S. (1994). "Coordination and control of a group of small mobile robots." *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, San Diego, CA, 3, 2315-2320.
- Choi, E., and Tcha, D.-W. (2007). "A column generation approach to the heterogeneous fleet vehicle routing problem." *Computers & Operations Research*, 34(7), 2080-2095.
- Choset., H., and Pignon, P. (1997). "Coverage path planning: The boustrophedon cellular decomposition." *Proceedings. International Conference on Field and Service Robotics*, Canberra, Australia.
- Colon, E., Cubber, G. D., Ping, H., et al. (2007). "Integrated robotic systems for Humanitarian Demining." *International Journal of Advanced Robotic Systems*, 4(2), 219-228.
- Cook, D. J., Gmytrasiewicz, P., and Holder, L. B. (1996). "Decision-theoretic cooperative sensor planning." *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(10), 1013-1023.
- Cordeau, J.-F., Laporte, G., and Mercier, A. (2001). "A unified tabu search heuristic for vehicle routing problems with time windows." *Journal of the Operational Research Society*, 52, 928-936.
- Debenest, P., Fukushima, E. F., and Hirose, S. (2003). "Proposal for automation of humanitarian demining with buggy robots." *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1, 329-334.
- Dollarhidea, R. L., and Agah, A. (2003). "Simulation and control of distributed robot search teams." *Computers and Electrical Engineering*, 29, 625-642.
- Fruergaard-Pedersen, R. (2006). "Optimal demining using a swarm of low-cost robotic units," University of Aarhus, Denmark.
- Gelenbe, E., and Cao, Y. (1998). "Autonomous search for mines." *European Journal of Operational Research*, 108(2), 319-333.
- Gooneratne, C. P., Mukhopahyay, S. C., and Gupta, G. S. (2004). "A Review of Sensing Technologies for Landmine Detection: Unmanned Vehicle Based Approach." *2nd International Conference on Autonomous Robots and Agents*, Palmerston North, New Zealand.
- Gribkovskaia, I., Halskau, s. O., Laporte, G., et al. (2007). "General solutions to the single vehicle routing problem with pickups and deliveries." *European Journal of Operational Research*, 180(2), 568-584.
- Habib, M. K. (2001). "Mine detection and sensing technologies-new development potentials in the context of humanitarian demining." *The 27th Annual Conference of the IEEE in Industrial Electronics Society*, Denver, CO, USA, 3, 1612-1621.

- Habib, M. K. (2007). "Humanitarian Demining: Reality and the Challenge of Technology -The State of the Arts." *International Journal of Advanced Robotic Systems*, 4(2), 151-172.
- Harder, R. W. (2000). "A Java Universal Vehicle Router in Support of Routing Unmanned Aerial Vehicles," Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio.
- Healey, A. J. (2001). "Application of formation control for multi-vehicle robotic minesweeping." *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, 2, 1497-1502.
- Hsu, H. C.-H. (2005). "Multi-Teams Formation Control for mobile robot ", National Chung Cheng University, Chia-Yi, Taiwan.
- Hsu, H. C.-H., and Liu, A. (2005a). "Multiagent-Based Multi-team Formation Control for Mobile Robots " *Journal of Intelligent and Robotic Systems*, 42(4), 337-360.
- Hsu, H. C., and Liu, A. (2005b). "Applying a Taxonomy of Formation Control in Developing a Robotic System." *Tools with Artificial Intelligence, 2005. ICTAI 05. 17th IEEE International Conference on*, 3-10.
- Hsu, H. C. H., and Liu, A. (2004). "Multiple teams for mobile robot formation control." *Intelligent Control, 2004. Proceedings of the 2004 IEEE International Symposium on*, 168-173.
- Hu, H., and Brady, M. (1997). "Dynamic global path planning with uncertainty for mobile robots in manufacturing." *Robotics and Automation, IEEE Transactions on*, 13(5), 760-767.
- Huang, W. H. (2000). "The minimal sum of altitudes decomposition for Coverage algorithms." Rensselaer Polytechnic Institute Computer Science Technical Report, New York.
- Kinney, G. W., Jr. (2000). "A Hybrid Jump Search and Tabu Search Metaheuristic for the Unmanned Aerial Vehicle (UAV) Routing Problem," Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio.
- Kruusmaa, M. (2003). "Global Navigation in Dynamic Environments Using Case-Based Reasoning." *Autonomous Robots*, 14(1), 71-91.
- Kumar, V., and Sahin, F. (2003). "Cognitive maps in swarm robots for the mine detection application." *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, 4, 3364-3369.
- Li, F., Golden, B., and Wasil, E. (2006). "The open vehicle routing problem: Algorithms, large-scale test problems, and computational results " *Computers and Operations Research*, , 34(10), 2918-2930
- Liu, B., Zhang, R., and Shi, C. (2006). "Formation Control of Multiple Behavior-based robots." *Computational Intelligence and Security, 2006 International Conference on*, Guangzhou , 1, 544-547.
- Ludwig, P. M. (2000). "Formation Control For Multi-Vehicle Robotic Minesweeping," NAVAL POSTGRADUATE SCHOOL MONTEREY CA.
- Milisavljevic, N., and Bloch, I. (2003). "Sensor fusion in anti-personnel mine detection using a two-level belief function model." *Systems, Man and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 33(2), 269-283.

- Milisavljevic, N., Bloch, I., and Acheroy, M. (2000). "Characterization of mine detection sensors in terms of belief functions and their fusion, first results." *Information Fusion, Proceedings of the Third International Conference on*, 2, THC3/15-THC3/22.
- Mudigonda, N. R., Kacelenga, R., and Erickson, D. (2003). "The application of Dempster-Shafer theory for landmine detection." *Proceedings of SPIE*, 5099(1), 103-112.
- Munirajan, V. K., Sahin, F., and Cole, E. (2004). "Ant colony optimization based swarms: implementation for the mine detection application." *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, 1, 716-721.
- O'Rourke, K. P., Bailey, T. G., Hill, R., et al. (1999). "Dynamic Routing of Unmanned Aerial Vehicles Using Reactive Tabu Search." AIR FORCE INST OF TECH WRIGHT-PATTERSONAFB OH.
- Potvin, J.-Y., and Bengio, S. (1996). "The Vehicle Routing Problem with Time Windows-- Part II: Genetic Search." *INFORMS Journal on Computing*, 8(2), 165-172.
- Potvin, J.-Y., Garcia, B.-L., and Rousseau, J.-M. (1996). "The Vehicle Routing Problem with Time Windows -- Part I: Tabu Search." *INFORMS Journal on Computing*, 8(2), 158-164.
- Prins, C. (2004). "A simple and effective evolutionary algorithm for the vehicle routing problem." *Computers and Operations Research*, 31(12), 1985-2002.
- Rajasekharan, S., and Kambhampati, C. (2003). "The current opinion on the use of robots for landmine detection." *Robotics and Automation, Proceedings. IEEE International Conference on*, 3, 4252-4257.
- Ramaswamy, K., Agarwal, S., and Rao, V. S. (2000). "Data fusion and evidence accumulation for land mine detection using the Dempster-Shafer algorithm." *Proceedings of SPIE*, 4038(1), 865-876.
- Russell, M. A., and Lamont, G. B. (2005). "A Genetic Algorithm for Unmanned Aerial Vehicle Routing." *Proceedings of the 2005 conference on Genetic and evolutionary computation*, , Washington DC, 1523 - 1530
- Ryan, J. L. (1998). "Embedding a Reactive Tabu Search Heuristic in Unmanned Aerial Vehicle Simulation," Air Force Institute of Technology (AU), Wright-Patterson Air Force Base, Ohio.
- Rybski, P. E., Papanikolopoulos, N. P., Stoeter, S. A., et al. (2000). "Enlisting rangers and scouts for reconnaissance and surveillance." *Robotics & Automation Magazine, IEEE*, 7(4), 14-24.
- Shafer, G. (1990). *Readings in Uncertain Reasoning*, Morgan Kaufmann
- Shaw, P. (1998). "Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems " *Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming* 417 - 431
- Tavakkoli-Moghaddam, R., Safaei, N., Kah, M. M. O., et al. (2007). "A New Capacitated Vehicle Routing Problem with Split Service for Minimizing Fleet Cost by Simulated Annealing." *Journal of the Franklin Institute*, 344(5), 406-425.

- Wong, S. C., and MacDonald, B. A. (2003). "A topological coverage algorithm for mobile robots." *Intelligent Robots and Systems, Proceedings. IEEE/RSJ International Conference on*, 2, 1685-1690.
- Zhang, Y. (2004). "Hierarchical spatial model and Monte Carlo Analysis of mine locations in robotic land-mine search," Carnegie Mellon University, Pennsylvania.
- Zhang, Y., Schervish, M., Acar, E. U., et al. (2001). "Probabilistic methods for robotic landmine search." *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, 3, 1525-1532.
- Zhang, Y., Schervish, M., and Choset, H. (2002). "Probabilistic hierarchical spatial model for mine locations and its application in robotic landmine search." *Intelligent Robots and System, 2002. IEEE/RSJ International Conference on*, 1, 681-689.

BIOGRAPHICAL INFORMATION

The author received her doctorate in Industrial Engineering from The University of Texas at Arlington. She received her M.S degree in Logistics from The University of Texas at Arlington and B.S. degree from National Cheng Kung University.

The author has participated in several software applications when she worked as a software engineer. These projects include: Airline Timetable Management System, Air Traffic Flow Statistic/ Analysis System, and Over flight charging system for Civil Aeronautic Administration, Taiwan, and Land Administration System for Dept of Land administration, Taiwan. The author is currently interested in acquiring knowledge of applied statistics, applied operation research and logistics to the area of simulation, software engineering, and system and data analysis.