Development of advanced virtual collaborative multi-vehicle simulation platform

Toh, Teow Ghee

2012

Toh, T. G. (2012). Development of advanced virtual collaborative multi-vehicle simulation platform. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/51107>

<https://doi.org/10.32657/10356/51107>

Development of Advanced Virtual Collaborative Multi-Vehicle Simulation Platform

Toh Teow Ghee

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Master of Engineering

2012

Acknowledgements

I wish to take this opportunity to express my utmost gratitude and appreciation to the people who have assisted me in the preparation of this thesis.

I would like to thank my thesis supervisor, Professor Xie Lihua, for his patient guidance and for allowing me the freedom to explore my interest in developing the entire simulation platform. I would also like to thank Dr. Xu Jun, for guiding me in the area of algorithm integrations and important opinions on enhancing the simulation platform

Special thanks to Dr Liu Shuai and Dr Hu Jin Wen for their guidance on formation algorithm and search algorithm respectively.

Thanks to Mr. Chen Chun Lin, from Chang Gung University for his help and involvement in developing the hardware required for conducting formation test

I am grateful for the assistance of Toh Yue Khing, Fan Tong Lin, Raymond, Tan Chun Chuan, Shen Ying and Chee Wei Hong in helping to build the virtual environment and conducting my experiments

Abstract

Setting up a hardware platform for multi-vehicle cooperation can be very complex, and resource and time consuming. Factors like vehicle dynamics and operating environment would affect the performance and need to be handled as well. Part of the purpose of this research is to develop a virtual platform that allows user to overcome these problems to test their algorithms without the need of setting up the real hardware. This virtual platform must have similar performance with the algorithm running in the real world and allows easy porting as well. The scope of work would include use of sensors for localization and collision avoidance, wireless communications for information exchanges, firmware and software programming for implementing control algorithms, etc.

With the virtual platform, the second major part of the work here is to implement formation and search control in this platform and address all the underlying difficulties. There are two underlying difficulty areas being identified here. The first is to ensure that the performance is close to that in the real world. As the implementation involves a number of algorithms, formation control, search control, obstacle avoidance, tracking and pattern recognition, some of the algorithms would compete between each other for the kinematics control of the vehicle. Thus the second difficulty here is to focus on strategies of switching between algorithms to allow smoother taking over during operations. There are more focused efforts being spent here on the switching between formation and obstacle avoidance, as it conventionally occupies the vast majority of the kinematics control. This research presents an obstacle avoidance algorithm that is based on logical understanding of the surrounding, and adaptively allows multi-vehicle to change formation on a real-time basis.

This report introduces the needs and motivation for developing the virtual platform and new algorithms for multi-vehicle navigation. Previous work and background information specific to the development of virtual platform are reviewed, followed by an overview of

the virtual platform and its constructing modules, namely Real World, Transmission, Central Processing/Intelligence & Interface to Virtual World, and Virtual World. The purpose of each portion will be elaborated. Details of the implementation of formation and search control will be covered. The problems and the shortcomings in implementation will be investigated and how these problems are being resolved would also be discussed. Finally, results and the performance of the virtual platform testing will be presented.

From the results of the comparison between real world scenario and virtual world scenario, we can conclude that the Virtual Platform has been able to serve as a hybrid simulator. It allows real vehicles and hardware to be simulated. From the 3D simulations done for autonomous air vehicles carrying out reconnaissance mission, we can conclude that the strategy of switching between algorithms has been successful. 3D simulation platform has been able to serve its purpose well. The proposed obstacle avoidance, being very light in computation requirement, shows great flexibility by allowing itself to be easily integrated with other algorithms. Further enhancement including upgrading the 3D rendering effect, stable server-client model and wireless simulation, makes the whole simulation platform even closer to running in the real world environment.

Table of Contents

Acknowledgement.....	I
Summary.....	II
Contents.....	IV
List of Figures.....	VI

CHAPTER 1 INTRODUCTION.....	- 1 -
1.1 MOTIVATION	- 1 -
1.2 PROBLEM STATEMENT	- 3 -
1.3 ACHIEVEMENTS	- 4 -
1.4 ORGANIZATION OF THE THESIS	- 5 -
CHAPTER 2 DEVELOPMENT OF VIRTUAL COLLABORATIVE MULTI-VEHICLE SIMULATION PLATFORM.....	- 6 -
2.1 BACKGROUND.....	- 6 -
2.2 COMPARISON OF 3D SIMULATORS	- 9 -
2.3 OVERVIEW OF PLATFORM DESIGN ARCHITECTURE	- 10 -
2.4 OVERVIEW OF VCMVSP FUNCTIONAL CONFIGURATION	- 13 -
2.5 SYNCHRONIZING VIRTUAL WORLD TO THE REAL WORLD	- 14 -
2.5.1 Unreal Editor and USARSim.....	- 14 -
2.5.2 Indoor and Outdoor Environment.....	- 15 -
2.5.3 Modeling of Vehicles	- 18 -
2.5.4 Selecting UAV and UGV Model	- 19 -
2.5.5 Defining Sensors.....	- 21 -
2.6 CENTRAL PROCESSING/INTELLIGENCE	- 21 -
2.7 SYNCHRONIZING REAL WORLD TO VIRTUAL WORLD (HYBRID CAPABILITY)	- 23 -
CHAPTER 3 EQUIPPING VCMVSP WITH ALGORITHM IMPLEMENTATIONS.....	- 26 -
3.1 IMPLEMENTATION 1: LEADER-FOLLOWER FORMATION ALGORITHM.....	- 26 -
3.1.1 Background Research on Formation Control.....	- 26 -
3.1.2 UGV Kinematic Model	- 27 -
3.1.3 UAV Kinematic Model.....	- 29 -
3.1.4 Explanation of Leader-Follower Formation Algorithm.....	- 34 -
3.1.5 Virtual World Setup vs Real World Setup for Implementation 1	- 39 -
3.1.6 Normal Simulation, Virtual Simulation and Real World Result Comparison	- 40 -
3.2 IMPLEMENTATION 2: OBSTACLE AVOIDANCE ALGORITHM	- 42 -
3.2.1 Explanation on Obstacle Avoidance Algorithm	- 42 -
3.2.2 Virtual World Setup	- 43 -
3.2.3 Result of Simulation in Virtual World	- 44 -
3.3 HOW SIMULATION CAN BE PERFORMED IN VIRTUAL WORLD	- 45 -
3.3.1 Simulating Effects of Gain on Formation	- 45 -
3.3.2 Simulating Effects of Momentum on Formation	- 46 -
3.3.3 Simulating Effects of Delay on Formation.....	- 47 -
CHAPTER 4 LOGIC BASED OBSTACLE AVOIDANCE.....	- 49 -
4.1 WHY LOGIC BASED OBSTACLE AVOIDANCE IS INTRODUCED INTO FORMATION	- 49 -
4.2 BACKGROUND RESEARCH ON OBSTACLE AVOIDANCE	- 49 -
4.3 KEY OBJECTIVES AND CHALLENGES	- 51 -
4.3.1 Navigating Through Narrow Path and Tendency of Getting into Dead Corners.....	- 52 -

4.3.2	<i>Real Time Processing Capability</i>	- 53 -
4.3.3	<i>Integration of Obstacle Avoidance into Formation</i>	- 53 -
4.4	OVERVIEW	- 54 -
4.5	DETAILED EXPLANATION.....	- 55 -
4.6	FORMATION WITH LOGIC BASED OBSTACLE AVOIDANCE IN VCMVSP.....	- 58 -
CHAPTER 5	IMPLEMENTATION OF UAV RECONNAISSANCE MISSION IN VCMVSP	- 60 -
5.1	OVERVIEW	- 60 -
5.2	SEARCH ALGORITHM.....	- 61 -
5.3	PATTERN RECOGNITION	- 63 -
5.3.1	<i>Optimizing Pattern Recognition for Search Algorithm</i>	- 63 -
5.4	INTEGRATION OF ALL ALGORITHMS.....	- 65 -
5.5	RESULT OF IMPLEMENTATION IN UNREAL VIRTUAL URBAN ENVIRONMENT	- 68 -
CHAPTER 6	FURTHER ENHANCING THE VCMVSP CAPABILITY	- 72 -
6.1	UPGRADING UNREAL ENGINE 2.5 TO UNREAL ENGINE 3	- 72 -
6.2	SIMULATING WIRELESS TRANSMISSION	- 74 -
6.2.1	<i>Introduction to OMNet++</i>	- 75 -
6.2.2	<i>Integration of OMNet++</i>	- 75 -
6.2.3	<i>Wireless Simulation Server</i>	- 76 -
CHAPTER 7	CONCLUSION AND POSSIBLE FUTURE WORKS.....	- 78 -
7.1	CONCLUSION	- 78 -
7.2	POSSIBLE FUTURE WORKS	- 79 -
AUTHOR'S PUBLICATIONS AND WORKS.....		- 80 -
REFERENCES.....		- 81 -

List of Figures

Figure 1. Typical Real Case: Algorithm Simulation with Matlab, Followed by Real World Testing.	- 7 -
Figure 2. Typical Process flow for Algorithm testing in Real World	- 8 -
Figure 3. Space Constraint Illustration on Stargazer Indoor Localization Setup	- 9 -
Figure 4. Overview of Hybrid Simulation Platform Design Architecture	- 12 -
Figure 5. Overview of Hybrid Simulator Selected Software Architecture	- 12 -
Figure 6. Illustration of VCMVSP Functional Configuration	- 14 -
Figure 7. Screen Capture of Unreal Editor	- 15 -
Figure 8. Real World Picture of Lab Environment	- 16 -
Figure 9. Lab Environment in Unreal World	- 17 -
Figure 10. Real World SRC (Top View) vs Unreal SRC	- 18 -
Figure 11. Real World UAV in comparison to Unreal World UAV	- 19 -
Figure 12. Motion Map of quadrotor Aircraft	- 20 -
Figure 13. Real World UGV in comparison to Unreal UGV	- 20 -
Figure 14. Overview of Role and Functionality of LabView and Unreal Engine	- 22 -
Figure 15. Real World Setup for Retrieving X-Y Coordinates	- 24 -
Figure 16. UWB Localization System Structure	- 25 -
Figure 17. Robots Implementation by different researchers [14], [15] & [16]	- 27 -
Figure 18. AmigoBot Kinematic Model	- 28 -
Figure 19. Quadrotor Frames	- 31 -
Figure 20. Three Vehicles Triangular Formation	- 35 -
Figure 21. Deriving Error Systems for New Coordinate System	- 38 -
Figure 22. Virtual World P2DX Formation	- 39 -
Figure 23. Real World Robot Formation	- 40 -
Figure 24. Pure Matlab simulation of Formation in Circular Movement (Matlab)	- 41 -
Figure 25. Virtual Simulation of Formation in Circular Movement (Matlab)	- 41 -
Figure 26. Real World X,Y Plot of Formation in Circular Movement (Excel)	- 42 -
Figure 27. Defining the Protection Radius	- 43 -
Figure 28. Single Block Obstacle Avoidance	- 44 -
Figure 29. Formation of 3 UAV with obstacle avoidance	- 45 -
Figure 30. Effects of tuning gain k_1	- 46 -
Figure 31. Effects of tuning gain k_2	- 46 -
Figure 32. Matlab simulation vs. Unreal Simulation	- 47 -
Figure 33. Changing of Sampling Time to Simulate Transmission Delay	- 48 -
Figure 34. Illustration of Problem Navigating through Narrow Path	- 52 -

Figure 35. Illustration of Tendency getting into Dead Corner	- 53 -
Figure 36. Illustration of Possible Follower Collision	- 54 -
Figure 37. Overview of Logical Based Obstacle Avoidance	- 54 -
Figure 38. Illustration of Positive and Negative Cutoff Crossing Point	-55 -
Figure 39. Illustration of Calculating True World Width	- 56 -
Figure 40. Illustration on possible available paths	- 58 -
Figure 41. Illustration of Path Chosen by Follower Robot	- 58 -
Figure 42. Testing of Formation with Logical Based Obstacle Avoidance in VCMVSP	- 59 -
Figure 43. X, Y coordinates plot from logged data in Virtual World	- 59 -
Figure 44. Overview of UAV Reconnaissance Mission	- 61 -
Figure 45. Search Algorithm	- 62 -
Figure 46. Sample illustration on Pattern Recognition	- 63 -
Figure 47. Enhanced Target Recognition Procedure	-65 -
Figure 48. Process Flow Chart for Algorithms Integration	- 67 -
Figure 49. Screen Capture of Autonomous Vehicle in Reconnaissance Mission – (a)	- 69 -
Figure 50. Screen Capture of Autonomous Vehicle in Reconnaissance Mission – (b)	- 70 -
Figure 51. Screen Capture of Autonomous Vehicle in Reconnaissance Mission – (c)	- 71 -
Figure 52. Comparing Usage of Google Sketchup for UT2004 and UDK	- 72 -
Figure 53. Comparing UDK and UT2004 3D Rendering	- 73 -
Figure 54. Quick Test of Obstacle Avoidance in UDK	- 74 -
Figure 55. Integration of OMNet++ and LabView	- 75 -
Figure 56. Communication between two UAVs with the use of WSS	- 76 -

Chapter 1 Introduction

1.1 Motivation

Potential applications for multi-autonomous vehicle systems can be rather wide. In space and aeronautics area, autonomous vehicle can be deployed into outer space or other planets for ground surveillance and data collection. In military area, bomb disposal, collaborative target search and air surveillance. In the future intelligent transport system, autonomous vehicle and even collaborative convoy system are currently in exploration. In civil defense, it was widely used for handling hazardous material and there are reports where snake like robot are deployed in disaster scene for search and rescue. The trend is towards cooperative control capabilities, and to realize all these applications, there are many different algorithms that need to be developed, enhanced and integrated among it. These algorithms include obstacle avoidance, formation control, rendezvous, cooperative search, leader and follower role assignment, target tracking, object recognition, stair climbing, etc.

The first purpose of this research is to develop a Virtual Collaborative Multi-Vehicle Simulation Platform (VCMVSP) for real and virtual simulation of cooperation among autonomous vehicles. Autonomous vehicle in the real world can be modelled and simulated in a total virtual environment. In a way, autonomous vehicle in real world and autonomous vehicle in a virtual world would be allowed to collaborate and be integrated together. This allows developers and researchers to test and develop algorithms meant for multiple UAVs or UGVs to achieve formation and obstacles avoidance with the virtual environment instead in real world. The aim here is to ensure the desired resulting algorithm needs no further amendment and could be easily implemented into real world robots to achieve the same intended purpose.

The second purpose of this research is to make use of the VCMVSP to implement the formation and coverage control. As the implementation involves a number of algorithms, formation control, coverage control, obstacle avoidance, tracking and pattern recognition, some of these algorithms would compete at the same time between each

Chapter 1: Introduction

other for the kinematics control of the vehicle. Thus the focus here would be on the strategy of switching between algorithms to allow smoother taking over. Special efforts are being spent here on the switching of algorithm between formation and obstacle avoidance, as they contribute more substantially to the kinematics control. This research presents a simple and computationally efficient obstacle avoidance algorithm that is based on logical understanding of the surrounding, and adaptively allows multi-vehicle to change formation on a real-time basis. It is also important to demonstrate that the implementation here satisfies the main objective of the first purpose.

Chapter 1: Introduction

1.2 Problem Statement

The Project Problem Statement is inferred from the desired functionality of the project application and the necessary control problems that must be addressed and various courses of actions to be taken to achieve the desired functionality.

The goal of this project is to build a VCMVSP that allows deploying and testing algorithms for multi-vehicle cooperation in virtual and real environment. The development includes both a basic and an advanced framework.

The main objectives of the basic VCMVSP framework include (Chapter 2):

- 1) Freedom in defining different environment, vehicles and sensors.
- 2) Adaptive to different algorithm with minimal change
- 3) Allows Virtual World developed algorithm to be easily implemented into Real World
- 4) Allows problems faced in Real World to be replicated in Virtual World for further troubleshoot.

The development of the advanced VCMVSP framework can be divided into several sub-goals. The completion of the collective sub-goals is necessary in order to develop a viable system control that will enable implementing the formation and coverage control into the VCMVSP and allows reconnaissance mission to be simulated in VCMVSP. The collective sub-goals of the project are listed below and serve as a course of action for the project.

- 1) To implement a formation algorithm into VCMVSP. The key requirement here is to ensure that the formation algorithm works exactly the same way in real world as in VCMVSP. (Chapter 3.1)
- 2) To implement an obstacle avoidance algorithm into VCMVSP. (Chapter 3.2)
- 3) Demonstrate how simulation can actually be performed in VCMVSP for analyzing how a parameter affects overall performance. (Chapter 3.3)

Chapter 1: Introduction

- 4) To introduce logic based obstacle avoidance algorithm into formation to reduce inflexibility in adapting to changing environment. (Chapter 4)
- 5) To implement a virtual mission based test scenario that involves coverage algorithm, logic based obstacle avoidance from item 3, tracking algorithm and pattern recognition. Through this implementation, it will fully demonstrate what the advanced VCMVSP framework is capable of. (Chapter 5)
- 6) At the same time in implementing item 5, it will tackle on the issue where different algorithms may compete for the kinematic control of the vehicle. (Chapter 5)

1.3 Achievements

- 1) By making use of the USARSIM, Unreal Editor, Unreal Engine, LabView and Matlab Script, we successfully created the VCMVSP.
- 2) Using the VCMVSP, existing formation control and obstacle avoidance algorithms has been successfully implemented in a hybrid manner. That is same algorithm can be run in a virtual environment clone of the real world, as well as the real world itself.
- 3) A logical base obstacle avoidance, which is very light in computation and capable of real-time obstacle avoidance, is newly introduced here.
- 4) An existing formation algorithm is further enhanced to be adaptive to the surrounding environment through introducing a simple and computationally efficient obstacle avoidance algorithm based on logical understanding of surrounding.
- 5) In the advanced usage VCMVSP, a reconnaissance mission has been implemented through integration of formation control, logical base obstacle avoidance, search algorithm, pattern recognition and target tracking. A systematic approach is adopted for the integration.
- 6) Our platform allows segregating out different algorithm's contention on the kinematic control of vehicle in a smoother switching manner.

1.4 Organization of the thesis

The organization of the thesis is as follows, Chapter 2 would discuss on the development of VCMVSP. Literature review will be shown; Concept of the whole design architecture will be explained with details of each portion further elaborated. Chapter 3 describes how formation and obstacle avoidance algorithm is being implemented into VCMVSP and the real world. Chapter 4 explains the need to introduce formation with logic based obstacle avoidance and provides detailed explanation on how this algorithm works. Comparison results are shown for both real world and virtual world. Chapter 5 shows how a real world reconnaissance mission based scenario, that includes various algorithms like formation, coverage, target tracking, pattern recognition and obstacle avoidance, can be implemented with VCMVSP, and the result of the implementation would be shown. Chapter 6 shows some further enhancement done on VCMVSP that is meant to increase its capability. Finally, Chapter 7 concludes the report and describes some possible future works.

Chapter 2 Development of Virtual Collaborative Multi-Vehicle Simulation Platform

2.1 Background

There are many simulation platforms that provide simple 2D visualization and easy means to manipulate and test algorithms. However, these platforms have a common shortfall that they are unable to simulate sufficiently the application of an algorithm in a real environment where they intend to use. Usually, when a real vehicle has been purchased, there could be problems that pertain to the size and response of the vehicle, which could result in the purchase of another vehicle or extensive modification to the algorithm. The whole process would be time-consuming and jeopardizes projects with short deadlines. In Figure 1 from reference [\[3\]](#), it shows a typical real case of doing algorithm simulation with Matlab followed by Real World testing. On the left of Figure 1 is a series of screen captures of a Box formation simulated with Matlab, and on the right is the real world setup. The intention of showing this simulation is to point out the shortcomings of general simulations. Firstly, the inability to simulate any non-idealities of the real-world would have impact on the overall performance.

Chapter 2: Development of VCMVSP

Figure 1. Typical Real Case: Algorithm Simulation with Matlab, Followed by Real World Testing.

Figure 2 shows the typical flow involved for experimental setup to verify the algorithm in the real world. With reference to what is shown in Figure 1, the experimental setup relies heavily on the overhead camera and the formation accuracy relies on the accuracy of camera pattern matching. As the whole setup was not tested outdoor, the feasibility of the formation algorithm in outdoor condition was not verified. Therefore, there is no environmental simulation element included for a general simulation.

Figure 2. Typical Process flow for Algorithm testing in Real World

Figure 3 below shows how a stargazer indoor localization system can be setup for a vehicle. There are many research studies [\[4\]](#) that utilize stargazer to do multi robot formation. The biggest issue here is the feasibility and difficulty of setting up in a bigger space, as many more passive landmarks would have to be deployed. As the virtual platform allows users to define their own space and details, this would easily solve the space and quantity issues faced by real world testing.

Chapter 2: Development of VCMVSP

Figure 3. Space Constraint Illustration on Stargazer Indoor Localization Setup

USARSim [\[5\]](#) is one of the most established performance evaluation tools of algorithms which provides a wide range of robots and sensors models and is able to give 3D visualization as it is built based on the Unreal game engine. This research makes use of USARSim as the ground basis and further extends its capability for use in developing the hybrid simulation platform.

2.2 Comparison of 3D Simulators

OpenSimulator (OpenSim) [\[6\]](#) can be considered as one of the earliest attempt for 3D robotic simulator in 2001. Unfortunately, since early 2006 improvements had been limited and by 2008 the whole project was suspended. It is still not strong in real-time rendering of robot environment and simulating dynamics.

Virtual Robot Experimentation Platform (V-REP) [\[7\]](#) is a commercial 3D Robot simulator. V-REP is not open source and simulation codes cannot be easily ported to Real World. This would restrict any possible further development required.

Modular OpenRobots Simulation Engine (MORSE) [\[8\]](#) is an open source 3D Robot simulator which makes use of Blender for 3D modeling and Bullet for physics

Chapter 2: Development of VCMVSP

simulation. It is developed and supported mostly on Linux and is not meant for hybrid simulation.

Gazebo [\[9\]](#) is yet another open source 3D Robot simulator and rather similar to MORSE. It makes use of OGRE for 3D modeling. Similarly, it is developed and supported mostly on Linux and is not meant for hybrid simulation as well.

Unified System for Automation and Robot Simulation (USARSim) [\[5\]](#) is a more established 3D Robot simulator. It makes use of Unreal Engine for 3D modeling and Karma physics Engine for physics simulation. Unreal Engine and Karma physics are well integrated to provide a better 3D modeling physics compared to Gazebo [\[9\]](#) and MORSE [\[8\]](#). It supports both Linux and Windows, thus allows interface to LabView. USARSim is chosen for this project.

2.3 Overview of Platform Design Architecture

This project has already been ongoing for almost reaching 3 years, and the basis of our VCMVSP was completed for almost 2 years. Our created platform has already been in use by several fellow NTU researchers and students for their use in developing algorithm. Nevertheless, I will still provide an explanation of the entire design in this section. The whole design architecture [\[1\]](#) here is being divided into four portions, namely Real World, Transmission, Central Processing/Intelligence & Interface to Virtual World and Virtual World as shown in Figure 4. The final selected software for each of these four portions are shown in Figure 5. For the real world, there are endless kind of sensors and robot models that we can use. In this research here the real world items that are used to develop the basic VCMVSP include UAVs, UGVs, UWB and UWB tags. The UAVs & UGVs servo motors, responsible for their motion, would be controlled by the onboard processor. Sensors that are attached for localization or navigation purpose would be controlled by their onboard processor as well. The UWB and UWB tags here are meant for localizing the UAVs' & UGVs' position. As the processing here would need to be fast, dedicated PC or Laptop for UWB is required. The final values that are

Chapter 2: Development of VCMVSP

passed to LabView from the UWB localization would thus only be the UAV & UGV location coordinates and orientation.

For the transmission portion, OMNet++ can be used to simulate the wireless transmission. In the case where real transmission is preferred over simulated, Wi-Fi, Zigbee or any fast and efficient wireless protocol can be used to establish the required TCP/IP connection. Though the simulation of wireless transmission has been included as part of the hybrid platform that we are going to build, however, since this is not the main focus of this thesis, it will not be discussed in much detail. In a later chapter, capability of OMNet++ in wireless simulation will be briefly explained.

For the portion on central Processing/Intelligence here, it refers to where all the key algorithm will be located and how it allow Interface to both the Virtual World and Real World. Each unit will consist of this platform capability, thus decentralize deployment can be achieved as well through exchanging of data for their respective intelligent processing. As LabView is a tool that supports wide range of sensor data acquisition and support embedded, it is preferred over other programming languages in terms of ease of deployment to the real world. Furthermore, on top of data acquisition, LabView also have wide range of data filtering techniques, many ready modules for charting display and it supports image and video processing as well. One further favorable feature in Labview is that it allows insertion of other programming languages. Thus in our design here, our mathematical model is done in a MatLab Script, as MatLab is well-known for handling all sorts of mathematical model, and this allows us to extract MatLab's capability for integration into our LabView program.

Chapter 2: Development of VCMVSP

Figure 4. Overview of Hybrid Simulation Platform Design Architecture

Figure 5. Overview of Hybrid Simulator Selected Software Architecture

Chapter 2: Development of VCMVSP

For virtual simulation, Unreal Engine and USARSim are used to do all the necessary simulations. Unreal Engine and USARSim are capable of simulating different UAVs & UGVs model, various kinds of sensors and is also able to simulate perspective of video cameras. On top of that, the use of Unreal Editor allows us to create a virtual world environment that is almost an exact replicate of the desired environment in the real world. Specific details like ground friction, exact dimension of the entire space, lightings and wind condition, different kinds of sensors, kinematics and dynamics of the vehicle can all be custom specified and integrated as part of the virtual environment.

2.4 Overview of VCMVSP Functional Configuration

After knowing the overall design architecture, next we will move on to discuss about the overview of VCMVSP functional configuration [\[1\]](#). It is generally difficult to implement the full dynamics of a physical UAV or UGV in Unreal, although Unreal Script provides the possibility. Hence, we realize the inner-loop control (rigid body dynamics and internal stability) in Matlab while Unreal only reflects the outer-loop control (Kinematic model). Figure 6 shows an illustration of the functional configuration for a UAV/UGV vehicle. Roughly speaking, the inner-loop guarantees the asymptotic stability and disturbance-rejection performance of the UAV/UGV vehicle, while the outer-loop response to the behaviour requests to achieve certain trajectory with a desired orientation. The data fusion can be of even more different kinds of sensors and different methods, Figure 6 shows only one of the possible ways. The way the interaction between the data fusion and mission control to determine the flight control, is the main focus of developing the advance VCMVSP. It will be further elaborated in later chapters.

Chapter 2: Development of VCMVSP

Figure 6. Illustration of VCMVSP Functional Configuration

2.5 Synchronizing Virtual World to the Real World

2.5.1 Unreal Editor and USARSim

Unreal Editor is an editor tool that can be used to create virtual environment for Unreal based on the Unreal Engine. Unreal Editor 2.5 is the version that this project is mainly using. Unreal Editor does not treat the virtual world as a giant space, but instead it treat as a solid space and user will define their own space through geometry subtraction. Basically, there are many different types of environment that can be created with Unreal Editor based on the Unreal Engine.

Chapter 2: Development of VCMVSP

Figure 7. Screen Capture of Unreal Editor

USARSim (Unified System for Automation and Robot Simulation) [5] is an existing tool that contains a number of simulation of commercially available robots and sensors for use in the environments based on the Unreal Tournament game engine. In the following section 2.5.2, we will present a few virtual world that is build with Unreal Editor, meant to simulate our desired real world testing. In 2.5.3 and 2.5.4, we will be discussing how we can actually construct our own robot with USARSim and Unreal Editor. However, as the desired kinematic are already available, the existing UAV and UGV models would be used instead. New constructions of the UAV and UGV models are only necessary if the desired kinematic is not available. The functionality of the existing sensor simulation models and their respective intended real sensor will be discussed in 2.5.5.

2.5.2 Indoor and Outdoor Environment

In order to model the environment in a more precise manner, it would be important to understand the scale and units used in Unreal. The unit used in Unreal is called an UU (Unreal Unit). Unreal Engine uses it to represent both length and angle. The exceptions in Unreal Engine are: 1) it uses degrees instead of UU to count FOV (Field of View); 2) in trigonometric functions, it uses radians. The unit conversion is basically $250 \text{ UU} = 1\text{m}$ and $32768 \text{ UU} = 3.1415 \text{ radian} = 180\text{degree} = 0.5 \text{ circle}$.

Chapter 2: Development of VCMVSP

For the Indoor Environment, the Nanyang Technological University Sensor Network Laboratory has been chosen for this project as the modeled environment. This model environment was created by a NTU student for use in his project [\[29\]](#). As this is an indoor environment, Universal Transverse Mercator offset would not be required, i.e. there is no real requirement of matching real world coordinate system with unreal. As for the room dimension, both have been set exactly to the same size. The Real World Lab Dimension is 16.4m x 9.5m x 2.6m, and the corresponding exact in Unreal World is 4100uu x 2375uu x 650uu

In Figure 8, it shows the actual real world Lab environment. In Figure 9, it shows the Lab environment in Unreal World. The lightings in Unreal World can be adjusted, and in the case of modeling here, the lightings are simulated based on daytime condition of the actual real world Lab environment. Tables and chairs are included as well to closely resemble the real world .

Figure 8. Real World Picture of Lab Environment

Chapter 2: Development of VCMVSP

Figure 9. Lab Environment in Unreal World

For Outdoor Environment, the Nanyang Technological University Student Recreation Centre has been chosen for this project as the modeled environment. The UTM offset parameters have not been utilized to align with the real world, as it is not required. The dimensions of the real world SRC are 300m x 250m x 40m and the corresponding Unreal World SRC dimension is 75000uu x 62500uu x 10000uu.

In Figure 10, it shows a side by side comparison of Real World SRC and Unreal SRC. As can be seen from both the indoor and outdoor virtual environments compared to their real world counterparts, the specific details of virtual environment can be easily adjusted in a way that it meets all the specific requirements of the project.

Figure 10. Real World SRC (Top View) vs Unreal SRC

2.5.3 Modeling of Vehicles

The modeling of vehicles is more complex and requires more attention in details as it would directly affect the kinematics and process of formula calculation in the intended application. Details on how to construct a generic vehicle or robot model will be explained in this section. Further details are available in this link http://usarsim.sourceforge.net/wiki/index.php/14.2_Advanced_User.

The following are a brief summary on how a robot can be constructed, more details can be found in chapter 14.4 of [5]. In the terms of robot modelling in USARSim, a robot can be constructed with four parts, namely chassis, parts, joints, and attached items. For chassis, by the name implies, it refers to the chassis of the robot. For parts, it refers to the motorized parts that are being used to construct a robot. For joints, it refers to the constraints that connect two parts together. Finally for attached items, it refers to all the auxiliary items attached to the robot.

After knowing that the robot consists of four parts, define and build the geometric model for all these four parts are required. Individual parts class are required for all these four parts, and on top of that, another separate class is required for the entire robot model. The class here is actually a Unreal Script where all the physical attributes of robot are

Chapter 2: Development of VCMVSP

programmatically defined. The physical connection relationship between the chassis, parts, joints and auxiliary items, must be specified and configured. More details of the UAV and UGV kinematics and dynamics model would be explained in Chapter 3.

2.5.4 Selecting UAV and UGV Model

In the left of Figure 11 below, it shows the GAUI 330X Quad Flyer. The Quad Flyer has four propellers and in the centre is where the payload will be situated. On the right of Figure 11 shows the Unreal World UAV which is modelled according to the Quad Flyer.

Figure 11. Real World UAV in comparison to Unreal World UAV

In Figure 12, it shows the general motion mapping of a quadrotor aircraft. In order to achieve a resulting up or down motion, all four thrust must maintain same speed and an equal increasing or decreasing speed will achieve respectively the up or down motion. For moving left or right and forward or backward, it is achieved through similar yaw motions. Differential control strategy is applied to each rotor generated thrust, and to counter the drift due to the reactive torques, the rotors must rotate in a specific manner. One pair of rotor, either the right and left or the front and back pair must rotate clockwise and the other pair will then rotate counterclockwise. Usually the configuration of which pair to rotate clockwise and which pair to rotate counterclockwise is fixed and the rotors rotating direction will have no more changes. In order to achieve a yaw motion, one pair of rotor will reduce the thrust and the other pair will increase. The resultant total thrust added up together must remain the same as before reducing and increasing, so as to prevent oscillation motion. For left side motion, it is achieved by

Chapter 2: Development of VCMVSP

increasing the left rotor thrust and decreasing the right rotor thrust. For right side motion, the opposite happens. Similarly to the yaw motion, the total thrust must remain constant to prevent oscillation. For forward motion, it is achieved by increasing the front rotor thrust and decreasing the back rotor thrust. For backward motion, the opposite happens. Again, the resultant total thrust must remain constant to prevent oscillation.

Figure 12. Motion Map of quadrotor Aircraft

The UGV model used is P2DX. P2DX is a 2-wheel drive plus a castor wheel pioneer robot from ActivMedia Robotics, LLC. The resulting kinematics can be considered very similar to the AmigoBot that will be used.

Figure 13. Real World UGV in comparison to Unreal UGV

Chapter 2: Development of VCMVSP

Figure 13 shows the side by side comparison of the real world UGV and Unreal UGV. The Unreal UGV model that is used here is direct from USARSim and not a model that is created by this project. Thus, there is no further elaboration of the UGV model explained here.

2.5.5 Defining Sensors

USARSim has quite extensive coverage of sensors, and every sensor type has a configuration file to allow us to edit base on the specification of the real hardware used. In fact, noise and distortion can be applied to the output values reported by the sensor in order to simulate noise and imperfections in the real world. In this report, only sensors that are being used in this project will be mentioned.

The first sensor consideration required would be sensors that are capable of helping to implement obstacle avoidance. The sensor that is being chosen here is range scanner sensor. The range scanner sensor simulation model is meant to simulate the LIDAR scanner LMS300. This LMS300 will be mounted on UGV, the FOV will be 180 degrees and the resolution will be 512 beams. As for the UAV, it is assumed a rotating range scanner can be mounted. It will be operated by continuously rotating in a fixed manner to obtain a series of data. As per actual testing with our simulation, the final beam resolution is 32 beams in one and a half circle. quadrotor UAV will be simulated here, thus the data from 360 degrees FOV is required. This is necessary as quadrotor UAV is capable of moving in all direction. For UGV, reverse motion will be restricted, thus only FOV of 180 degrees would be required. To replace the reverse motion, the UGV will do a static rotation instead.

2.6 Central Processing/Intelligence

As mentioned earlier, the programming language platform that is chosen in this project is LabView. The embedded Matlab Script will be used for executing algorithms. This As

Chapter 2: Development of VCMVSP

Matlab is still very popular among researchers and developers to do simulations; it would easily facilitate them to port the program over into LabView's embedded Matlab Script without making any changes. TCP/IP programming will be done on LabView to bridge the connection interface between the program's algorithm controls of the robot launched in the virtual world. In addition, the real world UGV hardware has been configured in such a way that it allows users to conveniently upload the completed program in a wireless manner. In the case where there is a lack of equipment or some vehicle is down, user can use the virtual vehicle to make up for the number. This is possible as the same set of program is able to control the real world robot as well as the virtual world robot.

In Figure 14, it gives the simplified overview of the overall role and functionality of LabView and Unreal Engine. Unreal Engine is capable of simulating camera on board of vehicle and feedback the video data to LabView. With these video data, object recognition programs can be established for tracking purposes.

Figure 14. Overview of Role and Functionality of LabView and Unreal Engine

Chapter 2: Development of VCMVSP

The screen capture from LabView in the bottom left of Figure 14 shows how LabView can retrieve the simulated sensor Lidar from Unreal Engine and use it for algorithm processing. Within the same screen capture, the box in the center is the Unreal Application camera view and on the bottom right shows the same screen with pattern matching algorithm running in the background.

As LabView supports running of Matlab Script by installing the necessary required module, the implementation of algorithm through Matlab Script in LabView can be easily achieved. The main idea here is to declare all variables required in the Algorithm as variable to be wired into Matlab Script and after calculation, the output can be wired out for further or next processing. Within the Matlab script itself, the exact same code that is written in Matlab can be written in this Matlab Script itself without any changes.

2.7 Synchronizing Real World to Virtual World (Hybrid Capability)

As mentioned that the intention of building VCMVSP, is to allow the capability of replacing real world up to a substantial extent. Thus the purpose of setting up the real world here, is to allow us to verify the performance of Algorithm in the Real World compared to the Virtual World. The overall real world setup can be broken into 3 portions as shown in Figure 15, first is UWB localization, second is Amigo robot and finally third is the use of Zigbee communication for wireless control of the Amigo robot.

Chapter 2: Development of VCMVSP

Figure 15. Real World Setup for Retrieving X-Y Coordinates

Commercially available UWB Localization system developed by the Ubisense Company is used. Four fixed sensors receive the ultra-wideband (UWB) radio signals emitted by location tags and use these signals to locate the positions of tags precisely. From Figure 16, the master sensor provides the timing source signal for synchronization of the slave sensors via a timing cable. The master sensor and slave sensor can be configured to exchange roles. The transmission delay via the timing cable is minimized as much as possible. A server runs the Ubisense platform server software to gather the data from fixed sensors. Ubisense tags can work with two non-overlapping radio channels or a single UWB channel. The difference between a dual channel and a single channel is that a dual channel has a bidirectional conventional telemetry channel. Each tag has its own identity number for identification.

Chapter 2: Development of VCMVSP

Figure 16. UWB Localization System Structure

Chapter 3 Equipping VCMVSP with Algorithm Implementations

3.1 Implementation 1: Leader-Follower Formation Algorithm

3.1.1 Background Research on Formation Control

Formation control can be defined as a group of robots or vehicles moving in a collaborative and autonomous manner, at the same time maintaining a pre-specified geometrical form with the aid of sensor feedbacks. Reference [10] has made a good study on the key concepts of different types of formation control and the background research here is done with reference to [10] as basis. It was mentioned that the current existing concepts of formation control can be classified into mainly three broad areas: leader-following [11], behavioural [12] and virtual structures [13]. For the leader-follower area, one of the robots will be pre-assigned as leader and the remaining will be follower. The actions of the leader robot will determine how the follower robots will response in a pre-specified manner. For the behavioural area, a set of basic behaviours are pre-defined and the overall weighted average of the control action for each vehicle's behaviour will determine the overall control action. In virtual structures area, the entire formation is considered as a rigid body, and each vehicle will be given a set of coordinates to follow. In this research, only leader-follower approach is used.

Fredslund and Mataric [14] show a behavioural approach on how can a group of distributed vehicles make use of local sensing and communication to achieve formation control. Das et al [15] proposed a switching paradigm and series of control algorithms which allows autonomous robots to preserve predetermined formation and allows changing of formation when the obstacles post a challenge to the predetermined formation. Considering the fact that when the formation loses the group leader, the entire leader-follower architecture group will fail. Thus, Sorensen [16] proposed a unified scheme which allows to the assignment of the group leaders to be arbitrary and

Chapter 3: Equipping VCMVSP with Algorithm Implementations

information exchange between the vehicles can be of arbitrary manner as well. All these three mentioned approach was experimentally implemented and validated on a multi-robot platform.

Figure 17. Robots Implementation by different researchers [\[14\]](#), [\[15\]](#) & [\[16\]](#)

As shown in Figure 17, typically all these researches require complicated and time consuming setup for testing and implementation of their formation algorithm. In the implementation 1 developed here, an attempt is made to show that the developed VCMVSP is capable of performing formation algorithm testing and can be ported directly to real world situation without involving any major changes.

3.1.2 UGV Kinematic Model

The UGV model used here consists of two fixed wheels and one castor wheel and the corresponding kinematic model can be derived from a standard procedure. This section

Chapter 3: Equipping VCMVSP with Algorithm Implementations

will discuss the kinematic model with reference from [17]. It is important to understand the kinematics as precise coding is required to implement this kinematics accurately in virtual world and real world. In Figure 18, it shows the kinematic model of AmigoBot. The (x_0, y_0) and (x_h, y_h) here are defined as the centre of two free wheels and the position of the caster wheel, separately. A line of length d intersects these two points. Usually, we control the robot as a point at (x_h, y_h) or the other point away from the central point (x_0, y_0) .

The dynamic kinematic model of Amigo is given by,

$$\begin{bmatrix} x_h(t) \\ y_h(t) \end{bmatrix} = \begin{bmatrix} x_0(t) \\ y_0(t) \end{bmatrix} + d \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \quad (3-1)$$

Here θ is the robot orientation with respect to the global reference frame.

Figure 18. AmigoBot Kinematic Model

Chapter 3: Equipping VCMVSP with Algorithm Implementations

The derivative of the central point is shown as

$$\begin{bmatrix} \frac{dx_0(t)}{dt} \\ \frac{dy_0(t)}{dt} \end{bmatrix} = v(t) \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \quad (3-2)$$

where $v(t)$ is the robot straight-line velocity that can be separated into two components along the x and y axis. By differentiating equation (2-1), the following equation is obtained,

$$\begin{aligned} \begin{bmatrix} u_x(t) \\ u_y(t) \end{bmatrix} &= \begin{bmatrix} \frac{dx_h(t)}{dt} \\ \frac{dy_h(t)}{dt} \end{bmatrix} = v(t) \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} + d\omega(t) \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix} \\ &= \begin{bmatrix} \cos(\theta) & -d\sin(\theta) \\ \sin(\theta) & d\cos(\theta) \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \end{aligned} \quad (3-3)$$

Here u_x and u_y are velocities at the caster wheel point along X and Y axes, respectively. Most of the time, the straight-line velocity $v(t)$ and rotational velocity $w(t)$ are used to control the robot as command inputs. This is represented by the equation (2-4) below:

$$\begin{aligned} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} &= \begin{bmatrix} \cos(\theta) & -d\sin(\theta) \\ \sin(\theta) & d\cos(\theta) \end{bmatrix}^{-1} \begin{bmatrix} u_x(t) \\ u_y(t) \end{bmatrix} \\ &= \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ \frac{-\sin(\theta)}{d} & \frac{\cos(\theta)}{d} \end{bmatrix} \end{aligned} \quad (3-4)$$

3.1.3 UAV Kinematic Model

The procedures of deriving the quadrotor kinematics are standard. In this section, the quadrotor kinematic models will be discussed with reference to [\[18\]](#). As mentioned in

Chapter 3: Equipping VCMVSP with Algorithm Implementations

previous section, understanding of the Kinematic Model is required as corresponding coding would be required for precise implementation.

Two reference frames are being here, namely

- Earth inertial reference (E-frame, Right hand Reference)
- Body-fixed reference (B-frame, Right hand Reference)

The E-frame basically consists of o_E , x_E , y_E , and z_E , where o_E is the axis origin, x_E refers to the direction toward North, y_E is the direction toward West and z_E points upwards with respect to the earth. The linear position ($\Gamma^E [m]$) and the angular position ($\Theta^E [rad]$) of the quadrotor would be defined with respect to this frame

The B-frame basically consists of o_B , x_B , y_B , and z_B , where o_B is the axis origin, x_B refers to the direction toward the front of quadrotor, y_B is the direction toward the left of quadrotor and z_B points upward. The o_B here is chosen such that it will coincide with the center of the quadrotor cross structure. The forces ($F^B [N]$), torques ($\tau^B [N\ m]$), linear velocity ($V^B [m\ s^{-1}]$) and angular velocity ($\omega^B [rad\ s^{-1}]$) would be defined with respect to this frame.

With respect to E-Frame, the linear position Γ^E of the quadrotor is derived by the coordinates of the vector between the origin of B-frame and E-frame according to equation (3-5). This is illustrated in Figure 19.

$$\Gamma^E = [X\ Y\ Z]^T \quad (3-5)$$

Figure 19. Quadrotor Frames

With respect to E-frame, the angular position Θ^E of the quadrotor is defined by the orientation of B-frame. With respect to the three rotations about the main axes for taking E-frame into B-frame, the set of "roll-pitch-yaw" Euler angles were used. Equation (3-6) shows the angular position vector.

$$\Theta^E = [\phi \ \theta \ \psi] \quad (3-6)$$

The rotation matrix $R_{\Theta}[-]$ can be obtained by doing a post-multiply of the three basic rotation matrices in the following order shown: (*the notation $c_k = \cos k$, $s_k = \sin k$, $t_k = \tan k$ has been adopted here*)

- Rotation about the z_E axis of the angle ψ (yaw) through $R(\psi, z) [-]$

$$R(\psi, z) = \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3-7)$$

Chapter 3: Equipping VCMVSP with Algorithm Implementations

- Rotation about the y_1 axis of the angle θ (pitch) through $R(\theta, y)$ [–]

$$R(\theta, y) = \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \quad (3-8)$$

- Rotation about the x_2 axis of the angle ϕ (pitch) through $R(\phi, x)$ [–]

$$R(\phi, x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi & c_\phi \end{bmatrix} \quad (3-9)$$

Equation (3-10) shows the composition of the rotating matrix R_Θ

Chapter 3: Equipping VCMVSP with Algorithm Implementations

$$\begin{aligned}
 R_{\ominus} &= R(\psi, z) R(\theta, y) R(\phi, x) \\
 &= \begin{bmatrix} c_{\psi}c_{\theta} & -s_{\psi}c_{\phi} + c_{\psi}s_{\theta}s_{\phi} & s_{\psi}s_{\phi} + c_{\psi}s_{\theta}c_{\phi} \\ s_{\psi}c_{\theta} & c_{\psi}c_{\phi} + s_{\psi}s_{\theta}s_{\phi} & -c_{\psi}s_{\phi} + s_{\psi}s_{\theta}c_{\phi} \\ -s_{\theta} & c_{\theta}s_{\phi} & c_{\theta}c_{\phi} \end{bmatrix} \quad (3-10)
 \end{aligned}$$

As stated before, the linear V^B and the angular ω^B velocities are expressed in the body-fixed frame. Their compositions are defined according to equations (3-11) and (3-12).

$$V^B = [u \ v \ w]^T \quad (3-11)$$

$$\omega^B = [p \ q \ r]^T \quad (3-12)$$

It is possible to combine linear and angular quantities to give a complete representation of the body in the space. Two vectors, can be thus defined: the generalized position $\xi[+]$ and the generalized velocity $v[+]$, as reported in equations (3-13) and (3-14).

$$\xi = [\Gamma \ \Theta^E]^T = [X \ Y \ Z \ \phi \ \theta \ \psi]^T \quad (3-13)$$

$$v = [V^B \ \omega^B]^T = [u \ v \ w \ p \ q \ r]^T \quad (3-14)$$

The relation between the linear velocity in the body-fixed frame V^B and that one in the earth frame $V^E[ms^{-1}]$ (or $\dot{\Gamma}^E[ms^{-1}]$) involve the rotation matrix $R_{\ominus}$ according to equation (3-15).

$$V^E = \dot{\Gamma}^E = R_{\ominus} V^B \quad (3-15)$$

As for the linear velocity, it is also possible to relate the angular velocity in the earth frame (or Euler rates) $\dot{\Theta}^E[rad\ s^{-1}]$ to that one in the body-fixed frame ω^B thanks to the transfer matrix $T_{\ominus}[-]$. Equations (3-16) and (3-17) show the relation specified above.

Chapter 3: Equipping VCMVSP with Algorithm Implementations

$$\omega^B = T_{\ominus}^{-1} \dot{\ominus}^E \quad (3-16)$$

$$\dot{\ominus}^E = T_{\ominus} \omega^B \quad (3-17)$$

The transfer matrix $T_{\ominus}$ can be determined by resolving the Euler rates $\dot{\ominus}^E$ into the body-fixed frame as shown in equations (3-18), (3-19) and (3-20).

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \emptyset \\ 0 \\ 0 \end{bmatrix} + R(\emptyset, x)^{-1} + \begin{bmatrix} 0 \\ \dot{\emptyset} \\ 0 \end{bmatrix} + R(\emptyset, x)^{-1} R(\emptyset, y)^{-1} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} = T_{\ominus}^{-1} \begin{bmatrix} \dot{\emptyset} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (3-18)$$

$$T_{\ominus}^{-1} = \begin{bmatrix} 1 & 0 & -s_{\emptyset} \\ 0 & c_{\emptyset} & c_{\emptyset} s_{\emptyset} \\ 0 & -s_{\emptyset} & c_{\emptyset} c_{\emptyset} \end{bmatrix} \quad (3-19)$$

$$T_{\ominus} = \begin{bmatrix} 1 & s_{\emptyset} t_{\emptyset} & c_{\emptyset} t_{\emptyset} \\ 0 & c_{\emptyset} & -s_{\emptyset} \\ 0 & s_{\emptyset}/c_{\emptyset} & c_{\emptyset}/c_{\emptyset} \end{bmatrix} \quad (3-20)$$

It is possible to describe equations (3-15) and (3-17) in just one equivalence which relate the derivation of the generalized position in the earth frame $\dot{\xi}[+]$ to the generalized velocity in the body frame v . The transformation is possible thanks to the generalized matrix $J_{\ominus}[-]$. In this matrix, the notation $O_{3 \times 3}$ means a sub-matrix with dimension 3 times 3 filled with all zeros. Equations (3-21) and (3-22) show the relation described above.

$$\dot{\xi} = J_{\ominus} v \quad (3-23)$$

$$J_{\ominus} = \begin{bmatrix} R_{\ominus} & O_{3 \times 3} \\ O_{3 \times 3} & T_{\ominus} \end{bmatrix} \quad (3-24)$$

3.1.4 Explanation of Leader-Follower Formation Algorithm

Chapter 3: Equipping VCMVSP with Algorithm Implementations

The existing formation controller design that is used for implementation can be found in reference [10]. The following is discussed with reference to [10] on how the control input, velocity and angular velocity can be derived.

The following defines the dynamic of vehicles:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\phi}(t) \end{bmatrix} = \begin{bmatrix} \cos\phi(t) & 0 \\ \sin\phi(t) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \quad [3-25]$$

With respect to an inertial coordination frame, x and y stands for the position of vehicle and ϕ stands for the orientation of vehicle. The v here denotes the linear velocity and ω denotes the angular velocity.

Figure 20. Three Vehicles Triangular Formation

In Figure 20, it shows a three vehicle triangle formation where R_l and R_f indicate leader and one of the follower respectively. To stay consistent with such a formation, follower R_f would need to maintain a desired ρ^d distance at a desired angle ϕ^d , with respect to the leader R_l .

A imaginary handling point h , which is off-the-axis, is defined to be located at a distance L from the centre of R_f . An imaginary virtual leader named R_v , is also defined to be located at the line which is perpendicular to the orientation of R_l , where $L = \rho^d \cos \phi^d$, and $L' = \rho^d \sin \phi^d$. The following equations can then be defined:

$$x_h = x_f + L \cos \phi_f$$

Chapter 3: Equipping VCMVSP with Algorithm Implementations

$$\begin{aligned} y_h &= y_f + L \sin \phi_f \\ \phi_h &= \phi_f \end{aligned} \quad [3-26]$$

$$\begin{aligned} x_v &= x_l + L' \cos(\phi_l - \frac{\pi}{2}) \\ y_v &= y_l + L' \sin(\phi_l - \frac{\pi}{2}) \\ \phi_v &= \phi_l \end{aligned} \quad [3-27]$$

After derivation, we got:

$$\begin{aligned} \dot{x}_h &= v_f \cos \phi_f - L \omega_f \sin \phi_f \\ \dot{y}_h &= v_f \sin \phi_f - L \omega_f \cos \phi_f \\ \dot{\phi}_h &= \omega_f \end{aligned} \quad [3-28]$$

$$\begin{aligned} \dot{x}_v &= v_l \cos \phi_l - L' \sin(\phi_l - \frac{\pi}{2}) \omega_l \\ \dot{y}_v &= v_l \sin \phi_l - L' \cos(\phi_l - \frac{\pi}{2}) \omega_l \\ \dot{\phi}_v &= \omega_l \end{aligned} \quad [3-29]$$

where $(v_l \ \omega_l)$, $(v_f \ \omega_f)$ is the linear velocity and angular velocity of leader and follower, respectively.

By subtracting R_v and h , that is [3-29] – [3-28], tracking error can be obtained,

$$\begin{aligned} \dot{\bar{x}} &= v_l \cos \phi_l - v_f \cos \phi_f - L' \sin(\phi_l - \frac{\pi}{2}) \omega_l + L \omega_f \sin \phi_f \\ \dot{\bar{y}} &= v_l \sin \phi_l - v_f \sin \phi_f - L' \cos(\phi_l - \frac{\pi}{2}) \omega_l - L \omega_f \cos \phi_f \\ \dot{\bar{\phi}} &= \omega_l - \omega_f \end{aligned} \quad [3-30]$$

where $\bar{x} = x_v - x_h$, $\bar{y} = y_v - y_h$, $\bar{\phi} = \phi_v - \phi_h$

Chapter 3: Equipping VCMVSP with Algorithm Implementations

Transform into a new coordination system, z_1, z_2 is the position error between R_v and h :

$$\begin{aligned} z_1 &= \bar{x} \cos \bar{\varphi}_f + \bar{y} \sin \bar{\varphi}_f \\ z_2 &= \bar{x} \sin \bar{\varphi}_f + \bar{y} \cos \bar{\varphi}_f \end{aligned} \quad [3-31]$$

The tracking error in new coordinate system now becomes:

$$\begin{aligned} \dot{z}_1 &= \dot{\bar{x}} \cos \bar{\varphi}_f - \bar{x} \sin \bar{\varphi}_f \omega_f + \dot{\bar{y}} \sin \bar{\varphi}_f + \bar{y} \cos \bar{\varphi}_f \cdot \omega_f = -v_f + v_l \cos \bar{\varphi} - z_2 \omega_f - L' \sin \left(\bar{\varphi} - \frac{\pi}{2} \right) \omega_l \\ \dot{z}_2 &= \dot{\bar{x}} \sin \bar{\varphi}_f + \bar{x} \cos \bar{\varphi}_f \omega_f + \dot{\bar{y}} \cos \bar{\varphi}_f - \bar{y} \sin \bar{\varphi}_f \cdot \omega_f = -v_l \sin \bar{\varphi} + z_1 \omega_f + L \omega_f - L' \cos \left(\bar{\varphi} - \frac{\pi}{2} \right) \omega_l \end{aligned} \quad [3-32]$$

Applying negative feedback rule, position error would converge to 0 and thus tracking error is minimized, the following control input is then defined:

$$\begin{aligned} v_f &= v_l \cos \bar{\varphi} + k_1 z_1 + \rho^d \omega_l \sin(\varphi^d + \bar{\varphi}) - L' \sin \left(\bar{\varphi} - \frac{\pi}{2} \right) \omega_l \\ \omega_f &= \frac{1}{L} \left[v_l \sin \bar{\varphi} + k_2 z_2 + L' \cos \left(\bar{\varphi} - \frac{\pi}{2} \right) \omega_l \right] \end{aligned} \quad [3-33]$$

We obtain following expression from Figure 21,

$$\begin{aligned} z_1 &= \rho \cos \varphi + L' \cos \left(\bar{\varphi} - \frac{\pi}{2} \right) - L \\ z_2 &= \rho \sin \varphi + L' \sin \left(\bar{\varphi} - \frac{\pi}{2} \right) \end{aligned} \quad [3-34]$$

$$v_f = v_l \cos \bar{\vartheta} + k_1 \left(\rho \cos \varphi + L' \cos \left(\bar{\vartheta} - \frac{\pi}{2} \right) - L \right) + \rho^d \omega_l \sin(\varphi^d + \bar{\vartheta}) - L' \sin \left(\bar{\vartheta} - \frac{\pi}{2} \right) \omega_l \quad [3-35]$$

$$\omega_f = \frac{1}{L} \left[v_l \sin \bar{\vartheta} + k_2 \left(\rho \sin \varphi + L' \sin \left(\bar{\vartheta} - \frac{\pi}{2} \right) \right) + L' \cos \left(\bar{\vartheta} - \frac{\pi}{2} \right) \omega_l \right]$$

Theorem: For the position error system described in (3-31), applying the tracking controller (3-33) to the follower, position errors z_1, z_2 will converge asymptotically to 0. This is based on assuming the leader's translational velocity is lower bounded and rotational velocity bounded, that is $v_l > 0$ and $|\omega_l| < K$.

$$\begin{aligned}\dot{z}_1 &= -k_1 z_1 - z_2 \omega_f \\ \dot{z}_2 &= z_1 \omega_f - k_2 z_2 \\ \dot{\bar{\phi}} &= -\frac{1}{L} \left[v_1 \sin \bar{\phi} - k_2 z_2 + L' \cdot \omega_1 \cos \left(\bar{\phi} - \frac{\pi}{2} \right) \right] + \omega_1\end{aligned}\quad [3-36]$$

Chapter 3: Equipping VCMVSP with Algorithm Implementations

Considering the Lyapunov function candidate:

$$V = \frac{1}{2}(z_1^2 + z_2^2)$$

where $V \geq 0$, and $V = 0$ only when $z_1 = z_2 = 0$. By using [3-35], we would obtain

$$\dot{V} = z_1\dot{z}_1 + z_2\dot{z}_2 = -k_1z_1^2 - k_2z_2^2$$

Thus we can see that only when $z_1 = z_2 = 0$, $\dot{V} \leq 0$ would be true and thus the position error system $\dot{z}_1, \dot{z}_2$ is shown to be asymptotically stable.

3.1.5 Virtual World Setup vs Real World Setup for Implementation 1

Figure 22. Virtual World P2DX Formation

Details on the setup have been explained in the previous chapters, and in the setup here, three P2DX robots formation has been simulated in the Virtual world. As shown in Figure 22, on the left, we can see that the P2DX robots are moving in formation and on the right are the X and Y coordinates and the angles of each robot extracted from their position in the Virtual world.

Chapter 3: Equipping VCMVSP with Algorithm Implementations

Figure 23. Real World Robot Formation

The real world setup here involves the use of UWB indoor localization to provide the X, Y and angle of each robot. The robot here that is being used is commercially available AmigoBot. Both the AmigoBot and UWB indoor localization system has been explained in Chapter 3. Figure 23 shows the AmigoBot moving in formation and its X,Y coordinates plot shown beside it.

In comparison between virtual world and real world setup to test the Leader-Follower Formation Algorithm, the virtual world is more effortlessly done.

3.1.6 Normal Simulation, Virtual Simulation and Real World Result Comparison

The performance comparison done here is between pure Matlab simulations versus the virtual simulation that makes use of Unreal. Figure 24 shows the resultant X, Y coordinates plot of formation moving in a circular direction under pure Matlab simulation. Whereas in Figure 25, it shows the X, Y coordinates plot of formation moving in a circular direction under virtual simulation. Figure 26 shows the resultant X, Y plot in the real world. The two follower movements are more unstable primarily due to the inaccuracy of UWB localization. The leader is more stable as it is a pre-defined circular movement. By comparing the movement result of leader in real world with the

Chapter 3: Equipping VCMVSP with Algorithm Implementations

movement in virtual world, we can conclude that virtual simulation is closer to the real world as it is capable of simulating the momentum of the engine as well.

The result of this test bed setup is considered successful as no major changes is required to get the formation algorithm to work in both real world and unreal world.

Figure 24. Pure Matlab simulation of Formation in Circular Movement (Matlab)

Figure 25. Virtual Simulation of Formation in Circular Movement (Matlab)

Figure 26. Real World X,Y Plot of Formation in Circular Movement (Excel)

3.2 Implementation 2: Obstacle Avoidance Algorithm

One of the simplest ways of achieving obstacle avoidance is to do path planning. The drawback of this method is that the environment needs to be well known. To acquire information of the environment, range scanner, ultrasonic, camera, and many other types of sensors can be used. In this implementation 2, Potential field method coupled with range scanner for obstacle avoidance is used for obstacles avoidance. Potential field method is commonly used in unknown environments as it is capable of treating the whole as a potential field. Conventionally, the goal point would appear as an attractive force to vehicles and obstacle appear as a repulsive force to vehicles. By defining repulsive force to be inversely proportional to the distance of obstacles, it would allow vehicles to avoid collision with obstacles, as the repulsive would push it away.

3.2.1 Explanation on Obstacle Avoidance Algorithm

Typically, the obstacles locations are unknown to the vehicles, and it would attempt to locate these obstacles through the use of sensors. In this implementation, range scanner sensor will be used to detect the distance between the UAV's range scanner location and the obstacles location. By applying the attractive force and repulsive force

Chapter 3: Equipping VCMVSP with Algorithm Implementations

concept combined with the scanner sensor information, the vehicle would naturally generate a path to navigate pass obstacles and move towards goal. As usually computation would take time, a response range, R , would need to be defined in such a way that it gives enough time for proper path to be generated, just before collision. To further prevent collision, a limit range, r , would be defined in such a way that once reached, it would increase drastically large to push the vehicle directly away. Figure 27, shows that the limit range is defined to be the radius just enough to cover the robot's body. The response range radius would then be much bigger than the limit range, depending on the computation speed.

Figure 27. Defining the Protection Radius

The repulsive force constant, F_c , can thus be computed as follow:

$$F_c = \frac{R^2 - x^2}{x^2 - r^2}$$

As mentioned, R and r would stand for the response and limit range respectively and x stands for the distance between the centre of vehicle and obstacles, assuming that range scanner is mounted on the centre of vehicle.

3.2.2 Virtual World Setup

The virtual sensor network lab has been used in this setup. In order for the leader to travel at a constant velocity in the virtual world, a destination point for the leader is

Chapter 3: Equipping VCMVSP with Algorithm Implementations

defined. Alternatively, manual control of the leader can be done as well. In the middle of the Virtual Lab, an obstacle pillar of half square meter has been inserted, it will generate a repulsive force to vehicles. As mentioned in the previous section, a respond range has been set such that only when the vehicle hits beyond, a repulsive force will then take effect. The repulsive force will become infinitely large when vehicle hits the limit range. The simulated range scanner resolution here has 32 beams and maximum range is set to 6m. The response range R has been set to 1.2m, limit range r set to 0.6m, sampling time is 10Hz.

3.2.3 Result of Simulation in Virtual World

In this section and the next few sections, with the help of student [Tonglin] conducting the experiments, the simulation result and parameters testing are collected and presented here. In Figure 28, it shows the X and Y plot of how a single vehicle would response to the obstacle. The blue trail here indicates the movement of the vehicle at the update rate of 10Hz. 0.1s, and the black square represents an obstacle. As there is no goal defined, the obstacle simply pushes the vehicle away to one side, instead of circumventing through the obstacle.

Figure 28. Single Block Obstacle Avoidance

In Figure 29, it shows the simulation result of combining formation algorithm with obstacle avoidance. The green and red trail represent the followers and the blue trail

Chapter 3: Equipping VCMVSP with Algorithm Implementations

here represents the leader. The black square block is the obstacle and the goal is defined as the red dot. As the red dot can be considered rather far away, thus the circumventing effect here is not strong. Nevertheless, it shows that both implementation 1 and implementation 2 has both been successfully integrated and implemented with VCMVSP architecture.

Figure 29. Formation of 3 UAV with obstacle avoidance

3.3 How Simulation Can Be Performed in Virtual World

Now that we have already implemented both Formation and obstacle avoidance, the next thing would of course be how to conduct the simulation to analyze the results of adjusting certain parameters. Three simple demonstrations will be shown here on how individual parameters can affect the performance of formation.

3.3.1 Simulating Effects of Gain on Formation

The parameters gain k_1 and k_2 of (3-35) has direct impact on the X and Y velocity of the followers. An effort will be made here to show the actual X and Y coordinate plot versus the changes of gain k_1 and gain k_2 . From Figure 30 and Figure 31, it can be

Chapter 3: Equipping VCMVSP with Algorithm Implementations

clearly seen that by adjust the values of gain k_1 and k_2 , overshoot and undershoot behaviour would be demonstrated.

Figure 30. Effects of tuning gain k_1

Figure 31. Effects of tuning gain k_2

3.3.2 Simulating Effects of Momentum on Formation

The simulated vehicle has been defined to have the same weight as the real vehicle, thus a momentum would exist when the vehicle is moving. The purpose of this simulation here is to show that the Momentum simulated in VCMVSP, would be more comprehensive compared to just a pure Matlab simulation. In this simulation, the X direction velocity of the leader has been set to 30cm/s and Y direction velocity has been set to 0. Update rate remains at 10Hz. Both Matlab and VCMVSP have the same

Chapter 3: Equipping VCMVSP with Algorithm Implementations

mentioned settings. We can clearly see from Figure 32 that the followers took a longer time to stabilize due to the impact of momentum, which is not handle in pure Matlab simulation.

Figure 32. Matlab simulation vs. Unreal Simulation

3.3.3 Simulating Effects of Delay on Formation

In the real world, there are mainly two areas which would cause significant delay, namely hardware computation delay and wireless transmission delay. The best way to simulate the effect of these delays would be to adjust the sampling speed. Rationale here is that the hardware computation delay and wireless transmission delay would be more or less consistent. Thus by adjusting the sampling frequency appropriately, these delays can be simulated as well

Sampling frequency of 10Hz, 2Hz, 1Hz and 0.5Hz has been used. In Figure 33, the X and Y coordinates plot of the leader and follower robots are shown. The cyan, green, blue, and red circles represents the sampling frequency of 10Hz, 2Hz, 1Hz, and 0.5Hz respectively. The expected overshoot and undershoot behaviour is again clearly seen.

Figure 33. Changing of Sampling Time to Simulate Transmission Delay

Chapter 4 Logic Based Obstacle Avoidance

4.1 Why Logic Based Obstacle Avoidance is introduced into Formation

Usually obstacle avoidance and formation are integrated together through some switching method. When there are no obstacles, formation algorithm will take control, and when obstacles are nearby, obstacle avoidance will take control for individual unit. Thus when obstacle avoidance is in control, the formation will definitely not be maintained as desired.

Another method is the reliance of mapping and path planning to integrate obstacle avoidance and formation. The main problem here is that mapping takes time to generate the occupancy grid, thus it will not be applicable to dynamic obstacle.

The approach here tries to take the best part from the above two methods. Instead of doing mapping and try to generate the occupancy, it will rely heavily on a high resolution range scanner sensor to know the surrounding and calculate very rapidly without the need of doing mapping. It will not entirely take control, as it will take into consideration of the formation required movement instantaneously, and changes formation if it requires adapting to the environment. In the following sections, explanation on how this can be achieved will be shown in details.

4.2 Background Research on Obstacle Avoidance

There are already enormous amount of great works and studies that have been done on obstacle avoidance. Reference [\[19\]](#) has made a good summary on the available methods, and the background research here is done with reference [19] as basis. Obstacle avoidance can be broadly categorized into three classic methods. They are edge-detection, certainty grids, and potential field methods. Edge detection, by its name implies, the vertical edges of obstacles are extracted and the robot will navigate around

Chapter 4: Logic Based Obstacle Avoidance

either one of the detected edges. Usually sensors like ultrasonic or range scanner are used for edge detection method. The main problem faced by edge detection is that the range scanner typically fails to correctly detect sharp edges and certain types of material edges. . As for ultrasonic sensor, frequent misreading may occur due to various reasons. For Certainty grid [\[20\]](#), it classifies the entire robot's surrounding space as an obstacle probabilistic map representation. Due to the fact that it uses a grid-type world model, it would be suitable for sensor data accumulation and fusion. The grid-type world model is actually a 2-D array of cells, with each cell containing a probabilistic value. This value would be a measure of confidence level where obstacle would exist in that cell area, base on sensor characteristics and readings. When the robot moves across an area, the cells would have been updated with the probabilistic value by the sensor readings, thus a fairly accurate map of the obstacle locations can be constructed. The main drawback of this method is that the accuracy depends on the cell size and mapping is required, thus it cannot really be considered real time. The accuracy of this method would depend on the resolution of the cell size defined. This would be impractical, as a very accurate map would require a very high resolution cell size, thus computation would be significantly slowed down. The potential field method has been mentioned in previous chapters.

In reference [\[21\]](#), it was mentioned that the potential field method was improved by integrating with the concept of certainty grid. Potential field is applied to a 2-D Cartesian histogram grid, which represents the probability of whether each cell contains obstacle. This method is called the Virtual Force Field (VFF) method. Conventionally, all potential field methods face the same problem, which is they are not capable of moving through narrow passage. The edges would pose as repulsive and blocks the passage path by pushing the robot away.

Reference [\[22\]](#) also mentioned a modified method based on the certainty grid map was introduced by using a Polar histogram (α -P) instead of 2-D Cartesian, making use of sensor readings to create the map. This method is called the Vector Field Histogram (VFH) method. The α and P here represents the sensor angle and the probability of

Chapter 4: Logic Based Obstacle Avoidance

obstacle presence in that direction respectively. By setting a threshold, values that are above it would mean that the representing directions are obstacle-free. VFH+ [23] is an incremental improvement of VFH by utilising the basic robot kinematics limitations to compute the robot possible trajectories using arcs or straight lines. *VFH** [23] is a further improvement of the VFH by introducing the idea of trajectory planning for a longer distance.

Dynamic Window Approach (DWA) [24] and Global Dynamic Window Approach (GDWA) [25] is an improvement to DWA. Both methods made use of the vehicle kinematics to generate all possible velocity paths. Improvement for GDWA is that it made use of the certainty grid cells to choose a better path that leads to goal.

Both the GDWA and VFH+ methods are able to satisfy their purpose of avoiding obstacles reasonably well. However, both require some form of sensor mapping and calculations before making any movement. During the process of mapping, it has no capability of avoiding moving obstacles. In comparison to the potential field method, this method allows us much better real time response as it does not require any mapping. The only drawback, as previously mentioned, is the inability to navigate through narrow paths. This research aims to propose a new algorithm that is both able to navigate through narrow passage and capable of doing real time obstacle avoidance.

When trying to integrate obstacle avoidance into multi-vehicle formation, it is very common to adopt the method of switching to obstacle avoidance mode in the presence of obstacle and back to formation mode when obstacle is cleared. The drawback of this approach is that in an environment where obstacles are always nearby, e.g. maze, the obstacle avoidance mode will always be triggered and thus formation will not be sustained. The proposed new algorithm aims to overcome this problem as well.

4.3 Key Objectives and Challenges

There are a few key challenging areas identified below,

Chapter 4: Logic Based Obstacle Avoidance

- i) Difficulty in navigating through a narrow path and tendency of getting into dead corners
- ii) For real time processing, reliance of mapping must be eliminated.
- iii) Integrating obstacle avoidance into formation.

4.3.1 Navigating Through Narrow Path and Tendency of Getting into Dead Corners

Typically in obstacle avoidance, it is hard to navigate through a narrow path round a corner. For the potential field method, as shown in Figure 34, the two edges will act as repulsive force and prevent it from entering. For edge detection, as shown in Figure 35, if there are openings on the side parallel to the robot, chances that the robot will miss is very high. Though the VFH* is able to tackle the two problems, however it is cannot be really considered real time as it requires mapping and the computation takes time.

Figure 34. Illustration of Problem Navigating through Narrow Path

Chapter 4: Logic Based Obstacle Avoidance

Tendency of Robot missing the opening and enter into the dead corner is high

Figure 35. Illustration of Tendency getting into Dead Corner

4.3.2 Real Time Processing Capability

The VHF* algorithm is able to solve the problems mentioned in the previous section by introducing the look-ahead verification. The only minor flaw here is that it would require the robot to map out the area first before it can navigate through. Though with currently available technology, the mapping time has been greatly reduced, however it will still be difficult to avoid dynamic objects with such a mapping method. The new algorithm that is being proposed here, makes use of the look-ahead concept to solve the first challenge. With regards to the second challenge of being real time reactive algorithm, the way of projecting the final trajectory is totally different here and details will be discussed in a later section.

4.3.3 Integration of Obstacle Avoidance into Formation

In the process of avoiding obstacles, the tendencies of robot collision with fellow followers are very high mainly due to what is mentioned in the second challenge. As shown in Figure 36, when a group of robots are moving in formation, while trying to navigate through a narrow opening, the tendency of collision between followers are high as their trajectory projection would be identical towards each other and it may be too late for both to react.

Chapter 4: Logic Based Obstacle Avoidance

Figure 36. Illustration of Possible Follower Collision

4.4 Overview

From the flow chart below, we can see that the idea is to first base on the range scanner data to analyze the environment and generate only 3 possible paths dynamically. Rationale behind generating only 3 paths is first to reduce computation. The second reason is that in terms of controlling the real hardware, at any point of time, the real resultant displacements of the vehicle can only be either motion towards left, right, front or reverse. In this case, as the sensor covers only the front, reverse option will not be allowed.

Figure 37. Overview of Logical Based Obstacle Avoidance

Chapter 4: Logic Based Obstacle Avoidance

After 3 paths are generated, the next will be path elimination for Leader robot and Follower robots respectively. For the leader robot, in this case we are considering free roaming with no desired movement goal. For the case with a desired goal, the leader robot will choose the path that will lead them closest to the goal in terms of the direct distance. For the Follower robots, they will have to consider the leader robot position and choose only their path that will lead them closer to the leader robot.

4.5 Detailed Explanation

Range Scanner sensor is used for this proposed algorithm. Let the resolution of Range Scanner be R , $RI(i)$ is the index of the resolution where i is between 0 to R , the field of view angle is denoted by FOV and each point of data by $D(i)$, and the range of $D(i)$ is between 0 to 20000mm. Parameter L is being introduced here for setting the look ahead distance.

We first compute all the positive and negative direction zero crossing point indexes,

Figure 38. Illustration of Positive and Negative Cutoff Crossing Point

For deriving positive direction zero crossing point indexes $P_{ZCPI}(y)$,

$$\begin{aligned} \text{Let } Y &= 1, & \text{if } D(i) > D(i-1) \\ Y &= 0, & \text{if } D(i) < D(i-1) \\ Z &= 1, & \text{if } (D(i) - L) \times D(i-1) - L \leq 0 \\ Z &= 0, & \text{if } (D(i) - L) \times D(i-1) - L \geq 0 \end{aligned}$$

$$\rightarrow P_{ZCPI}(y) = RI(i) \times Y \times Z \quad [4-1]$$

Chapter 4: Logic Based Obstacle Avoidance

For deriving positive direction zero crossing point indexes $P_{ZCPI}(y)$,

$$\begin{aligned} \text{Let } Y &= 1, & \text{if } D(i) < D(i-1) \\ Y &= 0, & \text{if } D(i) > D(i-1) \\ Z &= 1, & \text{if } (D(i) - L) \times D(i-1) - L \geq 0 \\ Z &= 0, & \text{if } (D(i) - L) \times D(i-1) - L \leq 0 \end{aligned}$$

$$\rightarrow N_{ZCPI}(y) = RI(i) \times Y \times Z \quad [4-2]$$

After we get the indexes of all the zero crossing points, we will be able to calculate the width of all the peaks. To reduce computation, only the three peaks with the biggest widths will be considered. The value 3 is chosen, as mentioned earlier, since usually there are only 3 ways for the robot to make, that is, left, right or centre.

Figure 39. Illustration of Calculating True World Width

$$\text{Width}_{\max} = \sqrt{(2 \times RI(NZCPI(y)) \times RI(PZCP(y)) - 2 \times RI(NZCP(y)) \times RI(PZCP(y)) \cos(NZCPI(y) - PZCPI(y)) \times \frac{FOV}{R}} \quad [4-3]$$

The above equation is to calculate the biggest real world width of a peak at the cutoff L . Similarly, the second and third biggest widths can be calculated using the above equation.

Chapter 4: Logic Based Obstacle Avoidance

For pure obstacle avoidance, the true real world width will be compared to the robot width, and robot will only choose a path that has width greater than its width. As mentioned earlier we want the new algorithm to consider maintaining a formation as well, thus there will be a need of considering the position of leader and follower robots. This prompted the following.

First we will calculate where the index position of the Leader when mapped onto Range Scanner view is.

Let Leader Robot position be LR_X and LR_Y , Follower Robot position and orientation be FR_X , FR_Y and FR_θ Let index position of leader be LR_{RI}

$$LR_{RI} = \tan^{-1} \left(\frac{LR_X - FR_X}{LR_Y - FR_Y} - FR_\theta \right) \times \frac{R}{FOV} + \frac{R}{2} \quad [4-4]$$

We will now discuss on how to choose a path based on leader position and robot's width. We can easily eliminate those paths whose true widths are smaller than robot's width by simply comparing the calculated width in (3) with the robot width. One criterion is that the Follower robot width cannot be bigger than the Leader robot width. For the remaining path, it will choose a path where it is nearest to the leader. The following details the path selection process.

First step is to eliminate the path with width smaller than robot's width. From the remaining paths, we will find the path that will direct us closest to the leader robot.

$$\text{Index of Path Chosen} = \text{Smallest of } \left\{ \frac{PZCPI(y) + NZCPI(y)}{2} - LR_{RI} \right\}, \text{Width} > \text{Robot Width} \quad [4-5]$$

In the following, we will discuss why the above logic based obstacle avoidance will work for an unknown environment without the need of doing mapping.

Simple Logical Proof:

Chapter 4: Logic Based Obstacle Avoidance

Consider we have the situation below, where only two of the paths allow the robot to pass and the other two paths are marginally smaller than the width of the robot.

Figure 40. Illustration on possible available paths

By applying Equation [4-1], [4-2] and [4-3], paths A and B would be eliminated, as it will be able to calculate path that is smaller than the robot's width.

Next, we illustrate the situation whereby the Follower Robot knows the location of the leader robot and choose the path accordingly.

Figure 41. Illustration of Path Chosen by Follower Robot

By applying Equation [4-4] and [4-5], Follower Robot would be able to rapidly choose the correct path that will lead them closest to the Leader robot.

4.6 Formation with Logic Based Obstacle Avoidance in VCMVSP

Chapter 4: Logic Based Obstacle Avoidance

Figure 42. Testing of Formation with Logical Based Obstacle Avoidance in VCMVSP

Inside the top left screen, we can see that the obstacles are located on left and right side. The result of testing was successful, though the video cannot be shown here, Figure 43 shows the X, Y coordinates plot logged from the Virtual world.

Figure 43. X, Y coordinates plot from logged data in Virtual World

Chapter 5 Implementation of UAV Reconnaissance Mission in VCMVSP

5.1 Overview

In this chapter, further efforts are being put in to allow VCMVSP to have the ability to integrate search algorithm, pattern recognition, obstacle avoidance, and formation algorithms to accomplish a reconnaissance mission in random virtual urban environment created using unreal engine. Each autonomous vehicle would be equipped with these integrated algorithms and thus reconnaissance mission in a random virtual urban environment created using unreal engine can be accomplished collaboratively through distributed processing.

The UPIS Image Server here is responsible for extracting the image from the Unreal Virtual Urban Environment and sending it to any client that requests for the image. The USARSIM Server is responsible for controlling all the virtual robots and sensors in the virtual environment. The data collected from the sensors in the virtual environment will be sent to the clients through the USARSIM Server, after algorithms processing, the resultant control output will be sent through the USARSIM Server back to the Virtual Environment.

Search algorithm here refers to how multiple vehicles can collaborate together to search for a target. Pattern Recognition, as its name implies, uses the vehicle on-board camera to achieve pattern recognition through matrix matching. The obstacle voidance used here is the one mentioned in chapter four. As for formation algorithm, it will only be used in the case where more than one robot searches the same sector and forms a leader-follower relationship. In the case where one robot searches one sector, the search algorithm takes priority over the formation algorithm.

Figure 44. Overview of UAV Reconnaissance Mission

5.2 Search Algorithm

For the implementation scenario here, the entire map of the area desired for surveillance is assumed to be already known. The entire desired surveillance region will be partitioned into cells where each cell is associated with a probability of target existence within the cell. The cell size can be defined base on the size of the desired target. Thus, probability map can be constructed to represent the locations of the desired targets. In order to correctly fill up the probability map, the search algorithm would therefore be required to work hand in hand with pattern recognition algorithm. By making use of the camera facing vertically downward, the pattern recognition would be able to provide the probability map with either '1's or '0's. This is further elaborated in the next section.

The following are quoted from [\[26\]](#) to briefly explain how the probability map was formed. For more details on map updating and path planning, refer to section IV in [\[26\]](#).

Chapter 5: Implementation of UAV Reconnaissance Mission in VCMVSP

The surveillance region O is assumed to be on a plane ground and has been uniformly divided into M cells of the same size. By a slight abuse of notation, each cell is identified with its center $g = [x; y]^T$, where x and y are the coordinates of its center, and “ T ” denotes the transpose operation. All UAVs are assumed to move on a fixed plane above the surveillance region and thus the position of each agent can be described by its projection onto O , which is denoted as $\mu_{i,k} = [x_{i,k}; y_{i,k}]^T$ for agent i ($i = 1, 2, \dots, N$) at time k , where $x_{i,k}$ and $y_{i,k}$ are the planar coordinates of its projection, and N is the number of agents. Each agent is assumed to have access to its own position at any time. Each cell in the surveillance region is associated with a probability or a confidence level of target existence within the cell, modelled as a Bernoulli distribution, i.e. $\theta_g = 1$ (a target is present) with probability $P_{i,k}(\theta_g = 1)$ and $\theta_g = 0$ (no target is present) with probability $1 - P_{i,k}(\theta_g = 1)$ for agent i and cell g at time k . Targets are assumed to be present from the beginning of the search process and remain stationary throughout.

Agent i independently takes measurements $Z_{i,g,k}$ over the cells within its sensing region $C_{i,k}$ at time k with sensing radius R_s (as shown in Figure 44), where $C_{i,k} \triangleq \{g \in O: \|g - \mu_{i,k}\| \leq R_s\}$ and $\|\cdot\|$ denotes the 2-norm for vectors. A cell is assumed to be wholly within $C_{i,k}$ if its center is within $C_{i,k}$. Only two observation results are defined for each $Z_{i,g,k} = 0$ or $Z_{i,g,k} = 1$. For all cells, $P(Z_{i,g,k} = 1 | \theta_g = 1) = p$ and $P(Z_{i,g,k} = 1 | \theta_g = 0) = q$ are constants which are assumed to be known beforehand as the detection probability and false alarm probability respectively.

Figure 45. Search Algorithm

5.3 Pattern Recognition

In order for VCMVSP to have Pattern Recognition incorporated, UPIS needs to be introduced. UPIS is an image server for UT2004. It uses the Microsoft Detours library to intercept the Direct3D calls that UT2004 uses to finalize its screen drawing. It then copies the display to an external buffer, and serves the image upon request to any clients. The FreeImage DLL is being used for decoding of JPEG pictures. The client will response with 'OK' message, each time it gets an image from the image server.

The method for pattern recognition used here is generally matrix matching between target and image captured by a camera. From each given target, we first look at the highest repeating same colour matrix number. When searching, we will identity areas with highest colour same matrix number as target. The procedure is repeated for the next highest colour, till we eliminate others left with only 1 closest target.

Figure 46. Sample illustration on Pattern Recognition

5.3.1 Optimizing Pattern Recognition for Search Algorithm

As mentioned in the previous section for Search Algorithm, the whole surveillance region is partitioned into cells where each cell is associated with a probability of target existence within the cell, which constitutes a probability map for the whole region. In the implementation here, these cells will be labeled with '0', indicating no target found or '1'

Chapter 5: Implementation of UAV Reconnaissance Mission in VCMVSP

indicating a target was found. As a wrongly indicated value would disrupt the entire Search Algorithm, it is important to ensure that the Pattern Recognition has very high accuracy of recognizing the correct target. There are generally two methods of Pattern Recognition being used here, shift-Invariant method and rotation-Invariant method.

The shift-Invariant method does not allow Pattern to be recognized at a tilted angle. It will be able to produce a very high matching score when a camera sees it at a correct angle, thus would be able to set score to very high and greatly reduce chances of false recognition. However when the camera sees the target at a tilted angle, the score produced is rather low. Since there is a high chance that UAV may fly past the vehicle at a tilted angle, the score produced will be lower than a high matching score and thus unable to pick up target.

The rotation-invariant method allows pattern to be recognized at a tilted angle, it thus solves the problem for UAV flying past target at a tilted angle. However the matching score produced is much lower than the shift-invariant method, thus the ability to filter off false detection is not high.

With the above in mind, the adopted strategy to improve accuracy is shown in the flow chart below. Basically it will go through 3 stages of matching, first color matching, second will be rotation invariant and the third will be shift invariant. To reduce computational time, through each stage, only the matching areas will be extracted and passed on to the next stage, thus not all 3 matches are required to run at the time. In fact, at any point of time, only one matching will be running.

Figure 47. Enhanced Target Recognition Procedure

5.4 Integration of All Algorithms

With the basic reference to [2], where a real world robot is created with various algorithms integrated to achieve autonomous reconnaissance mission. Similar methodology flow and approach here would be adopted for the integration of all algorithms in virtual world. There are a total of four algorithms, and as mentioned, the formation algorithm will be redundant in the case where only one UAV surveys a sector. Thus the focus here explains how to integrate the remaining three algorithms for an autonomous reconnaissance mission.

Figure 48 shows the entire process flow chart for integrating the three algorithms. We can see that the three main sensors are Range Scanner, GPS and Camera. Range Scanner provides the distance between agent and obstacle for use in determining when to activate the obstacle avoidance. There are two levels of obstacle avoidance defined here and the main difference between them is simply when the agent is trying to avoid

Chapter 5: Implementation of UAV Reconnaissance Mission in VCMVSP

an obstacle and in the process finds itself to be very near to the obstacle, it will activate level two of obstacle avoidance to navigate through at a very slow speed. The purpose of GPS here is to provide location coordinates of the agent meant to be used in the Search Algorithm. The camera here will keep feeding live images into the pattern recognition algorithm, and target tracking algorithm will take over the search algorithm when the respective agent's target is found.

The Search Algorithm and Pattern Recognition Algorithm simultaneously keep running in the background. As for Obstacle Avoidance Algorithm, as mentioned, it will be triggered when obstacle is nearby and only Search Algorithm will be put on hold while Pattern Recognition continues. When any of the agents has found their target, Search Algorithm will stop and Tracking Algorithm will take over.

Figure 48. Process Flow Chart for Algorithms Integration

5.5 Result of Implementation in Unreal Virtual Urban Environment

In this section, the result of implementation in Unreal Virtual Urban Environment will be described here. On the left of Figure 49, it shows two camera views in grayscale by an on-board downward facing camera of UAVs. In the centre of Figure 49, it shows a Matlab plot, which will plot the position of the two UAVs while it's running. The numbers 1 & 2 on the Matlab plot represent the location of the two targets. Below the Matlab plot, the small picture is the target that both UAVs are trying to search for. On the right of Figure 49, the screen is showing both the UAVs' camera view and also the generic view of the unreal virtual urban environment below the two UAVs' camera view.

The two UAVs are both running the search algorithm to comb their respective area and search for their target. Their location coordinate is assumed to be accurate with the simulation of GPS and inertia sensor. The images retrieved from both UAVs' on board cameras are constantly being sent to the pattern recognition algorithm to search for their desired target. At the same time obstacle avoidance is always in standby mode, ready to be triggered when the obstacle distance hits the trigger level.

In Figure 50, it shows a series of consecutive 3 screen captures while the two UAVs are executing their reconnaissance mission in searching for the target. We can clearly see that both UAVs are moving and the way it moves is determined by the search algorithm. Finally in Figure 51, it shows the continued last series of consecutive 3 screen captures. From the top screen capture picture in Figure 51, we can see that the UAV1 represented by the red color plot in Matlab plot has found its target. The search algorithm for UAV1 will be suspended after it has found its target, and target tracking algorithm will takeover. Meanwhile, UAV2 will continue to search for its target. The bottom picture shows both the UAV1 and UAV2 have found their target.

Chapter 5: Implementation of UAV Reconnaissance Mission in VCMVSP

Figure 49. Screen Capture of Autonomous Vehicle in Reconnaissance Mission – (a)

Chapter 5: Implementation of UAV Reconnaissance Mission in VCMVSP

Figure 50. Screen Capture of Autonomous Vehicle in Reconnaissance Mission – (b)

Chapter 5: Implementation of UAV Reconnaissance Mission in VCMVSP

Figure 51. Screen Capture of Autonomous Vehicle in Reconnaissance Mission – (c)

Chapter 6 Further Enhancing the VCMVSP Capability

6.1 Upgrading Unreal Engine 2.5 to Unreal Engine 3

UT2004 was built with Unreal Engine 2.5, in order to upgrade the 3D simulation platform to Unreal Engine 3, it involves the use of Unreal Development Kit (UDK). By comparing Unreal Engine 2.5 with Unreal Engine 3, the latter inherits all the things that Unreal Engine 2.5 has and on top of that, it supports greater visualization techniques. The Unreal Development Kit is free for non-commercial use Unreal runtime that allows creating games using UE3 without an Unreal Engine 3 license. By making some modifications to USARSIM configuration, USARSIM can be used in conjunction with UDK and UE3 for simulating robots and sensors in 3D virtual environment.

One important reason for upgrading UE2.5 to UE3 is that the UE2.5 support for image server is unstable and will cause the physical engine for UAVs to be affected. For UE3, this problem has been resolved.

Figure 52. Comparing Usage of Google Sketchup for UT2004 and UDK

Chapter 6: Further Enhancing the VCMVSP Capability

The last benefit for the upgrade is the reduction of development time of the virtual environment. 3D Google Sketchup models database, which is also known as Google 3D Warehouse, contains a lot of ready available 3D models. Unfortunately for Unreal Editor, it is only able to accept ASE (ASCII Export File) format, thus a series of conversions is required to use Google Sketchup. In Figure 52, it shows that a total of two additional steps are required to convert Google Sketchup to ASE format for UT2004, compared to direct usage for UDK.

In Figure 53, it shows the side by side comparison between UT2004 and UDK of the 3D rendering effect. We can easily see quite clearly that the UDK 3D rendering effect is much better than UT2004. The amount of 3D rendering details is much more in UDK compared to UT2004. Each object has much more details, and even the swaying of the trees can be included as well. The lighting effects are more evenly distributed for the case of using UDK.

UT2004:

UDK:

Figure 53. Comparing UDK and UT2004 3D Rendering

Chapter 6: Further Enhancing the VCMVSP Capability

The Obstacle Avoidance Algorithm is used to do quick basic functionality check on whether there will be any abnormality in replacing UT2004 with UDK. In Figure 54, it shows the X-Y plot of three autonomous vehicles moving in a formation manner, navigating through various obstacles.

Figure 54. Quick Test of Obstacle Avoidance in UDK

6.2 Simulating Wireless Transmission

The purpose of introducing wireless transmission simulation is to make the entire platform more complete, similarly to the real world. As in the real world, whenever there is wireless transmission, there will be inevitably transmission loss and delay. There are two ways here that the wireless transmission can be simulated or imitated. The first method is to run more than one computer with each controlling one vehicle and sending data wirelessly through wifi to each other for processing. This first method here has one problem that it will not be able to handle scalability easily, that is, if a lot of vehicles are required, it is difficult to gather and setup many computers and wifis.

The second method here is to make use of wireless transmission simulation software. The main work involved here would be on choosing appropriate software and also how

Chapter 6: Further Enhancing the VCMVSP Capability

to interface the selected software with LabView. By using simulation software, we would be able to easily solve the scalability problem faced by the first method. There are two wireless simulation software packages being considered here, OMNet++ and WSS.

6.2.1 Introduction to OMNet++

OMNet++ [27] is introduced here as it allow us to simulate the wireless transmission up to a great extend. For example, we can choose whether we wants to simulate 802.11a, 802.11b or 802.11g. Within these standards, there are different bit rate settings that OMNet++ allows us to simulate as well. More advance speaking, signal strength in presence of obstacle, physical layer parameters can be simulated as well.

6.2.2 Integration of OMNet++

The idea here is to first retrieve the coordinates of individual vehicles, each vehicle coordinate represents another vehicle in OMNet++. All vehicles will not know their position directly from Unreal, instead it is sent to each other through OMNet++. Finally, OMNet++ will send all the values back to LabView in order to calculate the next movement for the followers after the leader has moved. The communication between OMNet++ and LabView is done through localhost TCP/IP communication. After reading all the data from OMNet++, LabView will need to pass these data into its embedded Matlab script for algorithm processing. Figure 55 shows how OMNet++ and LabView can be integrated in order to simulate the wireless transmission of data with scalability.

Figure 55. Integration of OMNet++ and LabView

6.2.3 Wireless Simulation Server

Though OMNet++ is considered a very powerful and good wireless simulation tool, however due to the complexity, it is difficult to be integrated into the current platform. An easier alternative will be the use of Wireless Simulation Server (WSS) [28]. The key feature of the WSS is that it allows wireless network links to be simulated with a specified signal degradation model.

In Figure 56, it illustrates how two UAVs will communicate with each other after the introduction of WSS. The total number of ports required would be four, where the WSS will do the internal re-routing. Subsequently, every introduction of one UAV, the additional port required would be two and WSS will continue to do the internal routing.

Figure 56. Communication between two UAVs with the use of WSS

The signal attenuation model being used for the wireless simulation server is as follows,

$$S = P_{d0} - 10 * N * \log_{10}(d_m/d_0)$$

Where S is the calculated signal strength

P_{d0} is the path loss in dBm at d_0 meters from the source.

d_0 is the distance from the source

N is the signal attenuation factor of the environment over distance.

Chapter 6: Further Enhancing the VCMVSP Capability

As we can find out easily the receiver sensitivity of the wireless radio from its datasheet, the signal cutoff can be set to same as desired setting of the wireless radio intended to use.

Chapter 7 Conclusion and Possible Future Works

7.1 Conclusion

Three main contributions have been reported; the first is the development of the VCMVSP and the second is the proposed logic based obstacle avoidance. The second contribution here is the key to third contribution, in the sense that logic based obstacle avoidance allows much easier integration with other algorithms. Third and final contribution is further enhancing VCMVSP, to allow integration of various algorithms.

The VCMVSP can be considered a much better simulation platform for any testing that involves robots and sensors compared to existing simulators that do not provide any visualization and do not provide any easy way in creating any environmental constraints. From the implementation of obstacle avoidance and formation algorithm in chapter 3, it has successfully been demonstrated that the exact same algorithm works in both the real world and virtual environment. Thus, the fundamental function of VCMVSP can be considered successfully developed.

In chapter 4, the formation with the logic based obstacle avoidance gives the flexibility for the leader robot to call for a change of formation as the look-ahead processing gives it the capability to change according to the shape of an obstacle. Result of testing in the Virtual world was successful. Real world testing of the new algorithm of the obstacle portion was successfully conducted which addressed most of the challenges mentioned in chapter 4.. It can thus be concluded that the logic base obstacle avoidance works well in the real world as well as in the virtual world.

With the successful simulation of UAV Reconnaissance Mission with VCMVSP, which is shown in chapter 5, it can be concluded that advanced VCMVSP has been developed successfully. This UAV Reconnaissance Mission relies on the simplicity in integrating the logic based obstacle avoidance. This shows that the logic based obstacle avoidance

Chapter 7: Conclusion and Possible Future Works

is not just suited for formation, but also suitable for integrating with other algorithms as well. Through a systematic flow approach, it also helps to solve the problem of kinematic contention of various algorithms.

7.2 Possible Future Works

Though in the overall architecture of VCMVSP it was mentioned that wireless simulation can be implemented using OMNet++ simulation, there is no test bed done here that makes use of this feature. Thus one of the future works here would be to put more focus on the wireless simulation. To further enhance the capability of VCMVSP, multiple robot views can be included so as to provide a control station view of all status of robots.

One of the problems not ideally resolved by the formation with logic based obstacle avoidance is the need of a more robust way of deciding when to trigger change of formation when obstacle is cleared. The current way is to immediately trigger change of formation when obstacle is detected and when obstacle clearance is detected. This immediate trigger method will cause the followers to hover between obstacle avoidance and maintaining formation as followers have not completely passed through the narrow passage yet. Though it would still be able to keep track of the leader and pass through the passage successfully, the trajectory will not be an optimized one.

The issue of the change of communication topology can be another further research problem. In the case of a long chain leader-follower formation, leader formation would take some time to reach the last follower. This may cause formation to break up easily when connections become poor while trying to avoid obstacles. Methods like breaking up the long chain into series of leader-follower can be adopted to prevent breaking up of formation. More attention can be focused on improving the accuracy of the signal attenuation model, for example, by considering the number of obstacles that a signal has passed through and the materials of the obstacles.

Author's Publications and Works

- [1] J. Xu, L. H. Xie, T. G. Toh and Y. K. Toh, "SNLSim: A Hybrid 3D Simulator for Networked Multiple Vehicle Systems", Proc. The 31st Chinese Control Conference, Hefei, China, 25-27 July 2012.
- [2] Y. K. Toh, T. G. Toh, E. S. Tan, Y. Z. Tan and P. L. Dai, "Creating an Autonomous Robot for a National Competition Using LabView and NI Vision", NI ASEAN Paper Contest 2009, Best Innovation in Robotics Award, October 2009

References

- [3] E. A. Macdonald, "Multi-Robot Assignment and Formation Control", Unpublished Master Thesis, Georgia Institute of Technology, Page(s) 46 – 56, August 2011
- [4] Z. Wilson, T. Whipple and P. Dasgupta, "Multi-robot Area Coverage using Dynamic Coverage Information Compression," 8th International Conference on Informatics in Control, Automation and Robotics (ICINCO), pp. 236-241, 2011
- [5] J. Wang and S. Balakirsky, USARSim Manual Version 3.1.3, "A Game Based Simulation of Mobile Robots", NIST
- [6] OpenSim, "OpenSimulator", Website http://opensimulator.org/wiki/Main_Page
- [7] V-REP, "Virtual Robot Experimentation Platform", Website <http://www.v-rep.eu/>
- [8] MORSE, "the Modular OpenRobots Simulation Engine", Website <http://www.openrobots.org/wiki/morse/>
- [9] Gazebo, Website <http://gazebo.org/>
- [10] J. Shao, G. Xie and L. Wang, "Leader-following formation control of multiple mobile vehicles", IET Control Theory and Applications, 1(2), 545-552, 2007
- [11] X. Chen, A. Serrani, and H. Ozboy, 'Control of leader-follower formations of terrestrial UAVs'. Proc. IEEE Conf. on Decision and Control, Maui, Hawaii, USA, pp. 498–503, Dec 2003.
- [12] T. Balch and R. C. Arkin, 'Behavior-based formation control for multirobot team', IEEE Trans. Robot. Autom., 14, (6), pp. 926–939, 1998
- [13] P. Ögren, M. Egerstedt., and X. Hu, 'A control Lyapunov function approach to multi-agent coordination', IEEE Trans. Robot. Autom., 18, (5), pp. 847–851, 2002.
- [14] J. Fredslund, and M. J. Mataric, 'A general algorithm for robot formations using local sensing and minimal communication', IEEE Trans. Robot. Autom., 18, (5), pp. 837–846, 2002
- [15] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor, 'A vision based formation control framework', IEEE Trans. Robot. Autom., 18, (5), pp. 813–825, 2002

Bibliography

- [16]N. Sorensen and W. Ren “A unified formation control scheme with a single or multiple leaders”, Proceedings of the 2007 American control conference July 11-13 2007
- [17]R. Siegwart and I. R. Nourbakhsh, Mobile Robotics Textbook, “Introduction to Autonomous Mobile Robots”, The MIT Press, 2004
- [18]T. Bresciani, “Modelling, Identification and Control of a Quadrotor Helicopter”, Unpublished Master Thesis, Department of Automatic Control Lund University, Page(s) 119 -123, October 2008
- [19]M. Becker, C. M. Dantas and W. P. Macedo, “Obstacle Avoidance Procedure for Mobile Robots”, ABCM Symposium Series in Mechatronics - Vol. 2 - pp.250-257, 2006
- [20]H. P. Movarec and A. Elfes, “High Resolution Maps from Wide Angle Sonar”, IEEE Conference on Robotics and Automation, Washington D.C., pp. 116-121, 1985
- [21]J. Borenstein and Y. Koren, “The vector field histogram – fast obstacle avoidance for mobile robots”, IEEE Journal of Robotics and Automation, vol. 7, n. 3, pp. 278-288, 1991.
- [22]I. Ulrich and J. Borenstein, “VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots”, Proceedings of the 1998 IEEE International Conference on Robotics & Automation, Leuven, Belgium, pp. 1572-1577, May 1998
- [23]I. Ulrich, and J. Borenstein, “VFH*: Local Obstacle Avoidance with look-ahead Verification”, Proceedings of the 2000 IEEE International Conference on Robotics & Automation, San Francisco, CA, April 2000.
- [24]D. Fox, W. Burgard and S. Thrun, “The Dynamic Window Approach to Collision Avoidance”, IEEE Robotics & Automation Magazine, pp. 23-33, March 1997
- [25]O. Brock and O. Khatib, “High speed navigation using the global dynamic window approach”, Proceedings of the 1999 IEEE International Conference on Robotics & Automation, Detroit, MI, pp. 341-346, 1999
- [26]J. W. Hu, L. H. Xie, K. Y. Lum and J. Xu, “Multi-agent Information Fusion and Cooperative Control in Target Search”, IEEE Transactions on Control Systems Technology, Jan 2012

Bibliography

[27] OMNet++, Website: <http://www.omnetpp.org/>

[28] WSS, “Wireless Simulation Server”, Website:

<http://robotics.jacobs-university.de/VirtualRobots/WSS.html>

[29] T. Fan. Development of a hybrid ugv and uav cooperative system. FYP report A4160-101, NTU, 2010.