CRANFIELD UNIVERSITY

Hanlin Niu

Energy Efficient Path Planning and Model Checking for Long Endurance Unmanned Surface Vehicles

School of Aerospace, Transport and Manufacturing
Aeronautical Engineering

Doctor of Philosophy Degree

Supervisor: Dr Al Savvaris & Professor Antonios Tsourdos
September 2017

CRANFIELD UNIVERSITY

School of Aerospace, Transport and Manufacturing
Aeronautical Engineering

PhD thesis

Hanlin Niu

Energy Efficient Path Planning and Model Checking for Long Endurance Unmanned Surface Vehicles

Supervisor: Dr Al Savvaris & Professor Antonios Tsourdos
September 2017

This thesis is submitted in partial fulfilment of the requirements for the degree of PhD

ABSTRACT

In this dissertation, path following, path planning, collision avoidance and model checking algorithms were developed and simulated for improving the level of autonomy for Unmanned Surface Vehicle (USV). Firstly, four path following algorithms, namely, Carrot Chasing, Nonlinear Guidance Law, Pure pursuit and LOS, and Vector Field algorithms, were compared in simulation and Carrot Chasing was tested in Unmanned Safety Marine Operations Over The Horizon (USMOOTH) project. Secondly, three path planning algorithms, including Voronoi-Visibility shortest path planning, Voronoi-Visibility energy efficient path planning and Genetic Algorithm based energy efficient path planning algorithms, are presented. Voronoi-Visibility shortest path planning algorithm was proposed by integrating Voronoi diagram, Dijkstra’s algorithm and Visibility graph. The path quality and computational efficiency were demonstrated through comparing with Voronoi algorithms. Moreover, the proposed algorithm ensured USV safety by keeping the USV at a configurable clearance distance from the coastlines. Voronoi-Visibility energy efficient path planning algorithm was proposed by taking sea current data into account. To address the problem of time-varying sea current, Genetic Algorithm was integrated with Voronoi-Visibility energy efficient path planning algorithm. The energy efficiency of Voronoi-Visibility and Genetic Algorithm based algorithms were demonstrated in simulated missions. Moreover, collision avoidance algorithm was proposed and validated in single and multiple intruders scenarios. Finally, the feasibility of using model checking for USV decision-making systems verification was demonstrated in three USV mission scenarios. In the final scenario, a multi-agent system, including two USVs, an Unmanned Aerial Vehicle (UAV), a Ground Control Station (GCS) and a wireless mesh network, were modelled using Kripke modelling algorithm. The modelled uncertainties include communication loss, collision risk, fault event and energy states. Three desirable properties, including safety, maximum endurance, and fault tolerance, were expressed using Computational Tree Logic (CTL), which were verified using Model Checker for Multi-Agent System (MCMAS). The verification results were used to retrospect and improve the design of the decision-making system.

Keywords:

Collision Avoidance; Path Following; Path Planning; Energy Efficient; Spatially-Temporally Variant Sea Current; Model Checking; Decision Making;

ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr. Al Savvaris and Professor Antonios Tsourdos for giving me a great opportunity to undertake my Ph.D. research. In addition, throughout my Ph.D., their constant inspiring and practical guidance for the research has contributed significantly to a successful completion of my research. Special thanks to the team members of ASV Ltd, who give me the chance to work on the industrial projects (LEMUSV & USMOOTH) in Portsmouth.

I am thankful to Dr. Hyo-Sang Shin, Dr. Chang-Hun Lee, Dr. Aldo Vargas and Dr. Jiyoung Choi for the guidance regarding my research and help in developing creative solutions, which has not always been easy. I would like to also thank internal/external examiners Professor Rafal Zbikowski and Dr Wasif Naeem for taking time out of their busy schedule to review my thesis and providing valuable suggestions to improve the quality of my work.

I am grateful to my colleagues and friends Shuoyuan, Shaoming, Hao, Fan, Ye, Bilal, Siqi, Runqi, Xin, Si, Dr. Wang, Costas, Vasilis and Spiros with whom I enjoyed the time at Cranfield. I have gone through all of my ups and downs in my physical and spiritual life with them in the UK. Special thanks to Shirley for her company!

I also would like to express my gratitude to my family. Without their support, I could not make it this far. Although they are a thousand miles away, they have always made me feel their presence during these years.

Lastly, thanks to all the Chinese, Greek and Korean friends in Cranfield!

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AFM	Anisotropic Fast Marching
AIS	Automatic Identification System
CAD	Closest Approach Distance
CC	Carrot Chasing
CTL	Computational Tree Logic
COLREGS	International Regulations for Preventing Collision at Sea
CPA	Closest Point Approach
CTMCs	Continuous-Time Markov Chains
CVM	Curvature Velocity Methods
DTMC	Discrete-Time Markov Chain
DP	Dynamic Programming
DW	Dynamic Window
EA	Evolutionary Algorithm
FSA	Finite State Automata
GA	Genetic Algorithm
GCS	Ground Control Station
GNC	Guidance, Navigation and Control
GSHHS	Global Self-Consistent Hierarchical High-Resolution Shorelines
LEMUSV	Long Endurance Marine Unmanned Surface Vehicle

LTL	Linear Temporal Logic
LQR	Linear Quadratic Regulator
IMO	International Maritime Organization
ISPL	Interpreted Systems Programming Language
MAS	Multi-Agent Systems
MCMAS	Model Checker for Multi-Agent Systems
MDPs	Markov Decision Processes
MILP	Mixed Integer Linear Programming
MPC	Model Predictive Control
MSC	Message Sequence Chart
NLGL	Nonlinear Guidance Law
SPIN	Simple Promela Interpreter
PA	Probabilistic Automata
PLOS	Pure Pursuit and Line-of-Sight
PN	Proportional Navigation
PN	Petri Net
PRISM	Probabilistic Model Checker
PROMELA	Process Meta Language
PTAs	Probabilistic Timed Automata

TCPA	Time to Closest Point of Approach
TCTL	Timed Computation Tree Logic
UAV	Unmanned Aerial Vehicle
USMOOTH	Unmanned Safe Maritime Operations Over The Horizon
USV	Unmanned Surface Vehicle
UUV	Unmanned Underwater Vehicle
VF	Vector Field
VFH	Vector Field Histogram
VM	Voronoi Path Planning Algorithm Refined by Minimizing the Number of Waypoints
VTP	Virtual Target Point
VV	Voronoi-Visibility

1 Introduction

1.1 Motivation

Unmanned Surface Vehicles (USVs) have attracted a huge amount of interest in recent years because of their ease of use and low operational costs. USVs are ideal for deployment in missions that are tedious or dangerous for humans. USVs have been implemented in both the civil markets and the military. Examples of those developed platforms include Singapore-based Zycraft Independent Unmanned Surface Vehicle (IUSV) (Defense Media Network, 2013), Protector USV (Naval Technology, 2013), Israeli project SeaStar, British Thales Halcycon USV, Swedish Piraya USV and British C-Cat (ASV Global Ltd, 2016). More examples are shown in (Liu, et al., 2016), (Manley, J.E, 2008), (Motwani, A, 2012) and Appendix. Recently, the renewable energy technology is under development for long endurance missions, as the energy harvesting technology from the nature will improve the endurance capability of the USV. Liquid robotics has developed a USV harnessing energy from wave and solar panel. Wave glider USV has demonstrated extended deployments of 142 days during which over 2500 nautical miles were travelled. ASV global Ltd developed the C-Enduro USV that harness solar energy, wind energy, and diesel energy. This renewable energy technology allows the C-Enduro USV to travel 3 months. The C-Enduro USV is shown in Fig. 1. 1.

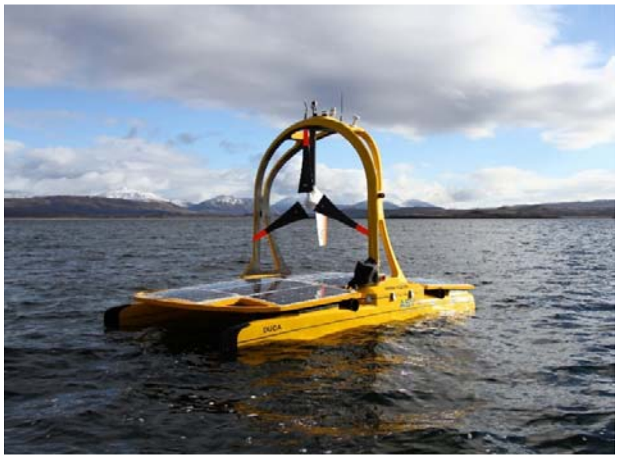


Fig. 1. 1. The C-Enduro USV (ASV Global Ltd, 2013)

As the brain of the long endurance USV, Guidance, Navigation and Control (GNC) system determines the level of autonomy. The GNC system, including path following, path planning and collision avoidance algorithms, plays an important role in determining the safety level of the USV. In the industrial project Safe Maritime Operations Over The Horizon (USMOOTH) (RESEARCH COUNCILS UK, 2015), The path following algorithms should be compared based on their accuracy and the required control effort in the presence of the wind and sea current disturbances, as the result can be used for determining the proper path following algorithm for the specific application. By taking account of the spatial and temporal variance of the sea current, the energy efficient path planning algorithm can contribute to the long endurance capability of the USV. The collision avoidance algorithm can be implemented for keeping the safety of the USV in the presence of multiple dynamic and static intruders and also obey the International Regulations for Preventing Collisions at Sea (COLREGS), which is required in the industrial project Long Endurance Marine Unmanned Surface Vehicle (LEMUSV) (ASV Global Ltd, 2013). Besides the path following, path planning and collision avoidance algorithms, the verification of the decision-making system is also necessary for ensuring the safety property of the long endurance USV in dealing with multiple environmental uncertainties.

1.2 Background

Path Following Algorithm

Most USV missions are following straight line or circular path using path following algorithms, which act on the vehicle orientation to drive it to the path. A basic requirement for the path following algorithms is that they must be accurate in the presence of the wind and the sea current disturbances. The accuracy can be calculated by using the cross-track error, which is the offset of the vehicle from the desired path. Another criterion to evaluate the performance of the path following algorithm is the control effort. The control effort quantifies the control demands of the algorithm. If the solution takes too many turns, then the control effort will be high, which is not desirable (Sujit, et al., 2014). The path following algorithms can be classified into geometric methods and control methods. The geometric algorithms include Carrot Chasing (CC), Pure Pursuit (Conte, et al., 2004), Line-of-Sight (LOS) (Fossen, 1994), the combination of Pure Pursuit and LOS (PLOS) (Kothari, et al., 2010), Nonlinear Guidance Law (NLGL) (Park, et al., 2007), Vector Field (VF), etc. The control methods includes Linear Quadratic Regulator (LQR), Sliding Model Control (Ashrafiuon, et al., 2008) (Liao, et al., 2011), Model Predictive Control (MPC) (Naeem, et al., 2006), backstepping control (Bibuli, et al., 2007), gain scheduling theory, adaptive control and dynamic programming.

As required by the industrial project Safe Maritime Operations Over The Horizon (USMOOTH), it is necessary to compare the accuracy and the required control effort of the existing geometric path following algorithms because of their simplicity, robustness and ease of implementation. The algorithms should be implemented with the USV dynamic model and the control system, while taking account of the wind and sea current disturbances.

Path Planning Algorithm

Path planning represents an important characteristic of autonomous systems. It represents the possibility for Unmanned Surface Vehicle (USV) and Unmanned Underwater Vehicle (UUV) deployment during a mission in presence of different disturbances and uncertainties, which is a complicated procedure due to existence of the unknown, inaccurate and varying information. Generating a feasible shortest path and taking account of the map inaccuracy and uncertainties and also keeping the computational efficiency is a challenging task. As the map data may be inaccurate, it is necessary to keep a clearance distance from the coastlines. While processing the large spatial dataset, keeping the computational efficiency of the algorithm is also necessary. The Voronoi diagram can be constructed in $O(n)$ time and generate a path with maximum clearance distance, where n is the number of the islands' vertices. However, the Voronoi path maybe far from optimal and it usually has redundant turns. The Visibility graph can generate better path than the Voronoi diagram (Kaluđer, et al., 2011), however, it will consume $O(n^2)$ time to construct the collision-free roadmap, which makes it impractical for large spatial dataset (Bhattacharya & Gavrilova, 2008). In terms of saving energy, the energy efficient path can be generated by taking account of the sea current state. However, when searching the energy efficient path under time varying sea current state, the deterministic and heuristic methods like A* suffer from expensive computational cost and criticized for their weak performance in high-dimensional problems (MahmoudZadeh, et al., 2016). The dynamic programming (Bryson & Ho, 1975) and Mixed Integer Linear Programming (MILP) (Yilmaz, et al., 2008) also have the same problem when dealing with the high-dimensional dataset. Therefore, there is also a strong need to develop the energy efficient path planning algorithm while keeping the computational efficiency when processing the high dimensional spatial dataset.

Collision Avoidance Algorithm

The collision avoidance algorithm, as an important part of the GNC system, helps to keep the safety of the USV in the dynamic and cluttered environment. In maritime navigation, vessels should obey the International Regulations for Preventing Collision at Sea (COLREGS) (Lloyd’s Register Rulefinder, 2005), agreed to by the International Maritime Organization (IMO) in 1972 (Kuwata, et al., 2011). According to the COLREGS, the overtaking, head-on and crossing situations are defined to describe the situations. These ‘rules of the road’ specify the type of maneuvers that should be taken in certain situations where there is a collision risk, its collision avoidance algorithm must abide by COLREGS, so that the USVs can safely avoid other vessels and the drivers of the other vessels can expect certain safe behaviours from USVs. Several collision avoidance algorithm obeying COLREGS have been proposed in the past, such as fuzzy logic (Lee, et al., 2004) (Perera, et al., 2009), Evolutionary Algorithm (EA) (Colito, 2007), neural network, hybrid of these algorithms (Statheros, et al., 2008), interval programming, and 2D grid map (Teo, et al., 2009). However, they do not scale well with multiple traffic boats (Kuwata, et al., 2011).

As required by the industrial project Long Endurance Marine Unmanned Surface Vehicle (LEMUSV), it is necessary to develop the collision avoidance algorithm to keep the safety of the USV in the existence of single or multiple intruders while obeying the COLREGS.

Model Checking for Decision-Making System

Formal verification can be an extremely advantageous alternative technique comparing with simulation, as it is not just complete in logic and rigorous in mathematics but flexible for the modelling and specification of complex behaviours. As one of the formal verification methods, deductive verification is proof-based and has significantly influenced the software development. However, deductive verification is time-consuming and can be performed only by experts in logic and mathematics (Sirigineedi, et al., 2009). Model checking is a method that can be used to verify a given system through modelling the system, specifying the property and verifying the property using model checkers (Clarke, 1997). The counterexamples of the model checkers can be used to retrospect and improve the design of the system. The correctness of decision-making system for long endurance USVs is fundamentally critical, as in the long-range mission, the USV may encounter multiple concurrent uncertainties or problems in relation with the communication loss, faults (hardware or software malfunctions), the energy generation and energy consumption. When multiple autonomous agents are deployed simultaneously, the verification of the complex decision-making systems becomes more important.

Therefore, it is also necessary to use model checking algorithm to verify the property of the USV decision-making system.

1.3 Aim of Research

In the industrial project USMOOTH (Unmanned Safe Maritime Operations Over The Horizon), which is funded by the Innovative UK, the performances of four

path following algorithms, including Carrot Chasing, Nonlinear Guidance Law, Pure pursuit and LOS, and Vector Field algorithm, should be compared using the dynamic model of the C-Enduro USV (see Chapter 2).

This research focuses on the development of **path planning algorithms, collision avoidance algorithm** and **model checking for decision-making system**, as listed:

- **The shortest feasible path planning algorithm** generates the shortest path between the starting point and the destination. The proposed algorithm should maintain the safety of USVs due to the uncertainties or inaccuracies in the map data. The algorithm should also be computationally efficient in dealing with large spatial dataset (see Chapter 3).
- **The energy efficient path planning algorithms** generates the energy efficient path by taking account of the spatially variant and spatially-temporally variant sea current data. The effects of the sea current state and the USV speed on the path planning results should also be analysed (see Chapter 4 and Chapter 5).
- **The collision avoidance algorithm** is designed for supporting the industrial project LEMUSV (Long Endurance Marine Unmanned Surface Vehicle), which is funded by the UK government-backed Small Business Research Initiative. The proposed algorithm should navigate the USV to avoid the potential collision risk, keep a safety clearance distance with the intruders and comply with the COLREGS (International Regulations for Preventing Collisions at Sea) (sea Chapter 6).
- **Model checking for decision-making system:** Motivated by the demand for certification for USV systems, this thesis aims to deal with three practical scenarios and use the model checking algorithm to verify the decision-making system of USVs under multiple non-deterministic and concurrent factors including communication states, malfunction, traffic information, energy consumption, energy generation and wireless mesh network (see Chapter 7).

1.4 Contributions to Knowledge

In this research, four path following algorithms are reviewed and compared using the dynamic model and control system of C-Enduro USV (see Chapter 2). The results are given in Publication [2], which is listed in Section 1.5. This research addresses the problem of developing the USV path planning algorithms, collision avoidance algorithm and the implementation of model checking method for the verification of the USV decision-making system.

Firstly, the thesis proposes a Voronoi-Visibility roadmap based shortest path planning algorithm (see Chapter 3). The Voronoi diagram is well known for its computational efficiency, however, the generated path is far from optimal. The Visibility graph can generate optimal path but the computational complexity is impractical. The proposed algorithm integrates the advantages of the Voronoi diagram and the Visibility graph. In ten USV missions in the strait of Singapore, the proposed algorithm was compared with the VM algorithm (Voronoi path planning algorithm refined by Minimizing the number of waypoints) in terms of the computational time and path length. It was demonstrated that the proposed algorithm improves the quality of the Voronoi shortest path and VM shortest path and also keeps the same level of computational efficiency as that of the Voronoi diagram approach. The proposed algorithm was also tested in five missions in the islands of the coast of Croatia, which demonstrated the feasibility of the proposed algorithm in different geographical scenario. Moreover, the proposed algorithm ensures that the USV remains at a user-configurable safety distance away from all islands and coastlines, hence maintaining the safety of USVs due to the uncertainties or inaccuracies in the map data.

Secondly, the proposed Voronoi-Visibility roadmap based shortest path planning algorithm is extended by taking account of the spatially variant sea current data and a Voronoi-Visibility roadmap based energy efficient path planning algorithm is proposed (see Chapter 4). In the proposed approach, Voronoi diagram, Visibility graph, Dijkstra's search and energy consumption function were combined, which allows USVs to avoid obstacles while at the same time using minimum amount of energy. The Voronoi-Visibility energy-efficient path and the corresponding shortest path were simulated and compared for ten missions in Singapore Strait and five missions in Croatian islands. The simulation results also provided an insight into the amount of energy that could potentially be saved by taking the sea current into account when planning the USV path and not only relying on the shortest path information. The effects of the mission time, the USV speed and the sea current state on the results were then analysed. The computational time of the Voronoi-Visibility (VV) algorithm and the Voronoi energy efficient path planning algorithm was also compared, it is demonstrated that the proposed VV algorithm improves the quality of the Voronoi energy efficient path but not reduce the computational efficiency of the Voronoi energy efficient path planning algorithm. The results are given in Publications [3] and [6].

Thirdly, the proposed Voronoi-Visibility energy efficient path planning algorithm was extended by being integrated with Genetic Algorithm (GA) to search for the energy efficient path in the time-varying sea current (see Chapter 5). To test the performance of the proposed algorithm, the geographical data of Singapore islands and the historical sea current data were applied. The energy efficiency of the proposed algorithm was validated by comparing with the Voronoi-Visibility energy efficient path planning algorithm in three USV missions with time varying sea current state.

Fourthly, a geometric algorithm-based collision avoidance algorithm was developed (see Chapter 6). The proposed algorithm integrated collision detection, decision-making, collision resolution and resolution guidance algorithms. The decision-making algorithm keeps USV comply with COLREGS. The holistic algorithm was finally validated in single intruder, multiple dynamic intruders and multiple static-dynamic mixed intruders scenarios. The results showed that USV can avoid the static and dynamic intruders and keep the safe distance with the intruders. The details are given in Publications [1] and [5].

Lastly, the feasibility of implementing model checking algorithm for verifying the decision-making logic of USV under concurrent uncertainties in the design-level was carried out in this research (see Chapter 7). Three USV mission scenarios were implemented to model and verify the USV decision-making system. In the first scenario, the non-deterministic and concurrent environmental conditions including communication loss, collision risk and fault event were modelled to verify the safety property of the decision-making system. The second scenario extended the first scenario by taking account of the energy states. The USV considered in the second scenario was proposed by referring to the long endurance C-Enduro USV, which has solar panel, wind turbine and diesel generator as energy resources. The environmental conditions those affect the energy generation condition include the solar irradiance, wind condition and diesel generator state. The environmental condition affecting the energy consumption include the sea current state. The interactive relations between these factors have been modelled to test the safety property and long endurance property of the USV decision-making system in the second scenario. In the third scenario, a group of autonomous vehicles, including a long endurance USV, a regular USV, an UAV and a Ground Control Station (GCS), and a mesh network were taken into account to improve the mission completion. Finally, the safety property, the long endurance property and the fault tolerance property of the system were verified. In these three scenarios, the USV decision-making system, the UAV decision-making system, the GCS decision-making system, the uncertainties of the environmental conditions, the wireless mesh network and the states of the corresponding sensors were modelled using Kripke model (Clarke, et al., 1999). The properties to be verified were expressed using Computational Tree Logic (CTL) and finally the properties were verified using model checker MCMAS. The verification results helped retrospect and improve the design of the system by considering additional environmental uncertainties, additional agents and behaviours during three steps of scenario modification. This work also represents the scalability and computational efficiency of the MCMAS which can verify the extended multi-agent system consisting of thirty-five Kripke models, including fifteen communication-related models, four collision risk-related models, seven fault event-related models, five energy-related models, and four autonomous systems-related models.

1.5 Organisation of the Thesis

The remainder of this thesis is organised as follows: Chapter 2 reviews four path following algorithms, including Carrot Chasing, Nonlinear Guidance Law, Pure Pursuit and Line-of-Sight, and Vector Field algorithms. These four path following algorithms have been integrated with the control system and dynamic model of C-Enduro USV. The accuracy and control effort were compared and the results are given in Publication [2]. Chapter 3 presents the development of the Voronoi-Visibility shortest path planning algorithm. The proposed algorithm was compared with the Voronoi-Visibility shortest path planning algorithm and VM shortest path planning algorithm in ten Singapore strait missions and five Croatian islands missions. Chapter 4 proposes the development of the Voronoi-Visibility energy efficient path planning algorithm in spatially variant sea current and the comparison between the Voronoi-Visibility energy efficient path and the Voronoi-Visibility shortest path is also given. Both the computational efficiency and energy efficiency of the proposed algorithm were demonstrated in ten Singapore strait missions. The results are given in Publications [3] and [6]. Chapter 5 presents the Voronoi roadmap and Genetic Algorithm (GA) based energy efficient path planning algorithm in spatially-temporally variant sea current. The GA based algorithm has been compared with the energy efficient path planning algorithm presented in Chapter 4, showing the energy efficiency in dealing with time-varying sea current state. Chapter 6 presents the development and implementation of the collision avoidance algorithm. The results of avoiding single and multiple intruders are given in Publications [1] and [5], respectively. In Chapter 7, the feasibility of using model checking for verifying the decision-making logic of USV under multiple concurrent uncertainties in the design-level is demonstrated. The decision-making algorithms of three USV scenarios were modelled and verified, respectively. In the third scenario, a multi-agent system, including a long endurance USV, a regular USV, an UAV, a GCS and a wireless mesh network, are modelled and verified successfully. Finally, the conclusions and future work are presented in Chapter 8.

1.6 Publications

[1]. Al Savvaris, Hanlin Niu, Hyondong Oh and Antonios Tsourdos. ***Development of Collision Avoidance Algorithms for the C-Enduro USV***, 2014, The international Federation of Automatic Control Cape Town, South Africa. August 24-29, 2014, Proceedings of the 19th World Congress, pp. 12174-12181.

[2]. Hanlin Niu, Yu Lu, Al Savvaris and Antonios Tsourdos. ***Efficient Path Following Algorithm for Unmanned Surface Vehicle***, In OCEANS 2016, Shanghai, China, 10-13 April 2016, IEEE, pp. 1-7.

[3]. Hanlin Niu, Yu Lu, Al Savvaris and Antonios Tsourdos. ***Efficient Path Planning Algorithm for Unmanned Surface Vehicle***, 2016, 10th IFAC Conference on Control Applications in Marine Systems CAMS 2016: Trondheim, Norway, 13-16 September 2016, Elsevier, IFAC-PapersOnLine 49, no. 23, pp. 121-126.

[4]. Yu Lu, Hanlin Niu, Al Savvaris and Antonios Tsourdos. ***Verifying Collision Avoidance Behaviours for Unmanned Surface Vehicles using Probabilistic Model Checking,*** 2016, 10th IFAC Conference on Control Applications in Marine Systems CAMS 2016: Trondheim, Norway, 13-16 September 2016, Elsevier, IFAC-PapersOnLine 49, no. 23, pp. 127-132.

[5]. Hanlin Niu, Al Savvaris and Antonios Tsourdos. ***USV Geometric Collision Avoidance Algorithm for Multiple Marine Vehicles*** , In OCEANS 2017, Anchorage, USA, 18-21 September 2017.

[6]. Hanlin Niu, Yu Lu, Al Savvaris and Antonios Tsourdos. ***An Energy Efficient Path Planning Algorithm for Unmanned Surface Vehicles*** (Accepted by Ocean Engineering)

[7]. Hanlin Niu, Al Savvaris and Antonios Tsourdos. ***Energy Efficient Path Planning Algorithm in Spatially-Temporally Variant Environment*** (Under internal review)

[8]. Hanlin Niu, Al Savvaris and Antonios Tsourdos. ***Verification for Decision Making System of Long Endurance Unmanned Surface Vehicle*** (Under internal review)

2 The Comparison of USV Path Following Algorithms

2.1 Abstract

This chapter presents the comparison and analysis of four common path following algorithms for the C-Enduro unmanned surface vehicle (USV), which is designed to operate at sea for extended periods of time (up to 3 months). Four path following algorithms include Carrot chasing path following, Nonlinear guidance law, Pure pursuit and line-of-sight (PLOS) path following and Vector field algorithms. The simulation was realized by implementing the 3 DOF dynamic model of C-Enduro USV and the control system. The simulation also took account of the environmental factors, i.e., wind and current. The cross-track error and control effort of these four algorithms were compared and analysed. The simulation results can be used to decide which path following algorithm C-Enduro USV needs to implement in order to deal with different missions efficiently.

2.2 Introduction

Most USV missions are straight line following and circular path following. The distance between the vehicle position and the predefined path is called cross-track error. The requirements of the path following algorithm are to reduce the cross-track error and minimize the difference between the vehicle course angle and the predefined course angle. The path following algorithms can be divided into two categories: geometric algorithms and control theory based algorithms.

The geometric algorithms are mostly simple to implement. The most common geometric algorithm is Carrot Chasing algorithm. The Carrot Chasing algorithm is to apply a virtual target point (VTP) on the path. When the vehicle is chasing the VTP, it will follow the path. Another path following method that is similar to Carrot Chasing algorithm is called nonlinear guidance law (NLGL) and Park et al has proved the Lyapunov stability of NLGL (Park, et al., 2007). Nonlinear guidance law also uses VTP concept but the VTP is the intersection between the circle that is around the USV and the predefined path. The other geometric algorithms include Line-of-Sight (LOS) algorithm, pure pursuit, and combination of pure pursuit and LOS (PLOS). These three algorithms have been proved in (Almeida, et al., 2009) (Conte, et al., 2004) and (Kothari, et al., 2010), respectively. The vector field method principle is to use vector field generate the flow direction, let the vehicle follow the vector field direction and the vehicle will follow the path.

The control methods includes PID based path following algorithm, Lyapunov vector field method, Linear Quadratic Regulator (LQR), Sliding Model Control (Ashrafiuon, et al., 2008) (Liao, et al., 2011), Model Predictive Control (MPC) (Naeem, et al., 2006), backstepping control (Bibuli, et al., 2007), gain scheduling theory, adaptive control and dynamic programming. The PID based path following algorithm has been demonstrated by Sun et al (2008), The Lyapunov stability argument has been demonstrated in (Nelson, et al., 2007). The LQR algorithm was proved by Wang et al (Wang, et al., 2010) and the LQR algorithm also takes account of the control error to drive the vehicle to follow the path and it minimizes the control effort to reduce the cross-track error. The adaptive control method was presented by Cao et al (Cao, et al., 2007).

In the USMOOTH project, Carrot Chasing, NLGL, PLOS and VF were selected due to their simplicity, robustness, and ease of implementation. These four algorithms were implemented and compared using the dynamic model of the C-Enduro USV and the control system. The cross-track error and the control effort were used to evaluate the performance of these four algorithms. This chapter is organized as follows: The USV dynamic model is introduced in Section 2.3. The hydrodynamic forces and moments are modelled in Section 2.4. The control system is presented in Section 2.5. The carrot chasing algorithm, nonlinear guidance law, pure pursuit and LOS algorithm, and vector field algorithm are introduced in Section 2.6, Section 2.7, Section 2.8 and Section 2.9, respectively. These four algorithms are simulated and compared in Section 2.10. The summary of this chapter is given in Section 2.11.

2.3 USV Dynamic Model

Because the two thrusters of the C-Enduro vessel are the two fixed pitch propellers on the left and right hull and the vessel has no rudder and fins on it, the control of the thrusters can only maneuver planar motions including surge, sway, and yaw. Thus, the C-Enduro model is designed to be 3DOF planar model neglecting heave, pitch, and roll dynamics.

According to the notation of (SNAME, 1950), the marine motion variables can be notated as Table. 2. 1.

Table. 2. 1. SNAME notation for marine vessels

DOF	Motions	Forces and moments	Linear and angular velocities	Position and Euler angles
1	Motion in x-direction (surge)	X	u	x
2	Motion in y-direction (sway)	Y	v	y
3	Motion in z-direction (heave)	Z	w	z
4	Rotation about the x-axis (roll)	K	p	ϕ
5	Rotation about the y-axis (pitch)	M	q	θ
6	Rotation about the z-axis (yaw)	N	r	ψ

Because we only concern about the planar motion, we will use the variables vector $[X, Y, N, u, v, r, x, y, \psi]^T$. The vehicle is also assumed to be both right-left and fore-after symmetric. By simplifying 6 DOF dynamics and kinematics equations of motion in (Fossen, 1994), the 3 DOF model can be given as below.

$$\dot{\eta} = J(\eta)\nu \quad \textbf{(2. 1)}$$

$$M\dot{\nu} + C(\nu)\nu + D(\nu)\nu + g(\eta) = \tau_E + \tau \quad \textbf{(2. 2)}$$

$$M = M_{RB} + M_A \quad \textbf{(2. 3)}$$

$$C(\nu) = C_{RB}(\nu) + C_A(\nu) \quad \textbf{(2. 4)}$$

Where

$\eta = [x, y, \psi]^T$

$\nu = [u, v, r]^T$

$J(\eta) =$ transformation matrix between the body coordinate and the earth-fixed coordinate system

$M =$ inertial matrix (including added mass M_A)

$C(\nu) =$ matrix of Coriolis and centripetal terms (including added mass $C_A(\nu)$)

$M_{RB} =$ rigid-body inertia matrix

$C_{RB}(\nu) =$ rigid-body coriolis and centripetal matrix

$D(\nu) =$ damping matrix

$g(\eta) =$ vector of gravitational forces and moments

$\tau_E =$ environmental forces and moments (including current and wind)

$\tau =$ propulsion forces and moments

2.4 Hydrodynamic Forces and Moments

The external forces and moments acting on the vehicle can be classified according to:

a. Radiation-induced forces, including added inertia, hydrodynamic damping and restoring forces.

b. Environmental forces, including ocean currents and wind.

c. Propulsion forces, including thruster/propeller forces.

2.4.1 Radiation- induced Forces

The inertial matrix M of the vessel consists of the rigid-body inertia matrix and the added mass M_A. Added mass is the inertia term added to the vehicle because the accelerating or decelerating vehicle must move some volume of surrounding fluid as the vehicle moves though the fluid. According to (Fossen, 1994), M can be represented as shown in (2. 5).

$$M = \begin{bmatrix} m - X_{\dot{u}} & 0 & 0 \\ 0 & m - Y_{\dot{v}} & -Y_{\dot{r}} \\ 0 & -N_{\dot{v}} & I_z - N_{\dot{r}} \end{bmatrix} \quad \textbf{(2. 5)}$$

Similarly, Coriolis and centripetal matrix consist of rigid-body Coriolis and centripetal matrix $C_{RB}(v)$ and added Coriolis and centripetal matrix $C_A(v)$, as shown in (2. 6).

$$C(v) = \begin{bmatrix} 0 & -mr & Y_{\dot{v}}v + \frac{Y_{\dot{r}} + N_{\dot{v}}}{2}r \\ 0 & 0 & (m - X_{\dot{u}})u \\ -Y_{\dot{v}}v - \frac{Y_{\dot{r}} + N_{\dot{v}}}{2}r & X_{\dot{u}}u & 0 \end{bmatrix} \quad \textbf{(2. 6)}$$

The hydrodynamic damping term $D(v)$ includes potential damping, skin friction, wave drift damping and damping due to vertex shedding. According to (Jørgensen, 2011), the hydrodynamic damping term can be summarized to a linear term $D_L(v)$ and a nonlinear term $D(v_r)$, v_r represents the relative linear velocity between the vehicle and the water. The linear term $D_L(v)$ can be represented as shown in (2. 7) and (2. 8).

$$D(v) = D_L(v) + D(v_r) \quad \textbf{(2. 7)}$$

$$D_L(\nu) = -\begin{bmatrix} X_u & 0 & 0 \\ 0 & Y_v & Y_r \\ 0 & N_v & N_r \end{bmatrix} \quad \textbf{(2. 8)}$$

The nonlinear term can be calculated by using the marine system simulator (MSS) Simulink toolbox, the surge resistance X and cross-flow drag Y and N can be calculated by using (2. 9), (2. 10) and (2. 11).

$$X = -\frac{1}{2}\rho A_x C_X |u_r| u_r \quad \textbf{(2. 9)}$$

where

ρ is the density of sea water, A_x is the frontal current area, C_X is the surge damping coefficient and u_r is the relative surge velocity.

Cross-flow drag is the sway damping term and the yaw damping term.

$$Y = -\frac{1}{2}\rho \frac{A_y}{L} C_Y \int_{-\frac{L}{2}}^{\frac{L}{2}} (v_r + xr)|v_r + xr| d_x \quad \textbf{(2. 10)}$$

$$N = -\frac{1}{2}\rho \frac{A_y}{L} C_Y \int_{-\frac{L}{2}}^{\frac{L}{2}} x(v_r + xr)|v_r + xr| d_x \quad \textbf{(2. 11)}$$

where

A_y is the transversal area, L is the length of the hull, C_Y is the transversal drag coefficient, v_r is the relative sway velocity and r is the angular velocity of yaw.

Because gravitational forces and moments $g(\eta)$ just have effect on the heave, pitch and roll motion, they will be ignored in this 3 DOF model.

2.4.2 Environmental Disturbances

In the modeling and control of the vehicle, the environmental disturbances, including the induced forces and moments from the ocean currents and wind, will make disturbance on the dynamic equation of motion. This section will present the environmental forces and moments acting on the vehicle.

2.4.2.1 Ocean Currents

Ocean currents are caused by several factors including the wind system over the sea surface, the heat exchange, the earth rotation and the planetary interactions like gravity. The ocean currents can be treated as a uniform movement of the water in the dynamic equations of motion. In the 3 DOF model, the ocean currents induced forces and moments are presented by using the relative speed between the unmanned surface vehicle and the water as shown in equations (2. 7) - (2. 11). The earth-fixed current velocity vector (including angular velocity) can be denoted by $[u_c^E, v_c^E, r_c^E]^T$ and the body -fixed components $[u_c, v_c, r_c]^T$ can be generated by equation (2. 12). Because the current movement is treated as uniform movement, the angular velocity r_c^E equals 0.

$$\begin{bmatrix} u_c \\ v_c \\ r_c \end{bmatrix} = \begin{bmatrix} cos(\psi) & sin(\psi) & 0 \\ sin(\psi) & -cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_c^E \\ v_c^E \\ r_c^E \end{bmatrix} \tag{2. 12}$$

Thus, the relative velocity vector ν_r, i.e. $[u_r, v_r, r_r]^T$, can be calculated by applying equation (2. 13).

$$\begin{bmatrix} u_r \\ v_r \\ r_r \end{bmatrix} = \begin{bmatrix} u \\ v \\ r \end{bmatrix} - \begin{bmatrix} u_c^E \\ v_c^E \\ r_c^E \end{bmatrix} \tag{2. 13}$$

Assume the current movement is uniformly movement, we can get equation (2. 14).

$$\dot{\nu} = \dot{\nu_r} \tag{2. 14}$$

Taking the current-induced forces and moments into consideration, equation (2. 2) is represented as in equation (2. 15).

$$M\dot{\nu} + C(\nu_r)\nu_r + D(\nu_r)\nu_r + g(\eta) = \tau_E + \tau \tag{2. 15}$$

τ_E does not include the current components, because the current components have been combined into the left side of the equation.

2.4.2.2 Wind

The wind effect on the USV motion can be described by a constant resultant force that will be determined by the relative speed scalar V_r between the USV and the wind, the frontal projected USV area A_T, the transverse projected area A_L and the force/moment coefficients C_X, C_Y and C_N. According to (Isherwood, 1973), the wind force and moment on surge, sway and moment motions can be calculated by equations (2. 16) (2. 17) (2. 18).

$$X_{wind} = \frac{1}{2} C_X \rho_a u_r^2 A_T \tag{2. 16}$$

$$Y_{wind} = \frac{1}{2} C_Y \rho_a v_r^2 A_L \tag{2. 17}$$

$$N_{wind} = \frac{1}{2} C_N \rho_a V_r^2 A_L L \tag{2. 18}$$

L is the length of the hull and ρ_a is the density of the air.

2.4.3 Propeller Model

The C-Enduro unmanned surface vehicle utilizes catamaran structures, and it has two propellers on the rear of each hull. Because the C-Enduro USV has no rudder or fins on it, the propulsion system will only be the two fixed pitch propellers. The vehicle speed will be controlled by regulating the propeller revolution speed. According to the bilinear thruster model in (Blanke, 1981), the thrust can be calculated by using equation (2. 19).

$$T = \rho D^4 (\alpha_1 + \alpha_2 J_0)|n|n \tag{2. 19}$$

T is the thrust of the propeller, ρ is the density of sea water, D is the diameter of the propeller, α_1 and α_2 are two constants, n is propeller revolutions per second (rps) and J_0 is the open water advance coefficient. J_0 can be calculated by equation (2. 20).

$$J_0 = \frac{V_a}{nD} \qquad \textbf{(2. 20)}$$

V_a is the speed of the water flowing through the propeller, the difference between the water flowing speed V_a and the vehicle speed u is called wake number w as shown in equation (2. 21).

$$V_a = (1 - w)u \qquad \textbf{(2. 21)}$$

According to (Caccia & Veruggio, 2000), when the USV is operating at low speed, the bilinear model can be approximated by an affine model, i.e. the advance speed is zero, as shown in equation (2. 22).

$$T = a_n|n|n \qquad \textbf{(2. 22)}$$

The parameter a_n, which has different values for different rotation speeds, can be tested in the experiment. Therefore, the thrust can be mapped to propeller rate $n = \sqrt{|T|/a_n}$, which can be used for optimizing the energy consumption in the future work. The speed of the vehicle is controlled by the sum of the two propeller thrusts that can be denoted as T_s.The yaw angle is controlled by regulating the difference of two propellers thrusts, because the difference of two propellers thrusts result in yaw moment. The starboard thrust is denoted by T_1, the port thrust is denoted by T_2 and the moment N_d can be calculated by equation (2. 23).

$$N_d = (T_1 - T_2)\frac{d}{2} \qquad \textbf{(2. 23)}$$

Where d is the distance between two propellers.

Therefore, the propeller force and moment vector τ can be calculated by using equation (2. 24).

$$\tau = \begin{bmatrix} T_1 + T_2 \\ 0 \\ (T_1 - T_2)\frac{d}{2} \end{bmatrix} \tag{2.24}$$

In order to denote the propeller force and moment vector component more efficiently, we will denote $T_1 + T_2$ as T_s , denote $(T_1 - T_2)\frac{d}{2}$ as N_d and denote $T_1 - T_2$as T_d in the following section.

2.5 Speed Controller and Heading Controller Design

Applying equation (2.2) and neglecting the environmental disturbance influence, we can describe the speed nonlinear dynamics using equation (2. 25).

$$m(\dot{u} - vr) = X_{\dot{u}}\dot{u} + X_u u + X_{vr}vr + X_{rr}r^2 + X_{|u|u}|u|u + T_s \tag{2.25}$$

Neglecting the coupled term and the yaw motion, we can get equation (2. 26).

$$(m - X_{\dot{u}})\dot{u} = X_u u + X_{|u|u}|u|u + T_s \tag{2.26}$$

According to the gain-scheduling controller design method presented in Khalil (1996), the controller can be specified into a list of constant operating points. In each operating point, the controller provides a feed-forward constant control value T_{s1} and a feed-back control value T_{s2}. The feed-forward constant value is used for yielding zero error. The feed-back action assigns a desired characteristic equation to the closed-loop linearized form. Therefore, we can get the controller from (2. 27).

$$T_s = T_{s1} + T_{s2} \tag{2.27}$$

For every desired speed value u_0, the feed-forward thrust value T_{s1} can be obtained using equation (2. 28).

$$T_{s1} = -X_u u_0 - X_{|u|u}|u_0|u_0 \tag{2.28}$$

The feed-back control design can be achieved by linearizing equation (2. 29) around the desired speed value u_0.

$$(m - X_{\dot{u}})\Delta\dot{u} = (X_u + 2X_{|u|u}u_0)\Delta u - T_{s2} \tag{2. 29}$$

Where,

$$\Delta u = u_0 - u \tag{2. 30}$$

In order to force the speed error Δu to be zero, a PI speed controller is designed. Proportional term will be used for compensating the velocity change by regulating the thrust. Integral term will be used to eliminate the steady state error. This controller can compensate the environmental disturbance. Assume the desired characteristic equation is (2. 31).

$$s^2 + 2\sigma s + \sigma^2 + \omega_n^2 = 0 \tag{2. 31}$$

The PI controller form:

$$T_{s2} = k_p\Delta u + k_i\gamma \tag{2. 32}$$

$$\dot{\gamma} = \Delta u \tag{2. 33}$$

Substituting equation (2. 32) and (2. 33) into equation (2. 29), we can get equation (2. 34).

$$(m - X_{\dot{u}})\Delta\dot{u} = (X_u + 2X_{|u|u}u_0)\Delta u - k_p\Delta u - k_i\int \Delta u\, dt \tag{2. 34}$$

Applying Laplace transform to equation (2. 34) and compare it with equation (2. 31), we can get k_p and k_i. The controller gains k_p and k_i can be designed as below.

$$k_p = X_u + 2X_{|u|u}u_0 + 2(m - X_{\dot{u}})\sigma$$

$$k_i = (m - X_{\dot{u}})(\sigma^2 + \omega_n^2)$$

Regarding the heading control, PD controller will be used in this research to regulate the course angle of the USV. Firstly, we simplify the steering model, by combing the yaw moment term of equation (2. 5) to (2. 11), neglecting the coupled term and treating the environmental moments as disturbance, we can get (2. 35).

$$I_r\dot{r} = X_r r + N_d \tag{2. 35}$$

Applying equation (2. 35) to (2. 36), we can get equation (2. 37).

$$r = \dot{\psi} \tag{2. 36}$$

$$T\ddot{\psi} + \dot{\psi} = KT_d \tag{2. 37}$$

Then consider a PD control law.

$$T_d = K_p(\psi_d - \psi) - K_d\psi \tag{2. 38}$$

$K_p$and K_d are the controller design parameters. Combining equation (2. 37) and (2. 38), we can get equation (2. 39).

$$T\ddot{\psi} + (1 + KK_d)\dot{\psi} + KK_p\psi = KK_p\psi_d \tag{2. 39}$$

The equation (2. 39) corresponds to a 2nd-order system in the following form.

$$\ddot{\psi} + 2\zeta\omega_n\dot{\psi} + \omega_n^2\psi = \omega_n^2\psi_d \tag{2. 40}$$

ω_n is the natural frequency and ζ is the relative damping ratio, which can be treated as design parameters.

From equation (2. 39) and equation (2. 40), we can get K_p and K_d.

$$K_p = \frac{T\omega_n^2}{K};\ K_d = \frac{2T\zeta\omega_n - 1}{K}$$

2.6 Carrot Chasing

Carrot chasing path following algorithm is most widely used and easy to implement. To follow a predefined path, a virtual target point (VTP) is introduced on the path. The USV will always follow the VTP and the VTP is called carrot, therefore, this algorithm is called carrot chasing algorithm.

In Fig. 2. 1, the straight line is represented by using the line between two waypoints W_i and W_{i+1}. The USV position is denoted by $p\ (x, y)$ and the course angle is denoted by ψ. The projection of p on line AB is q. The distance between p and q is denoted by d. The VTP is denoted as s. The distance between q and s is δ, which needs to be defined by the user.

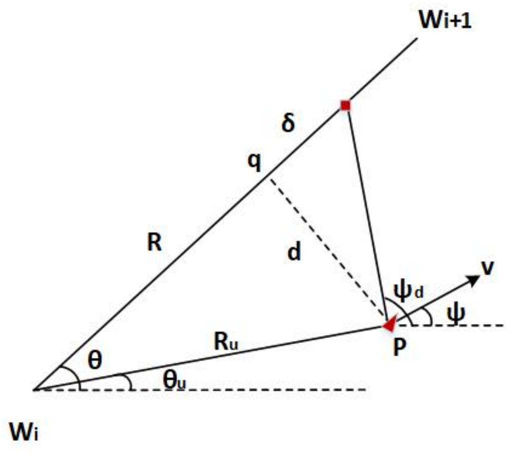


Fig. 2. 1. Carrot chasing straight line following algorithm

$$R_u = \|W_i - p\| \tag{2. 41}$$

$$\theta = \operatorname{atan}\left(\frac{y - y_i}{x - x_i}\right) \tag{2. 42}$$

$$\beta = \theta - \theta_u \tag{2. 43}$$

$$R = R_u \cos(\beta) \tag{2. 44}$$

Then, the position of VTP $s\ (x_t, y_t)$ is given.

$$x_t = (R + \delta\ ) \cos(\theta) \tag{2. 45}$$

$$y_t = (R + \delta\ ) \sin(\theta) \tag{2. 46}$$

The desired course angle ψ_d is given.

$$\psi_d = atan\left(\frac{y_t - y}{x_t - x}\right) \tag{2. 47}$$

The desired course angle turning rate $\dot{\psi}$ is given.

$$\dot{\psi} = \frac{\psi_d - \psi}{\Delta t} \tag{2. 48}$$

When the USV is following the predefined path, the path following algorithm is calculating iteratively. The logic is given in Fig. 2. 2. Firstly, calculate out the cross-track error d. Secondly, update the VTP position. Finally, update ψ_d and $\dot{\psi}$.

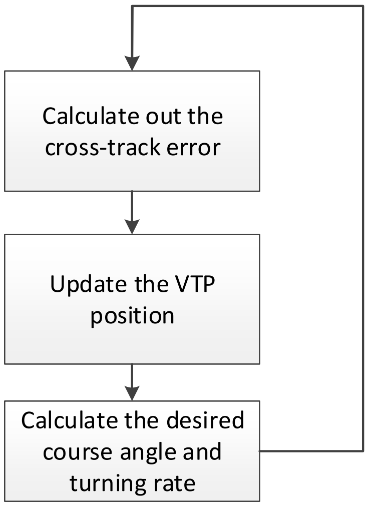

Calculate out the cross-track error
Update the VTP position
Calculate the desired course angle and turning rate

Fig. 2. 2. The logic of carrot chasing algorithm

2.7 Nonlinear Guidance Law

The Nonlinear guidance law (NLGL) also uses the concept VTP for USV to follow. However, unlike carrot chasing that uses a preconfigured value δ or λ to make the USV move forward, NLGL employs the intersection between a given radius circle around the USV position and the predefined path as the position of VTP.

The geometry of NLGL straight line following has been shown in Fig. 2. 3, where $P(x, y)$ is the position of the USV. The given radius of the USV circle is denoted by L. The two intersections between the circle and the straight line are noted as s and s'. The positions of the two waypoints W_i and W_{i+1} are denoted as (x_i, y_i) and (x_{i+1}, y_{i+1}).

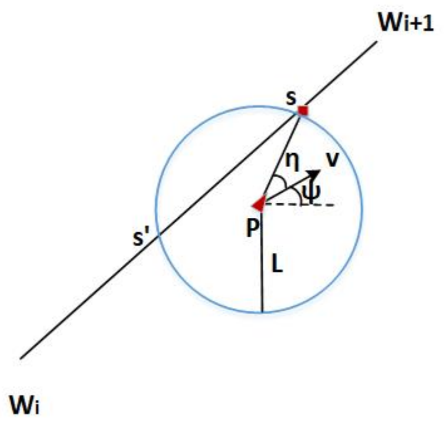


Fig. 2. 3. NLGL straight line following algorithm

The following two equations will give the two intersections position solutions. Assume the VTP position is denoted by $s(x_t, y_t)$.

$$(x_t - x)^2 + (y_t - y)^2 = L^2 \tag{2. 49}$$

$$y_t = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} x_t - \frac{y_{i+1} - y_i}{x_{i+1} - x_i} x_i + y_i \tag{2. 50}$$

There will be two solutions for $s(x_t, y_t)$ and the one that is closer to W_{i+1} is the VTP. By calculating the line-of-sight angle from USV position $p\ (x, y)$ to $s(x_t, y_t)$, we get the desired course angle ψ_d.

$$\psi_d = atan(\frac{y_t - y}{x_t - x}) \tag{2. 51}$$

Then, the desired course angle turning rate is given.

$$\dot{\psi} = \frac{\psi_d - \psi}{\Delta t} \tag{2. 52}$$

The logic of the NLGL algorithm is the same as that of carrot chasing algorithm, as shown in Fig. 2. 2.

2.8 Pure pursuit and LOS

Pure pursuit and line-of-sight (PLOS) path following is a combination between pure pursuit guidance and line-of-sight guidance laws. It not only drives the USV to the destination using pure pursuit guidance laws but also minimizes the cross-error d using LOS guidance. The algorithm of PLOS straight line following is shown in Fig. 2. 4.

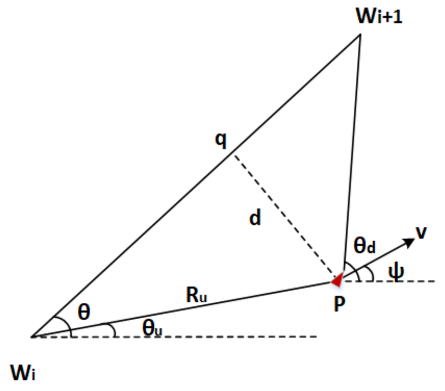


Fig. 2. 4. PLOS straight line following algorithm

The line-of-sight angle between USV position $p\ (x, y)$ and the destination W_{i+1} is denoted by θ_d.

$$\theta_d = atan\frac{y_{i+1} - y}{x_{i+1} - x} \tag{2. 53}$$

$$d = \frac{(y_i - y_{i+1})x + (x_{i+1} - x_i)y + (x_i y_{i+1} - x_{i+1} y_i)}{\|W_i - W_{i+1}\|} \tag{2. 54}$$

$$\psi_d = k_1(\theta_d - \psi) + k_2 d \tag{2. 55}$$

k_1 and k_2 are two weighted factors that should be defined by the users. Then, the desired course angle turning rate can be calculated out.

$$\dot{\psi} = \frac{\psi_d - \psi}{\Delta t}$$

2.9 Vector Field Based Path Following

The principle of vector field (VF) path following is that the unmanned vehicles will follow the direction of the vector field. The vector field path following method can be used for straight line following and loiter. Lyapunov stability arguments are applied to demonstrate the stability of VF path following algorithm that has been presented by Nelson et al (Nelson, et al., 2007).

One demonstration of VF straight line following has been done in Matlab. Fig. 2. 5 shows the principle of VF straight line following algorithm.

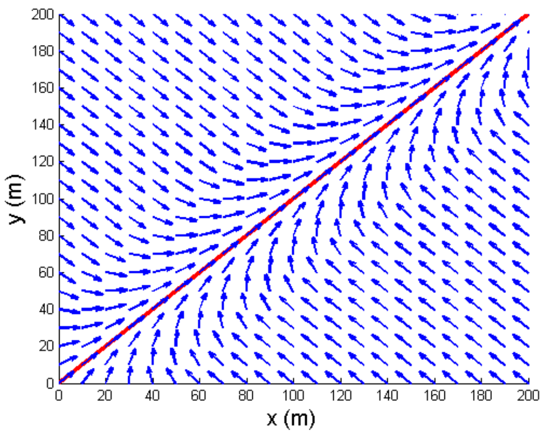


Fig. 2. 5. Vector Field straight line following algorithm

In Fig. 2. 5, the red line represents the predefined path and the blue arrows represent the vector field direction. The principle of the vector field straight line following algorithm is that when the vehicle is inside the boundary τ that is close to the predefined path, the desired course angle will be updated to keep the vehicle follow the path. When the vehicle is far away from the path, the vehicle will move vertically to the path until it moves into the boundary.

The equations used for deriving the desired course angle turning rate has been given as below:

The line-of-sight angle of the straight path is denoted by θ.

$$\theta = atan\frac{y_{i+1} - y_i}{x_{i+1} - x_i} \quad \textbf{(2. 56)}$$

$$s^* = \frac{(p - W_i)^T (W_{i+1} - W_i)}{||W_{i+1} - W_i||^2} \tag{2. 57}$$

$$\varepsilon = ||p - (s^*(W_{i+1} - W_i) + W_i)|| \tag{2. 58}$$

$$\rho = sign[(W_{i+1} - W_i) \times (p - W_i)] \tag{2. 59}$$

$$\varepsilon = \rho\varepsilon \tag{2. 60}$$

There are two situations when the vehicle is inside the boundary or outside.

When the vehicle is inside the boundary τ,

$$\psi_d = \theta - \rho\chi^e \tag{2. 61}$$

When the vehicle is outside the boundary τ,

$$\psi_d = \theta - (\chi^e)\left(\frac{\varepsilon}{\tau}\right)^k - (\frac{k\chi^e v_a}{\alpha\tau^k})\varepsilon^{k-1} sin\psi \tag{2. 62}$$

In these equations, χ^e, k and τ should be defined by the users. The higher the χ^e is, the quicker the USV will move toward the LOS line. If χ^e is small, the cross-track error will also be large.

2.10 USV Path Following Simulation and Result Analysis

The USV dynamic model, the environmental disturbance model, the control system and the path following guidance system are integrated using Simulink. The performances of the path following algorithms are compared under two situations. The first scenario is simulated without wind and current. The second scenario is simulated in the presence of wind and current disturbances.

In the first case, the wind speed V_w and the current speed V_c were assumed to be $0\ m/s$. The initial heading angle was assumed to be true north. The predefined path was from (0,0) to (1000,1000). The USV initial position was assumed to be (0,0), the initial heading angle is assumed to be $90^{\text{o}}$and the USV was assumed to be travelling at a constant speed $2\,m/s$. The simulation duration was 500 seconds. The parameters of each algorithm are shown in Table. 2. 2.

In the second case, both wind and current disturbances were taken into account. The wind speed V_w was assumed to be $3\ \mathrm{m}/s$ and the sea current speed V_c was assumed to be $0.2\ \mathrm{m}/s$. The wind and sea current directions were assumed to be 270^{o}.

Table. 2. 2. Path following parameters

Carrot Chasing (CC)	Nonlinear Guidance (NLGL)	Vector Field (VF)	Pure pursuit and LOS (PLOS)
δ = 10	L = 15	$\chi^e = \frac{\pi}{2}$ τ = 35 k = 1 α = 20	k_1 = 1 k_2 = 0.1

To evaluate the performance of the algorithms, two criteria are used. The first criterion is total cross-track error D. The second criterion is total control effort U.

$$D = \sum_{t=0}^{t=T} d(t)^2$$

$$U = \sum_{t=0}^{t=T} \Delta\psi(t)^2$$

The trajectory of the Carrot Chasing algorithm when there is no disturbance is shown in Fig. 2. 6 and the cross-track error is shown in Fig. 2. 7. The total cross-track error D and the total control effort U of the four algorithms for both situations are shown in Table. 2. 3.

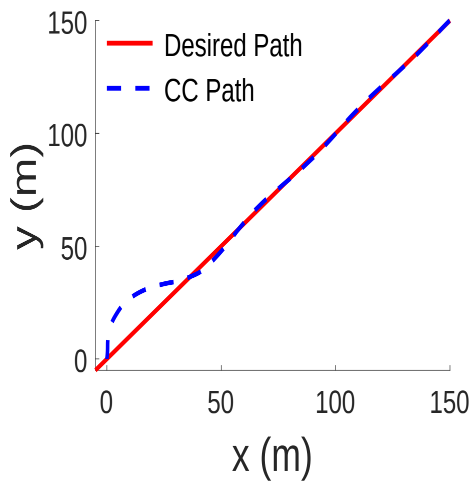


Fig. 2. 6. Carrot chasing path

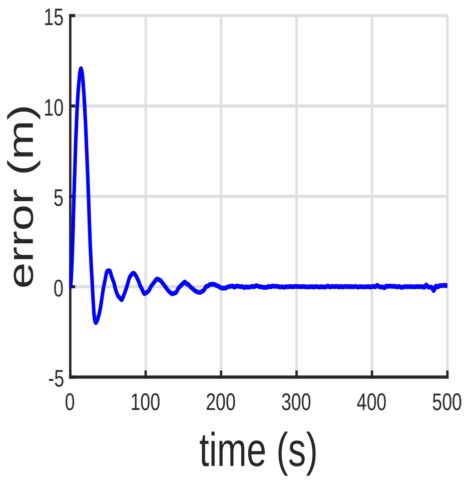


Fig. 2. 7. The cross-track error of carrot chasing algorithm

Since the main contribution to the total cross-track error was caused when the USV was in the process of turning from initialized heading angle. To remove this bias from the data, the cross-track error from 100s to 500s was calculated to show how much error the path following algorithm generated after the USV reached the path.

From Table. 2. 3, it can be seen that, in terms of cross-track error, when there is no wind and current, Carrot Chasing algorithm provided the most accurate results. However, in the presence of wind and current disturbances, nonlinear guidance algorithm provided the most stable guidance. In terms of total control effort, vector field method needs the least control effort to control the USV to follow the path, which means the USV did not change its control command very often.

The characteristics of these path following algorithms can be used to design a switchable guidance system that when the USV needs more accuracy of following the path, the guidance system needs to be switched to nonlinear guidance algorithm; when the USV needs to save energy as a priority, the guidance system can be switched to vector field algorithm.

Table. 2. 3. Total cross-track error D and total control effort U

	Carrot Chasing	Nonlinear Guidance	Pure pursuit and LOS	Vector field
D (100s to 500 s) (no wind no current)	5.4	31.9	5.8	29.2
D (100s to 500 s) (wind current)	431.9	83.5	415.5	186.6
U (no wind no current)	11.4	10.4	17	5.9
U (wind current)	10.7	9.6	14.9	5.7

2.11 Summary

This chapter presents the implementation and comparison of four path following algorithms, including Carrot Chasing algorithm, Nonlinear Guidance Law algorithm, Pure Pursuit and LOS (PLOS), and Vector Filed algorithm. In this chapter, the 3 DoF dynamic model of the C-Enduro USV and the control system were developed. The simulation also took account of the environmental factors including wind and sea current disturbances. The accuracy and control effort of these four algorithms were compared and analyzed in two scenarios. The simulation results can be used to decide which path following algorithm the C-Enduro USV needs to implement in order to deal with different missions efficiently.

3 Voronoi-Visibility Shortest Path Planning Algorithm

3.1 Abstract

The Voronoi-Visibility shortest path planning algorithm is covered in this chapter. It is known, Voronoi diagram has the advantage of computational efficiency and it can generate roadmap in $O(nlogn)$ time, where n is the number of the island vertices, however, the quality of the path is far from optimal. The Visibility graph algorithm can generate optimal path, however, the computational time is $O(n^2)$ time. In dealing with the large spatial dataset, the Visibility graph method is impractical. The proposed algorithm integrates the Voronoi diagram, the Visibility graph and the Dijkstra search algorithm. In this work, the VM (Voronoi shortest path refined by Minimizing the number of waypoints) algorithm was applied for comparison. The proposed Voronoi-Visibility algorithm and the VM algorithm were compared in ten Singapore USV missions and five Croatia USV missions. To test the computational time of the proposed algorithm, high-resolution and large spatial dataset were used, containing 98 Singapore islands and 114 Croatian islands. In the Singapore missions, the quality and computational time of the Voronoi-Visibility shortest path planning algorithm were compared with the VM algorithm. It is demonstrated that the proposed algorithm not only improves the quality of the Voronoi shortest path and also keeps the computational efficiency of the Voronoi diagram approach. Moreover, the proposed algorithm ensures the USV safety by keeping the USV at a configurable clearance distance c from the coastlines. The results of the five missions in Croatian islands demonstrated the flexibility of the proposed algorithm in different geographical scenario.

3.2 Introduction

USV path is not commonly a straight line since it is affected by the USV dynamics, efficiency, avoiding threat areas and timing constraints. Trajectories should be achievable within the dynamic constraints of the vehicles. Depending on the environment, the path planning algorithms can be classified into four categories: roadmap approaches, cell decomposition, potential fields, and bug algorithms.

Firstly, the goal of the roadmap approaches is to reduce the N-dimensional configuration space to a set of one-dimensional paths to search. The roadmap method attempts to capture the free-space connectivity with a graph. The sampling-based roadmap method includes probabilistic roadmap method (Amato & Wu, 1996), rapidly exploring random tree (Kuffner & Latombe, 2000), expansive space planner (Hsu, et al., 1997) and random walk planner (Carpin & Pillonetto, 2005). Some other well-known roadmap based approaches based on computational geometry include Voronoi diagram approach and Visibility graph approach. While the use of Voronoi diagram for USV dynamic path planning was presented in (Wu, et al., 2013). The Visibility graph in determining the shortest path was proved in (Kaluđer, et al., 2011). The advantage of Voronoi diagram is the computational efficiency and has the maximum clearance distance *c* while the disadvantage is the Voronoi roadmap is far from optimal and it includes redundant waypoints. The advantage of Visibility algorithm is it can generate an optimal path and the disadvantage is the computing time is much longer than Voronoi diagram. One Voronoi diagram is shown in Fig. 3. 1 and one Visibility graph example is shown in Fig. 3. 2.

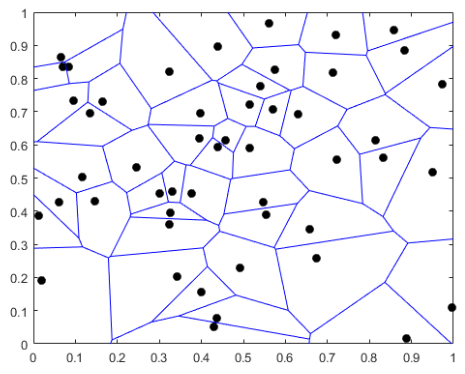


Fig. 3. 1. Voronoi diagram

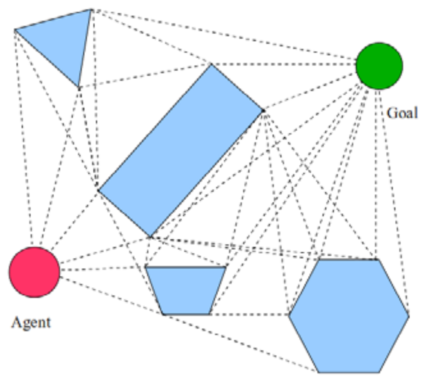


Fig. 3. 2. Visibility graph

In Fig. 3. 1, the blue lines divide the plane into different cells. In each cell, all the vertices of this cell are closer to the corresponding black point than the other black points in other cells. The edges of two adjacent regions are comprised of points equidistant from the two given points. Therefore, the set of lines equidistant from multiple points forms the Voronoi diagram. In Fig. 3. 2, all the lines '--' are the candidate paths for the vehicle, the vehicle can travel along the edges of the obstacles. It can be seen, a Visibility graph is a graph of inter-visible locations, typically for a set of points and obstacles in the Euclidean plane.

Another method is the Cell Decomposition approach, which includes exact cell decomposition approach and approximate cell decomposition approach. In (Li, et al., 2011), cell decomposition approach is applied for UAV path planning. The exact cell decomposition approach can also be called trapezoidal decomposition. The first step of the trapezoidal decomposition is decomposing the free space into trapezoidal and triangular cells, then it processes the adjacency relation between the cells and finds the optimal path in the end. The approximate cell decomposition approach is also called Quadtree Decomposition approach. The Quadtree Decomposition approach is to subdivide the mixed obstacle and free region into four quarters and repeat the subdividing. At a certain level of resolution, only the cells whose interiors lie entirely in the free space are used. Thereafter use a graph-search algorithm to find a path from the starting point to goal or target point; for example, the A* algorithm can be used to carry-out this operation. The application of A* algorithm for unmanned surface vehicle avoiding underwater obstacles was tested experimentally in (Phanthong, et al., 2014). A* is an informed search algorithm and it solves problems by searching among all possible paths to the destination for the one that incurs the smallest cost (least distance traveled, shortest time, etc.). To optimize the cost, A* algorithm minimizes

$$f(n) = g(n) + h(n)$$

Where n is the last node on the path, $g(n)$ is the cost of the path from the starting node to n, and $h(n)$ is a heuristic that estimates the cost of the cheapest path from n to the goal. Fig. 3. 3 is an illustration of A* path re-planning algorithm (Campbell, et al., 2012), which allows the USV to avoid the island.

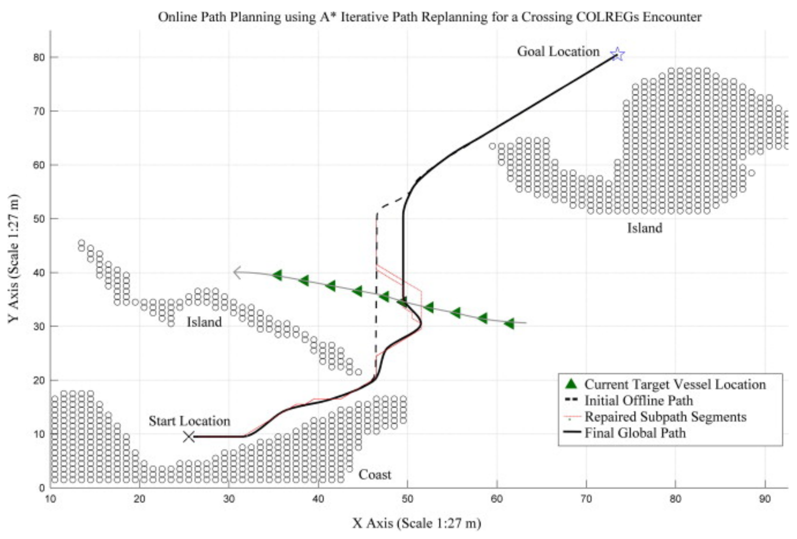


Fig. 3. 3. An illustration of A* path re-planning algorithm (Campbell, et al., 2012)

The Potential Field method is also widely used because the computational load required to generate the trajectory is small. In potential field method, the attractive potential field is assigned to the target and the repulsive potential field represents the obstacles, as shown in Fig. 3. 4 and Fig. 3. 5. The vessel is repelled to the target. The combination of attractive potential field and repulsive potential field is illustrated in the Fig. 3. 6. By using potential field method, the trajectory can be generated in real-time, moreover, planning and control are merged into one function. The Potential Field path planning can be coupled directly to a control algorithm. However, Potential Field method has also its drawbacks; it may be trapped in local minima in the potential field. This limitation is presented in (Koren & Borenstein, 1991). Due to this local minima limitation, it has been mainly used for local path-planning.

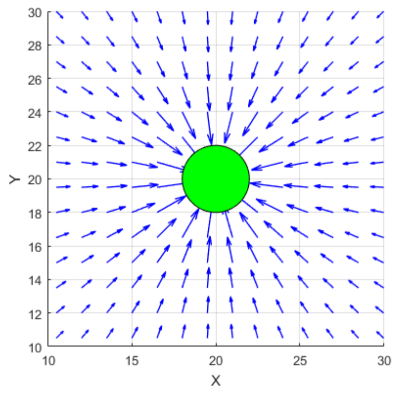


Fig. 3. 4. Attractive potential field

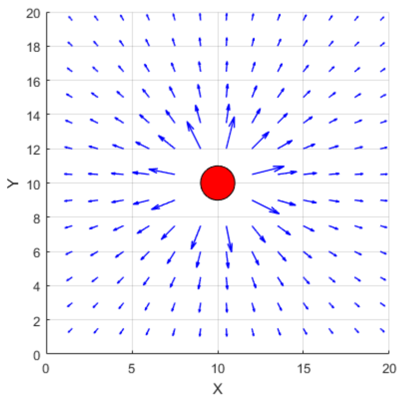


Fig. 3. 5. Repulsive potential field

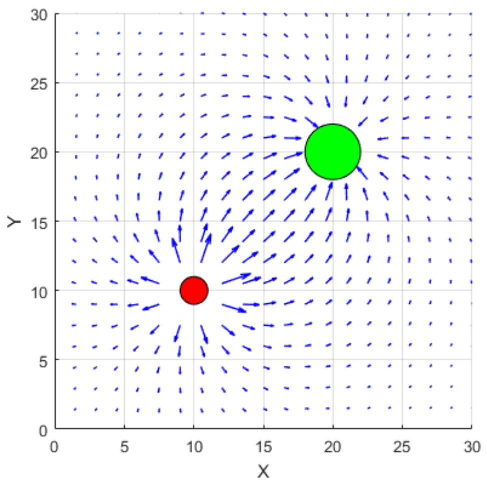


Fig. 3. 6. Combination of attractive potential field and repulsive potential field

Finally, the Bug Algorithms is a limited-knowledge path planning approach. The algorithms assume only local knowledge of the environment and a global goal. One example of the Bug Algorithms is insect-inspired "Bug" algorithm. We only know the direction to the goal using only local sensing information. This Bug Algorithm includes three steps: (1) heading toward the goal; (2) follow obstacles until you can head toward the goal again; (3) continue the previous two steps. The efficiency of Bug Algorithm for robot path planning using range sensor was presented by Buniyamin et al (Buniyamin, et al., 2011). The Bug Algorithms belongs to the methods for local path planning, i.e., obstacle avoidance. The obstacle avoidance algorithms include Vector Field Histogram (VFH), Vector Field Histogram+ (VFH+), the Bubble Band concept, basic Curvature Velocity Methods (CVM), lane Curvature Velocity Methods, and Dynamic Window approach, etc. These obstacle avoidance algorithms were reviewed in (Loe, 2008).

This chapter is organized as follows: Section 3.3 presents the literature review; Section 3.4 defines the problem of developing the path planning algorithm. The details of the methodology are introduced in Section 3.5. The proposed Voronoi-Visibility shortest path planning algorithm and the VM algorithm are simulated and compared in Section 3.6. Conclusions are given in Section 3.7.

3.3 Literature Review

The work will focus on the roadmap-based path planning approach. The advantage of using the Voronoi diagram as a roadmap, among which the visibility graph prevails, is its efficiency. The Voronoi diagram can be constructed in just $O(nlogn)$ time, where n is the number of the vertices. The fastest known algorithm for constructing Visibility graph takes $O(n^2)$ time (Ghosh & Mount., 1991) and it has $O(n^2)$ edges in the worst case. Since Voronoi diagram has $O(n)$ edges, searching a Voronoi diagram based roadmap is much faster than searching a visibility graph. Another advantage of Voronoi diagram is that the constructed roadmap will keep the path away from the obstacles as far as possible, while Visibility graph will keep the path as near as possible to the obstacles, since the Visibility graph uses the edges of the obstacles as possible paths. The disadvantage of the Voronoi diagram is that the generated path may be far from optimal. Therefore, to take the advantage of the computational efficiency of the Voronoi diagram, the generated path needs to be refined.

A general method for refining a path obtained from a roadmap based on classical numerical optimization techniques has been proposed in (Kim, et al., 2003). A Dijkstra's search algorithm was proposed to determine an optimal path, the edges those are closer to the obstacles will be assigned higher costs. However, this method will not generate an optimal path, since the path is constrained to the edges in the roadmap. A B-Spline approximation method was used to improve the smoothness of the path obtained from the roadmap (Ibarra-Zannatha, et al., 1994).

In the work (Wein, et al., 2007), a new diagram called $VV^{(C)}$ diagram was proposed. The $VV^{(C)}$ diagram integrated Visibility-Voronoi diagram and can also keep the path away from the obstacle with clearance distance c, which is a configurable value. However, this algorithm is Visibility graph based, the processing time is $O(n^2 log(n))$, which is impractical for large spatial datasets. In (Masehian & Amin-Naseri, 2004), the Voronoi diagram, Visibility graph and potential field are integrated into a single architecture to provide a parametric tradeoff between the safest and shortest path and the generated paths are shorter than then Voronoi and potential field methods, and faster than the Visibility graph. However, this algorithm is fairly complicated, and although the path length is shorter than those generated by the Voronoi diagram and potential field method, it still contains bumps and rudimentary turns. A Voronoi diagram based shortest path planning algorithm was proposed in (Bhattacharya & Gavrilova, 2008), the generated Voronoi shortest path was refined by minimizing the number of the waypoints and the final path was finally smoothed by using corner-cutting technique. It was demonstrated that the proposed algorithm reduced the length of the Voronoi path and still kept the computational efficiency $O(nlog(n))$.

A Voronoi-Visibility shortest path planning algorithm is proposed in this thesis by integrating the Voronoi diagram and the Visibility graph. The Voronoi diagram was first implemented to generate a collision-free roadmap with clearance distance c, which is a configurable value. Then the Dijkstra search algorithm was implemented to search for the Voronoi shortest path. The generated Voronoi shortest path contained redundant waypoints and still need to be refined. Subsequently, the Visibility graph was applied to refine the Voronoi shortest path, and the Voronoi-Visibility shortest path was generated finally by applying

Dijkstra's search algorithm again. To demonstrate the performance of the proposed algorithm, the proposed algorithm was compared with the VM algorithm (Voronoi shortest path planning algorithm that is refined by minimizing the number of the waypoints), which was proposed by (Bhattacharya & Gavrilova, 2008). The proposed Voronoi-Visibility algorithm and the VM algorithm were finally tested in ten Singapore Strait mission scenarios and five Croatian Islands mission scenarios. The path length and the computational time of these two algorithms were compared, it is demonstrated that the proposed algorithm can generate shorter path than the Voronoi shortest path planning algorithm and will not reduce the computational efficiency of Voronoi diagram approach. The proposed algorithm can also keep the USV a configurable clearance distance from the coastlines.

3.4 Problem Statement

The USV path planning algorithm for long range mission mainly includes two challenges: Firstly, keeping the computational efficiency in processing large spatial dataset; Secondly, keeping a clearance distance from the island coastlines.

3.4.1 Large Spatial Dataset

The realistic and high-resolution data will improve the accuracy of the path planning algorithm result, but will also cause a computational burden. For example, there are 98 islands in Singapore Strait and 114 islands in the considered coast of Croatian islands. The Singapore Strait map and the Croatian islands map are shown in Fig. 3. 7 and Fig. 3. 8, respectively. By querying the high-resolution islands data in Singapore Strait, it was found that there were around 4128 vertices for representing Singapore islands, even using the fastest Visibility graph to construct the roadmap, it will cost $O(n^2)$ time and edges number will be $O(n^2)$ in the worst case. Each edge needs to be checked whether it crosses any island coastlines and the large number of the edges will slow down the roadmap construction time and also the shortest path's searching time. Therefore, it is impractical to use the Visibility graph directly due to its inefficiency and the proposed algorithm should be able to process the large spatial dataset in a computationally efficient way.

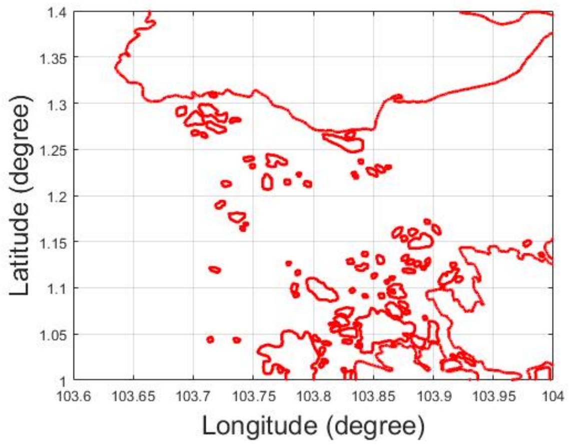


Fig. 3. 7. Singapore islands

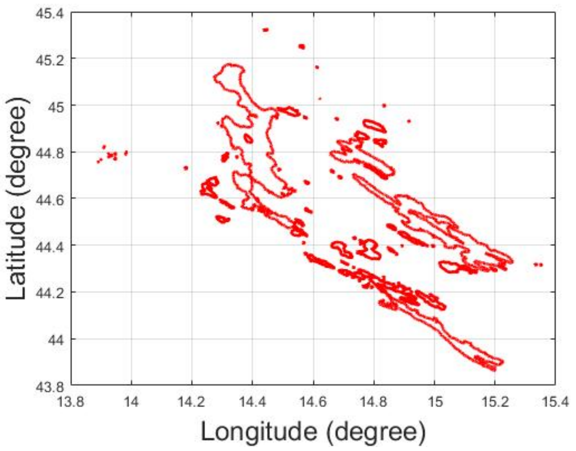


Fig. 3. 8. Croatia islands

3.4.2 Clearance Distance c

In some cases, the map data may be inaccurate. For example, in Fig. 3. 9, a Croatian island is shown and the data is obtained from the Global Self-Consistent Hierarchical High-Resolution Shorelines (GSHHS) dataset.

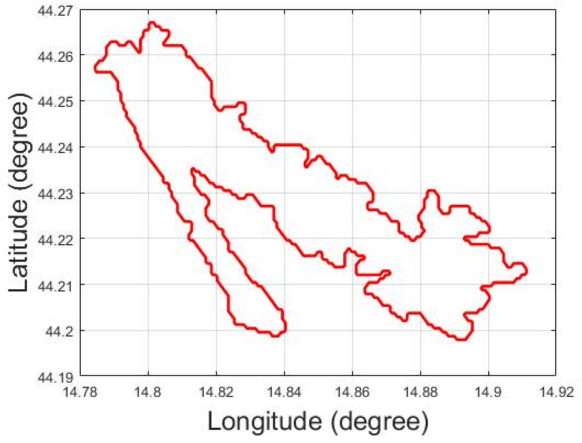


Fig. 3. 9. Croatian island

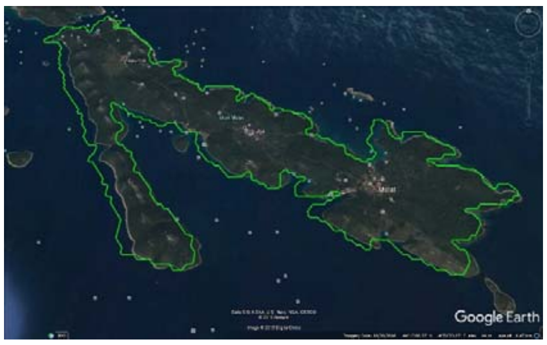


Fig. 3. 10. Illustration of data inaccuracy in Croatia

However, when the island data is mapped into google map in Fig. 3. 10, it can be seen that the island has two profiles from different map providers. The distance between these two profiles is around 200 meters. Therefore, the path planning algorithm should keep the USV a clearance distance from the island coastlines to address the problem of the map inaccuracy and the clearance distance can also keep the USV away from the crowded environment nearby the coastlines.

3.5 Methodology

The methodology of the approach used will be covered in this section. This section is organized as follows: Section 3.5.1 summarizes the architecture of the proposed algorithm. Section 3.5.2 presents the environmental dataset used in this research. The Voronoi collision-free roadmap generation is described in Section 3.5.3 and the Voronoi shortest path generation is presented in Section 3.5.4. Visibility graph generation approach is described in Section 3.5.5. Finally, the Voronoi-Visibility shortest path generation is introduced in Section 3.5.6.

3.5.1 Algorithm Architecture

The architecture of the proposed Voronoi-Visibility shortest path planning approach is shown in Fig. 3. 11. This algorithm includes four parts: the collision-free Voronoi roadmap generation, the Voronoi shortest path generation, the Visibility graph generation and Voronoi-Visibility shortest path generation.

In the collision-free Voronoi roadmap generation section, the spatial dataset is processed by the coastline expanding algorithm and the expanded coastlines will keep the subsequently generated roadmap a clearance distance *c* from the original coastlines. The extended coastline data is processed by the Voronoi diagram, the generated edges those cross the obstacles are removed and finally the collision-free Voronoi roadmap is generated. In the Voronoi shortest path generation section, the starting point and the destination are inserted into the Voronoi roadmap and Dijkstra's search algorithm is applied to generate the Voronoi shortest path. In the Visibility graph generation section, the Voronoi shortest path is processed using Visibility graph. Finally, Dijkstra's search algorithm is applied again to search for the Voronoi-Visibility shortest path in the Visibility graph.

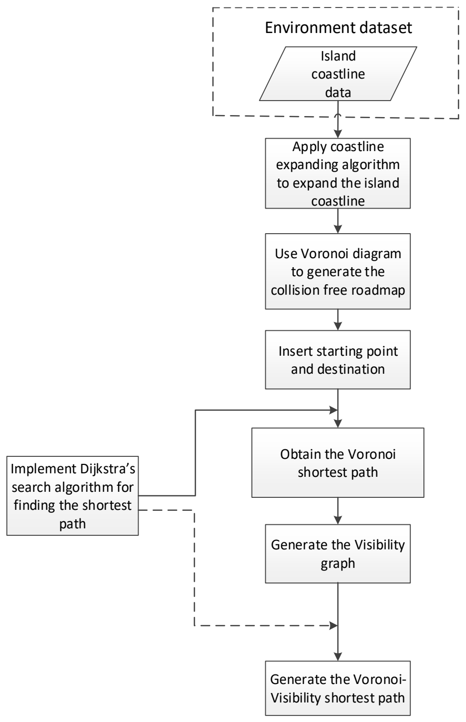

Environment dataset
Island coastline data
Apply coastline expanding algorithm to expand the island coastline
Use Voronoi diagram to generate the collision free roadmap
Insert starting point and destination
Obtain the Voronoi shortest path
Implement Dijkstra's search algorithm for finding the shortest path
Generate the Visibility graph
Generate the Voronoi-Visibility shortest path

Fig. 3. 11. The architecture of the Voronoi-Visibility shortest path planning algorithm

3.5.2 Environmental Dataset

In this research, coastline data from the Global Self-Consistent Hierarchical High-Resolution Shorelines (GSHHS) dataset were used. The high-resolution coastline data used consist of points approximately 200 meters apart. The high-resolution real navigation data not only provide a real simulated environment but also can be used to demonstrate the efficient computing capability of the proposed algorithm.

3.5.3 Voronoi Roadmap Generation with Clearance c

The collision-free Voronoi roadmap generation includes three steps: expanding the coastlines, applying the Voronoi diagram algorithm and removing the unreachable paths.

3.5.3.1 Coastline Expanding Algorithm

To ensure the safety of the USV, each island coastline is expanded by c meters. Note that the parameter c is configurable based on the requirement of the user.

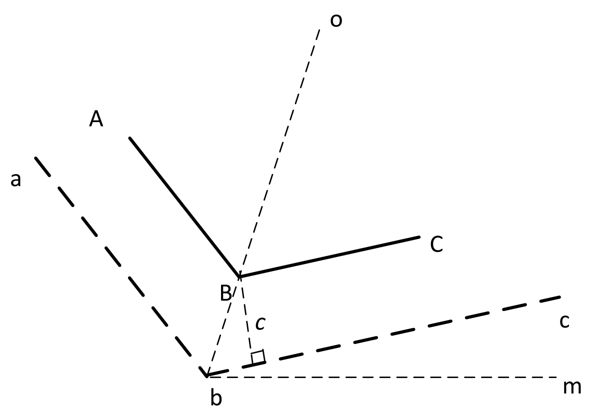


Fig. 3. 12. Coastline expanding algorithm

The coastline expanding algorithm is realized by calculating the position of each expanded coastline endpoint. In Fig. 3. 12, the coastline $A - B - C$ is expanded to $a - b - c$ and the expanding distance is c meters. The segment $A - B$ is parallel to segment $a - b$ and their distance is c meters. The segment $B - C$ is parallel to segment $b - c$ and their distance is also c meters. The positions of points A, B and C are denoted by $A(x_A, y_A)$, $B(x_B, y_B)$ and $C(x_C, y_C)$.

The aim of the coastline expanding algorithm is to calculate the position of $b(x_b, y_b)$. The angle bisector of angle θ_{ABC} is denoted by line *ob*. The Line-Of-Sight (LOS) angle of line bc is denoted by θ_{cbm}. The LOS angle of line ba is denoted by θ_{abm}.

First, the angle $\boldsymbol{\theta_{abm}}$ and $\boldsymbol{\theta_{cbm}}$ are calculated using equation (3. 1) and (3. 2).

$$\theta_{abm} = atan(\frac{y_a - y_b}{x_a - x_b}) \quad \textbf{(3. 1)}$$

$$\theta_{cbm} = atan(\frac{y_c - y_b}{x_c - x_b}) \quad \textbf{(3. 2)}$$

Then, the angle θ_{obc} and the length of Bb are calculated using equation (3. 3) and (3. 4).

$$\theta_{obc} = \frac{\theta_{abm} - \theta_{cbm}}{2} \quad \textbf{(3. 3)}$$

$$|Bb| = \frac{c}{sin(\theta_{obc})} \quad \textbf{(3. 4)}$$

Next, the position of $b(x_b, y_b)$ can be calculated by equation (3. 5) and (3. 6).

$$x_b = x_B - |Bb| \times \cos(\theta_{obc} + \theta_{cbm}) \quad \textbf{(3. 5)}$$

$$y_b = y_B - |Bb| \times \sin(\theta_{obc} + \theta_{cbm}) \quad \textbf{(3. 6)}$$

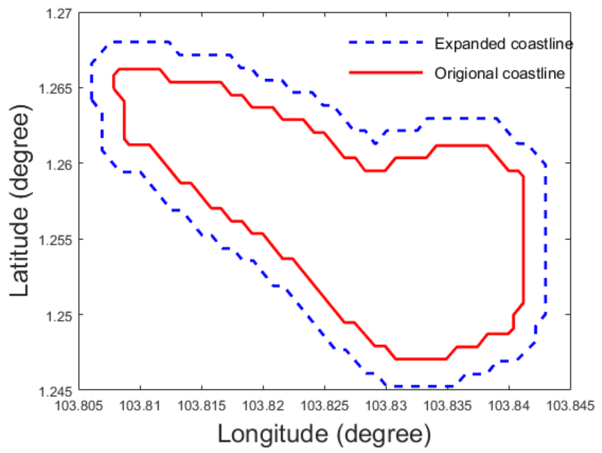


Fig. 3. 13. Illustration of coastline expanding algorithm

Using the coastline expanding algorithm, all the expanded coastline point positions can be calculated in $O(n)$ time. Fig. 3. 13 shows the result of expanding one of the Singapore islands. The red line is plotted using the original data from the GSHHS dataset. The blue line is the expanded island coastline. The expanded distance is 200 meters.

The islands are expanded before implementation in the simulation. After expanding the coastline, all the subsequently generated paths are checked to determine whether they cross the expanded coastlines or not, and only the paths that avoid the expanded coastline are retained. Therefore, the USV can always travel safely with clearance distance *c*.

3.5.3.2 Voronoi Roadmap Generation

The Voronoi roadmap is built in $\mathrm{O}(nlog(n))$ time. For implementation, the Delaunay triangulation should be created first, and then generate the Voronoi diagram from it. The details have been included in the built-in MATLAB function '$voronoin$'; the $voronoin$ function can be expressed in (3. 7).

$$[v, ce] = voronoin(x, y) \tag{3. 7}$$

In (3. 7), the inputs x and y represent the longitude and latitude of all of the coastline points, respectively. The output v is a matrix with size $N \times 2$ and N is the number of the generated Voronoi nodes. The first column of v is the longitude for all the generated Voronoi nodes and second column of v is the latitude for all the generated Voronoi nodes. In (3. 7), the output ce is a vector cell array and stores the Voronoi cells information. Each element of ce contains the indices into v of the vertices of the corresponding Voronoi cell. Each Voronoi cell consists of several Voronoi node indices. If Voronoi node i and node j are in the same cell and they are adjacent, then they are connected. By processing matrix v and ce, we can obtain a node matrix F for the relationships among all Voronoi nodes.

For example, in Fig. 3. 14, an obstacle is represented as a red polygon. The vertices of the polygon and the boundary points are used as the inputs of the $voronoin$ function. By processing the Voronoi nodes matrix and cell data, we can obtain the labelled Voronoi nodes. In Fig. 3. 14, the adjacent nodes are connected using blue segments.

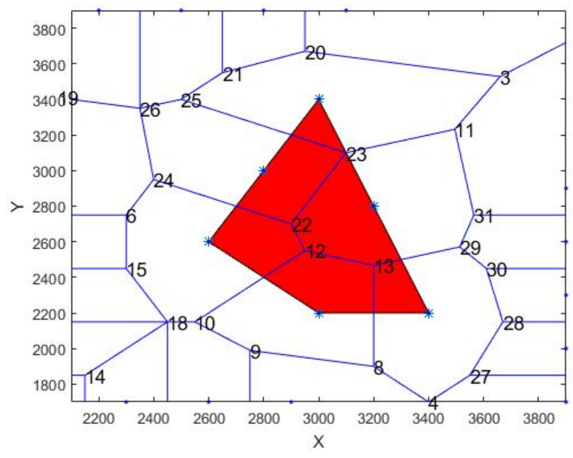


Fig. 3. 14. The Voronoi diagram of a polygon

All the vertices connection information is stored in a $N \times N$ matrix. If node i and node j are found to be connected, then $F(i,j)$ and $F(j,i)$ of matrix F equal 1, otherwise, $F(i,j)$ and $F(j,i)$ equal 0.

However, in Fig. 3. 14, not all the blue lines are reachable for the USV because certain lines are inside the islands or cross the coastlines. Therefore, the unreachable paths should be removed and the node matrix F should also be modified accordingly.

3.5.3.3 Removing Unreachable Path

In Fig. 3. 14, the segments 23-11, 23-25, 23-22, 22-24, 12-22, 12-10, 12-13, 13-8, and 13-29 are not reachable because they cross the obstacle. There are two type of paths required to be removed: (1) if the path segment intersects with the coastlines, it needs to be removed. (2) if one node is inside the island profile, all the paths connected to this node need to be removed.

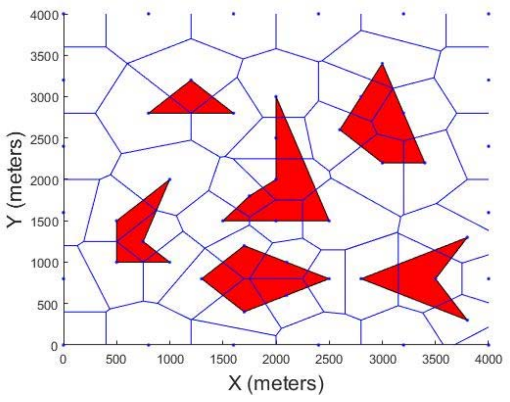


Fig. 3. 15. Voronoi roadmap of polygon obstacles

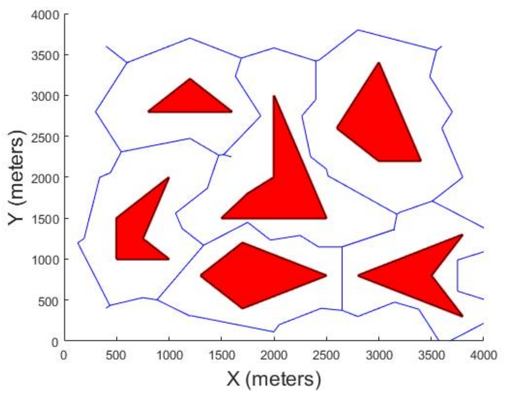


Fig. 3. 16. Collision-free roadmap of polygon obstacles

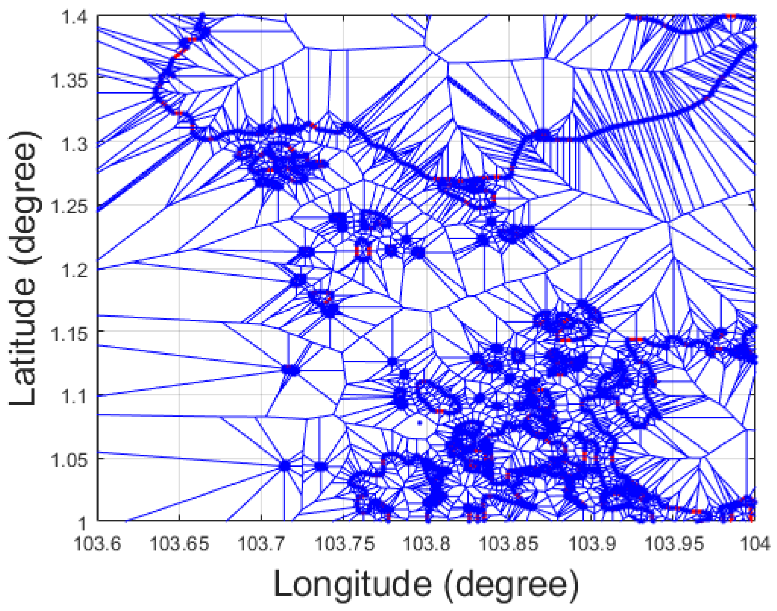


Fig. 3. 17. The Voronoi roadmap of Singapore islands

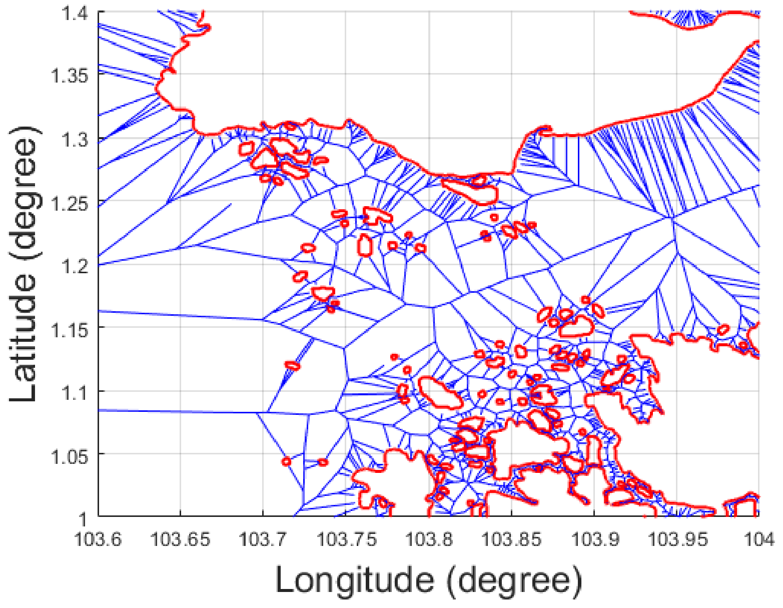


Fig. 3. 18. Collision-free roadmap of Singapore islands

In Fig. 3. 15, the obstacles are represented by red polygons and the paths are represented by the blue segments. After the unreachable paths are removed, we get Fig. 3. 16.

In this section, the high-resolution dataset of the Singapore islands coastlines are also tested. shows the Voronoi roadmap of the Singapore islands, which is represented by the blue lines. The islands are represented by the red lines. Because the dataset is quite large, the roadmap density is very high in Fig. 3. 17. When the unreachable paths are removed, a clear roadmap emerges, which is shown in Fig. 3. 18.

For checking the clearance of one Voronoi edge, the complexity is $O(log(n))$. Because the Voronoi diagram has $O(n)$ edges, removing unreachable path requires $O(nlog(n))$ time. Correspondingly, the node matrix F needs to be modified. Assume a removed path has two endpoints i and j, then the values of $F(i,j)$ and $F(j,i)$ will be updated to 0.

3.5.4 Voronoi Shortest Path Generation

3.5.4.1 Insertion of Starting Point and Destination

Assume there are N nodes generated from the Voronoi diagram, then the positions of the nodes will be queried and the nearest reachable Voronoi nodes to the starting point and the destination point are denoted as node S and node D, respectively. The starting point and the destination point are added to the Voronoi nodes as node $N+1$ and node $N+2$; The node matrix F must be expanded from size $N \times N$ to be size $(N+2) \times (N+2)$, as shown in Table. 3. 1.

Table. 3. 1. The node matrix F with starting point and destination

	$\dots$	S	D	$\dots$	$N+1$	$N+2$
$\dots$	$\dots$	0	0	$\dots$	0	0
S	0	0	0	0	1	0
D	0	0	0	0	0	1
$\dots$	$\dots$	0	0	$\dots$	0	0
$N+1$	0	1	0	0	0	0
$N+2$	0	0	1	0	0	0

3.5.4.2 Dijkstra's Search Algorithm

Now the node matrix F contains the information of the nodes connection and if the two nodes are connected, the corresponding matrix element is 1, otherwise, it equals 0. Then the positions of the node v should be queried and the element of matrix F should be changed from 1 to the length of the two corresponding nodes. The matrix F in Table. 3. 1 will be modified to the matrix shown in Table. 3. 2.

Table. 3. 2. The node matrix F with length information

	$\dots$	S	D	$\dots$	$N+1$	$N+2$
$\dots$	$\dots$	0	0	$\dots$	0	0
S	0	0	0	0	$L_{S,N+1}$	0
D	0	0	0	0	0	$L_{D,N+2}$
$\dots$	$\dots$	0	0	$\dots$	0	0
$N+1$	0	$L_{N+1,S}$	0	0	0	0
$N+2$	0	0	$L_{N+2,D}$	0	0	0

The Dijkstra's search algorithm is then applied to search the shortest path from node $N+1$ to node $N+2$. Now we have stored all the roadmap information into v, ce and F. We stored the positions and indices of the nodes into v, the cell information are stored into ce and the lengths between the connected nodes are stored into F. The pseudocode of the Dijkstra's search algorithm is shown as follows.

```
Function Dijkstra (Roadmap, starting point)
    For each vertex A in Roadmap:      //initialization
            Dist[A] :=infinity                    // Initial distance from starting point to vertex A is
set to infinite
            Previous[A] : = undefined     // Previous node in optimal path from starting point
    Dist[Starting point] :=0                    // Distance from starting point to starting point
    $v$ := The set of all nodes in Roadmap   // All the nodes are stored in $v$
    While $v$ is not empty:                         // main loop
          U := node in $v$ with smallest Dist[]
          Remove U from $v$
          For each neighbour A of U:         // where A has not yet been removed from $v$
                Alt := Dist[U] + Dist_between(U, A)
                    if Alt < Dist[A]
                         Dist[A] := Alt
                         Previous[A] :=U
    Return previous[]
```

Assume the starting point is (103.95, 1.15) and the destination point is (103.65, 1.25), by applying the Dijkstra's search algorithm, the Voronoi shortest path is generated, as shown in Fig. 3. 19. The Voronoi shortest path is shown using a red solid line and the Voronoi roadmap is shown using blue lines. It can be seen that the Voronoi shortest path is far from optimal and it needs to be refined.

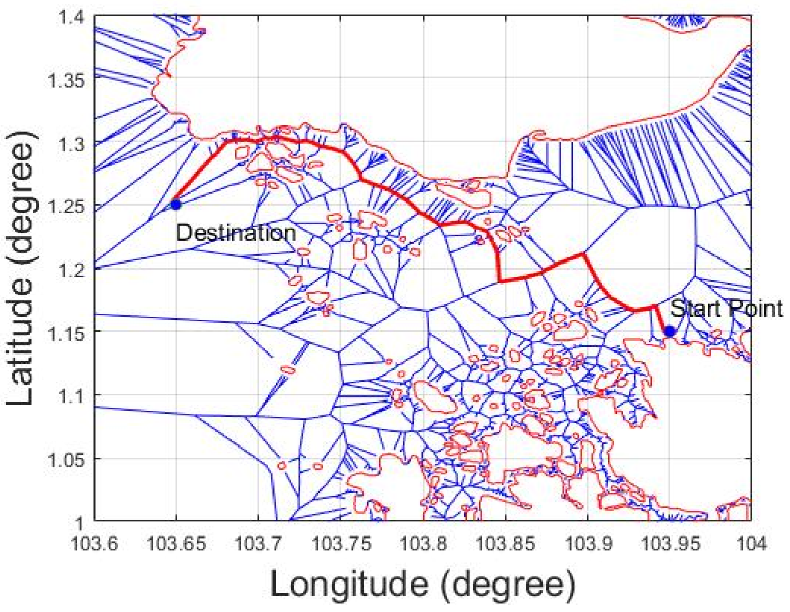


Fig. 3. 19. Voronoi shortest path

3.5.5 Visibility Graph Generation

To refine the Voronoi shortest path, the Visibility graph is applied. Assuming we have a waypoint w_i on the Voronoi shortest path ($i = \{1..m\}$), where m is the number of the waypoints on the Voronoi shortest path, we check whether the line segment $w_i w_1$ has intersection with the expanded coastlines. If there is no intersection, we update the node matrix node $F_{i,1} = L_{i,1}$, otherwise, we check the line segment $w_i w_2$ until the line segment $w_i w_m$. After the first iteration loop, the line segment $w_{i+1} w_j$ will be checked. The pseudocode is shown as below:

```
Function Visibility graph (VP, F)
Assume $VP$ is the Voronoi path, which has $m$ waypoints.
For $i = 1{:}m$
  For $j = 1{:}m$
    Check if line segment $VP(i,j)$ has intersection with the coastlines or not.
    If yes, continue.
    If no, update $F(i,j) = L(i,j)$.
  end
end
```

Visibility graph has $O(m^2)$ edges and constructing the collision-free Visibility graph requires $O(m^2 log(n))$ time. After applying the Visibility graph algorithm, the Voronoi-Visibility roadmap is obtained, Fig. 3. 19 is updated to Fig. 3. 20.

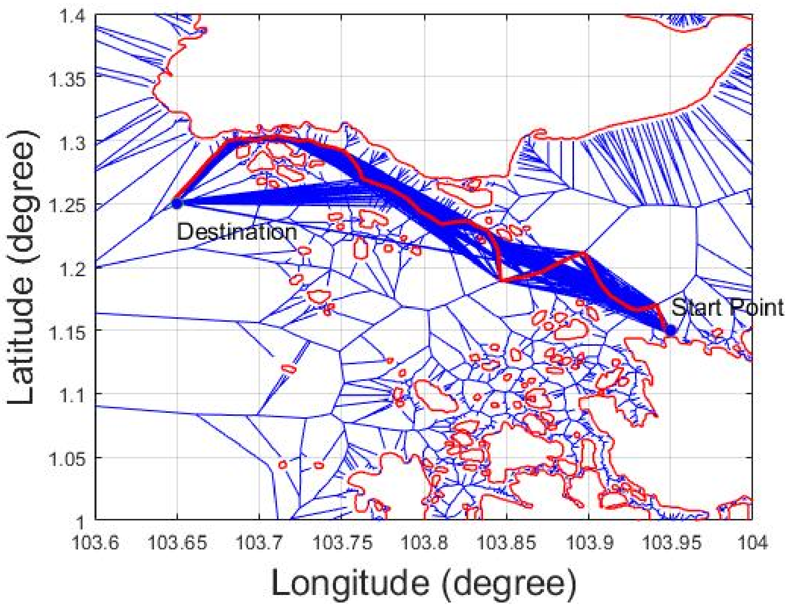


Fig. 3. 20. Voronoi-Visibility roadmap

In Fig. 3. 20, the Voronoi-Visibility roadmap is represented by the blue lines. As can be seen, the Visibility graph generates more possible paths to the Voronoi roadmap.

3.5.6 Voronoi-Visibility Shortest Path Generation

The Dijkstra's search algorithm is applied again to search the Voronoi-Visibility shortest path from the Voronoi-Visibility roadmap. The Voronoi shortest path and the Voronoi-Visibility shortest path are shown in Fig. 3. 21. The Voronoi shortest path is represented by the red line and the Voronoi-Visibility shortest path is shown by the green line. As it can be seen, the green line is smoother and shorter than the red line.

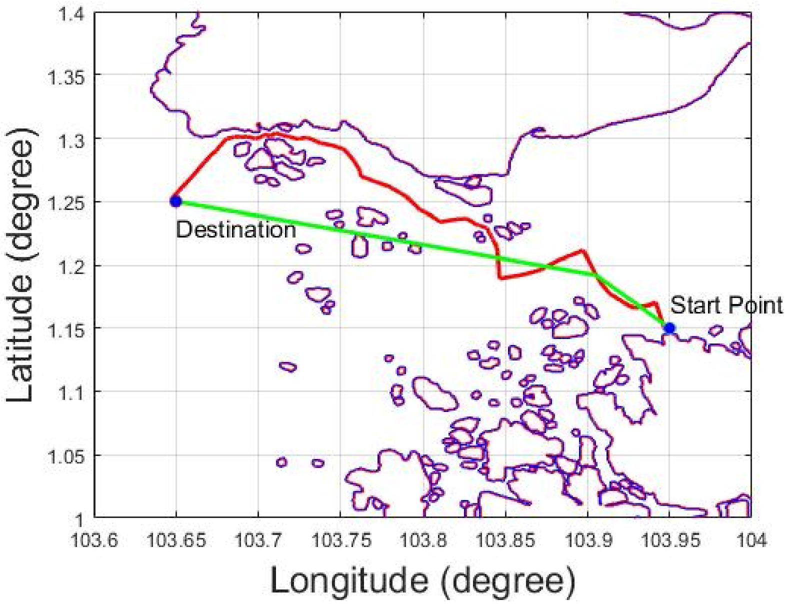


Fig. 3. 21. Voronoi-Visibility shortest path and Voronoi shortest path

The graph is also zoomed in to check if the path keeps the clearance distance from the expanded coastlines, as shown in Fig. 3. 22 and Fig. 3. 23. In Fig. 3. 22, the selected area to be zoomed is in the blue rectangle. In Fig. 3. 23, the original islands are represented by the red lines and the expanded coastlines are represented by the blue lines. The green Voronoi-Visibility shortest path keeps the clearance distance with the expanded coastlines, which means the path generated by the proposed algorithm can keep the USV safe.

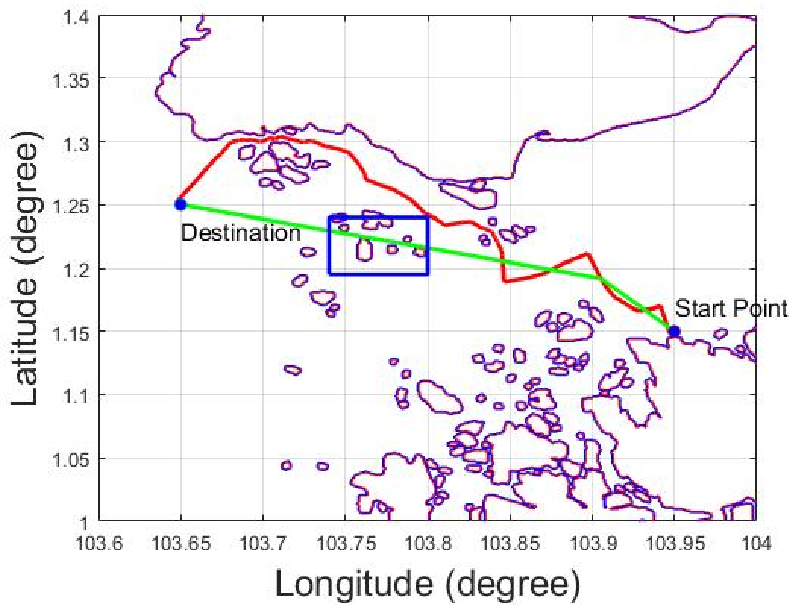


Fig. 3. 22. Voronoi-Visibility shortest path

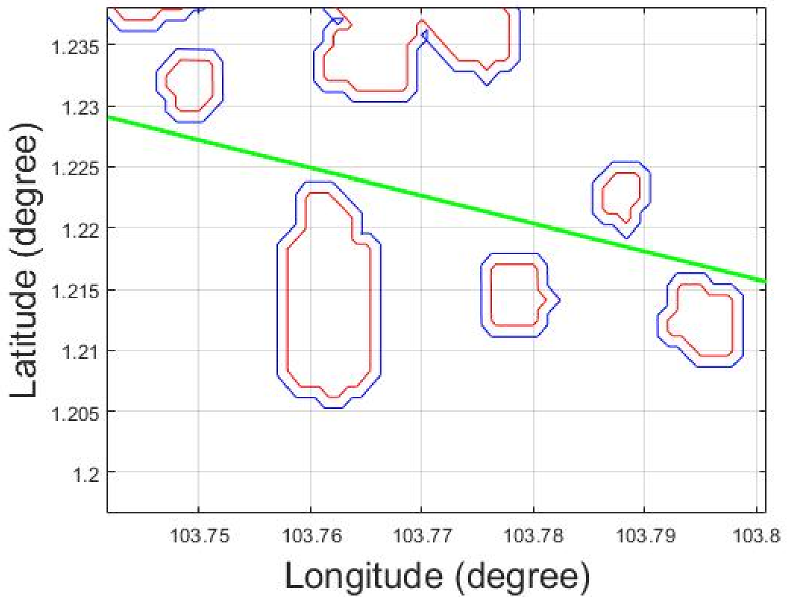


Fig. 3. 23. Voronoi-Visibility shortest path (Zooming in)

3.6 Numerical Simulation

3.6.1 Voronoi Diagram Based Shortest Path Planning Algorithm for Comparison

For comparison, the VM (Voronoi path planning algorithm refined by minimizing the number of waypoints) path planning algorithm is applied (Bhattacharya & Gavrilova, 2008). Similar to the proposed algorithm, the coastlines were first expanded and then the starting point and destination point were inserted into the roadmap. The Dijkstra's search algorithm was applied to search the shortest path between the starting point and the destination point. Once the Voronoi shortest path was generated, the path was refined by minimizing the number of the waypoints on the Voronoi path. The architecture of this algorithm is shown in Fig. 3. 24.

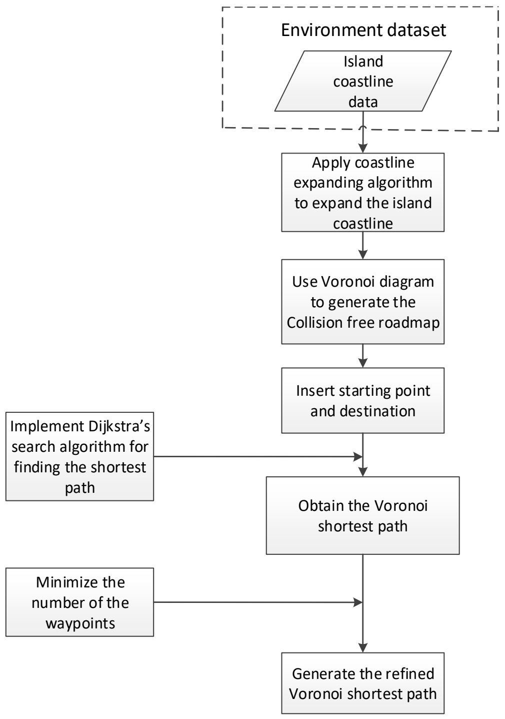

Environment dataset
Island coastline data
Apply coastline expanding algorithm to expand the island coastline
Use Voronoi diagram to generate the Collision free roadmap
Insert starting point and destination
Implement Dijkstra's search algorithm for finding the shortest path
Obtain the Voronoi shortest path
Minimize the number of the waypoints
Generate the refined Voronoi shortest path

Fig. 3. 24. The architecture of the VM path planning algorithm for comparison

The detail of minimizing the number of the waypoints is given as follows: for a waypoint w_i on the Voronoi shortest path ($i = \{1..m-2\}$), where m is the number of the waypoints on the Voronoi shortest path, we check whether the line segment $w_i w_{i+2}$ has intersection with the expanded coastlines. If no, we remove w_{i+1} from shortest path and repeat the process from waypoint w_{i+2}. If yes, we retain w_{i+1} and consider it as the next waypoint for processing.

The Voronoi shortest path, the VM shortest path and the VV (Voronoi-Visibility shortest path) shortest path are shown in Fig. 3. 25.

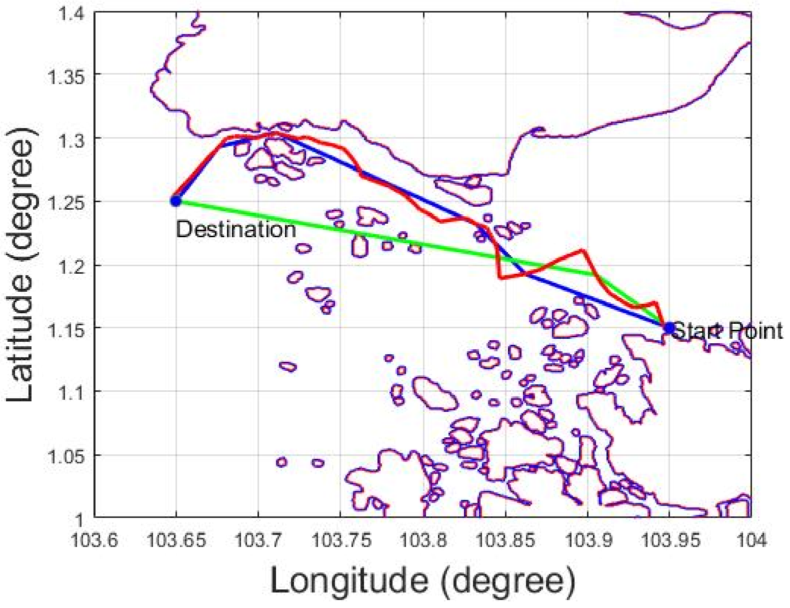


Fig. 3. 25. Comparison of Voronoi shortest path, VM shortest path and VV shortest path

The red line represents the Voronoi shortest path. The blue line represents the VM path and the green line represents the VV shortest path. It is shown that the VV shortest path save $13.89\ \%$ of the total length of the VM path.

3.6.2 Ten USV Mission Scenarios in Singapore

All the simulations were carried out on a 2.7 GHz Intel Core i7-6820HK processor with 16.0 GB. The programs were implemented using Matlab R2016b. In Section 3.6.2, the proposed VV algorithm was compared with VM algorithm for ten mission scenarios in the Singapore Strait. The lengths of the paths and the computational time were compared.

3.6.2.1 Path Length Comparison

The comparison of the path lengths is shown in Table. 3. 3. In these ten missions, the clearance distance between the USV and the coastline was set to be 100 meters. The missions have different starting and destination end points. The path length saved varies from 4.23% to 13.89%. The Voronoi shortest path, the VM shortest path and the VV shortest path of Mission No.1 and No.10 are shown in Fig. 3. 26 and Fig. 3. 27, respectively.

Table. 3. 3. The comparison of VV algorithm and VM algorithm in ten Singapore missions

Mission No.	Starting point	Destination	VM length(km)	VV length(km)	Saved length
1	(103.95,1.2)	(103.75,1.08)	29.121	27.888	4.23%
2	(103.95,1.25)	(103.78,1.08)	29.223	27.425	6.15%
3	(103.95,1.15)	(103.65,1.25)	41.711	35.919	13.89%
4	(103.68,1.3)	(103.95,1.2)	36.317	32.497	10.52%
5	(103.74,1.3)	(103.95,1.2)	29.880	26.081	12.71%
6	(103.95,1.2)	(103.65,1.05)	40.311	38.576	4.31%
7	(103.7,1.25)	(103.98,1.2)	37.013	33.942	8.30%
8	(103.65,1.27)	(103.98,1.30)	40.708	38.111	6.38%
9	(103.95,1.2)	(103.65,1.25)	38.798	34.274	11.66%
10	(103.95,1.3)	(103.65,1.25)	40.067	34.873	12.96%

In Fig. 3. 26 and Fig. 3. 27, the red lines represent the Voronoi shortest paths, the blue lines represent the VM shortest paths and the VV shortest paths are represented by the green lines. As can be seen, the Voronoi shortest paths are far from optimal, although the VM path planning algorithm refined the Voronoi shortest path, its performance is not as good as the proposed VV shortest path planning algorithm. Based on the comparison of the ten Singapore missions, we can conclude that the Visibility graph provides more available paths to the Voronoi roadmap, after applying the Dijkstra's search algorithm, the generated Voronoi-Visibility shortest path has a shorter length than the Voronoi path that is refined by minimizing the number of the waypoints.

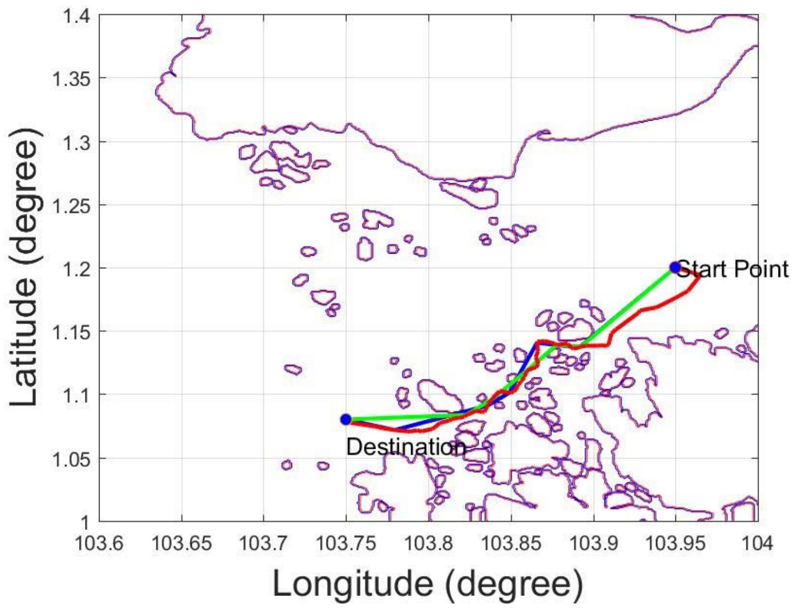


Fig. 3. 26. The comparison of Voronoi shortest path, VM shortest path and VV shortest path in Mission No.1

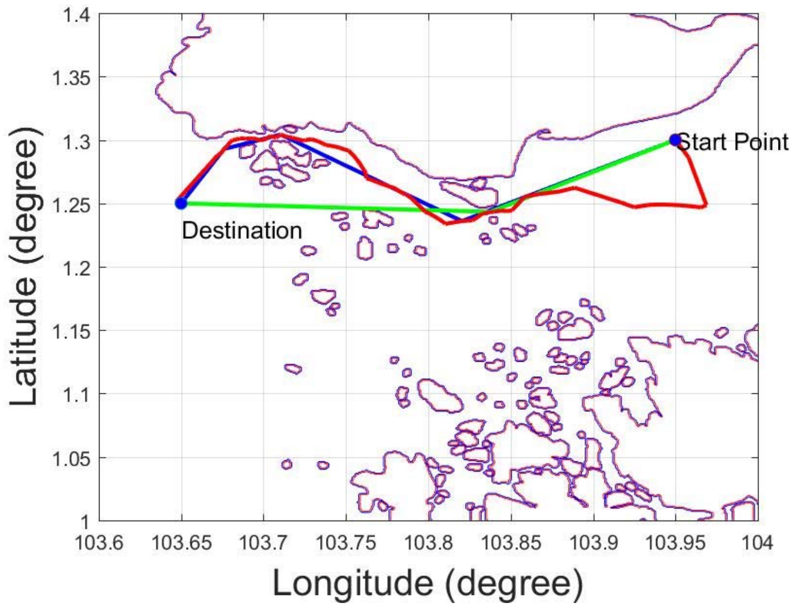


Fig. 3. 27. The comparison of Voronoi shortest path, VM shortest path and VV shortest path in Mission No.10

3.6.2.2 Computational Time Comparison

The computational time of Voronoi algorithm, VM algorithm and VV algorithm in these ten USV mission path generation are also compared, as shown in Table. 3. 4. As it can be seen, the Voronoi algorithm needed the least computational time; The VM approach cost more time than Voronoi algorithm, which is less than 0.1 second. The VV algorithm needed the longest time, however, when we compared the computational time between VV and VM, it was found that the extra computational time is less than 2.2 seconds. The extra computational time is proportional to the number of the waypoints on the Voronoi path. Because the complexity of fastest way for building the Visibility graph is $O(n^2)$, instead of applying the Visibility graph to the entire map, whose vertices number is 4128, we apply the Visibility graph to the Voronoi shortest path, whose vertices number is between 52 and 110. Note that the codes implemented in this research do not include any optimization technique to accelerate the computation. Moreover, according to the simulations of (Bhattacharya & Gavrilova, 2008), when applying the Visibility graph to the map with 1866 vertices, the computational time is longer than 60 seconds, however, using our proposed algorithm, it was found that the proposed Voronoi-Visibility (VV) algorithm just consumed 2.152 seconds more than the VM algorithm in No.3 mission and hence can conclude that the proposed Voronoi-Visibility algorithm still keeps the same level of computational efficiency as that of the Voronoi path planning algorithm.

Table. 3. 4. Computational time comparison of Voronoi algorithm, VM algorithm and VV algorithm

Mission No.	Voronoi shortest path planning computational time t_V (seconds)	VM shortest path planning computational time t_{VM} (seconds)	VV shortest path planning computational time t_{VV} (seconds)	Number of waypoints on the Voronoi path	$t_{VV} - t_{VM}$ (seconds)
1	8.885	8.903	10.381	81	1.479
2	8.471	8.479	9.397	52	0.918
3	8.489	8.506	10.657	109	2.152
4	8.608	8.622	10.561	101	1.940
5	8.644	8.654	9.841	67	1.187
6	8.483	8.507	10.098	89	1.591
7	8.598	8.618	9.857	71	1.239
8	8.594	8.677	10.804	110	2.127
9	8.587	8.604	10.707	107	2.102
10	8.652	8.667	10.757	106	2.089

3.6.3 Five USV Mission Scenarios in Croatia

To test the performance of the proposed algorithm in different spatial scenario, the proposed algorithm was also compared with the VM algorithm in five USV mission scenarios of the coast of Croatia, as shown in Table. 3. 5. The saving distance is between 3.06% and 8.60%. The clearance distance again was set to be 100 meters. The results of the two algorithms for Mission No.4 are shown in Fig. 3. 28. The green line represents the VV shortest path and the black line represents the VM shortest path. As can be seen, in the Croatia scenario, the proposed VV algorithm can still save travelling distance when compared with the VM algorithm, which demonstrated the flexibility of the proposed algorithm in dealing with different geographical scenario.

Table. 3. 5. The comparison of VV algorithm and VM algorithm in five Croatia missions

Mission No.	Starting point	Destination	VM length(km)	VV length(km)	Saved length
1	(14.45, 45.20)	(14.50, 44.10)	144.006	136.229	5.40%
2	(14.50, 44.80)	(14.50, 44.10)	96.111	89.162	7.23%
3	(14.60, 44.90)	(14.50, 44.10)	100.190	97.121	3.06%
4	(14.45, 45.20)	(14.50, 44.30)	119.321	109.054	8.60%
5	(14.45, 45.20)	(14.60, 44.20)	128.330	120.553	6.06%

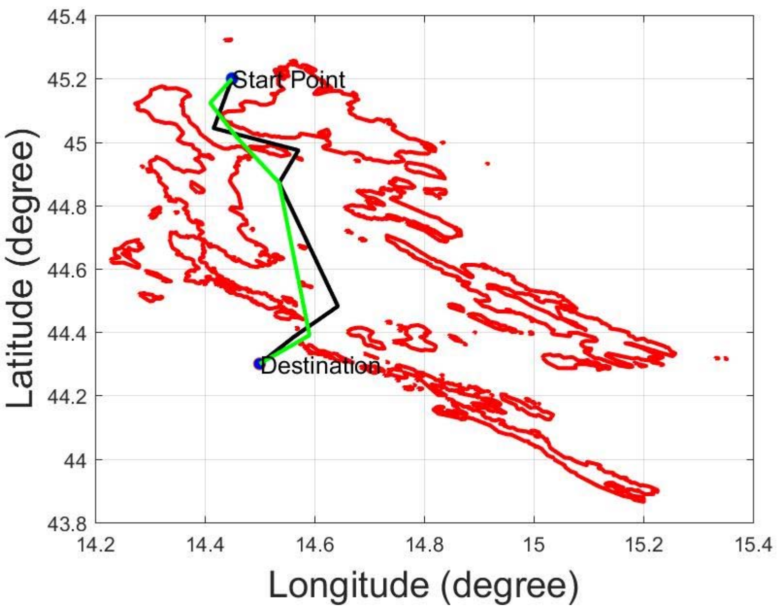


Fig. 3. 28. The VV shortest path and VM shortest path in Mission No.4

3.7 Summary

In this chapter, the Voronoi diagram, Visibility graph and Dijkstra's search algorithm are integrated to solve the problem of USV shortest path planning algorithm. First, an algorithm was developed to expand the coastline, hence ensuring the safety of the USV by keeping a configurable clearance distance from the coastlines. The Voronoi diagram was then applied to generate the Voronoi collision-free roadmap. Dijkstra's search algorithm was implemented for searching the shortest path from the Voronoi roadmap, and Visibility graph was also implemented for generating more candidate paths. Dijkstra's search algorithm was applied again to search for the shortest VV path. Comparing the path length of the proposed VV algorithm and that of the VM algorithm in the Singapore Strait missions and the Croatian islands missions, it can be concluded that the proposed algorithm generates shorter paths than the VM algorithm and it can be applied in multiple geographical scenarios. As for the computational time comparison of the proposed VV algorithm, Voronoi shortest path planning algorithm and VM algorithm, it can be concluded that the proposed VV algorithm still keeps the computational efficiency of the Voronoi diagram approach. Hence the proposed Voronoi-Visibility shortest path planning algorithm generates shorter path than the Voronoi shortest path planning algorithm and VM algorithm, and still keeps the same level of computational efficiency as that of the Voronoi shortest path planning algorithm.

4 Energy Efficient Path Planning Algorithm in Spatially Variant Environment

4.1 Abstract

The sea current state affects the energy consumption of USVs significantly and the path planning approach plays an important role in determining how long the unmanned surface vehicle (USV) can travel. However, the research is somehow limited on USVs path planning approaches based on roadmap method that incorporates the energy consumption of USVs. The limited energy capacity of USVs is the limiting factor affecting the range and duration of military and civil ocean applications. To improve the USVs endurance, an energy efficient path planning approach for computing feasible paths for USVs that takes the energy consumption into account based on sea current data is proposed. In the proposed approach, Voronoi diagram, Visibility graph, Dijkstra's search and energy consumption function are combined, which allows USVs to avoid obstacles while at the same time using the minimum amount of energy. The approach also ensures that the USV remains at a user-configurable safety distance away from all islands and coastlines, hence maintaining the safety of USVs due to the uncertainties or inaccuracies in the map data. The Voronoi-Visibility (VV) energy-efficient path and the corresponding shortest path were simulated and compared in ten missions in Singapore Strait and five missions in the coast of Croatia. The simulation results also provided an insight into the amount of energy that could potentially be saved by taking the sea current into account when planning the USV path and not only relying on the shortest path information. The effects of the mission time, the USV speed and the sea current state on the results were analyzed. The computational time of the VV algorithm and the Voronoi energy-efficient path planning algorithm were also compared, it is demonstrated that the proposed VV algorithm improves the quality of the Voronoi energy-efficient path but not reduces the computational efficiency of the Voronoi energy-efficient path planning algorithm.

4.2 Introduction

In the sense of saving energy, the effect of the sea current in the path planning is presented in (Garau, et al., 2005) and an A* search algorithm with time optimal cost is proposed. A* is also applied to find a path for an AUV to navigate in the presence of currents (Koay & Chitre, 2013) and obstacles while consuming the minimum amount of energy. However, the final paths are not optimal and have redundant turns. An energy efficient path planning algorithm based on EEA* was proposed in (Lee, et al., 2015). The proposed algorithm used a realistic energy cost considering the tidal current and water-depth. The proposed algorithm was finally compared with a classical distance based A* algorithm. In this approach, the collision-free space was constructed using cell decomposition approach. The cell decomposition method includes exact decomposition method and approximate decomposition method. The cell decomposition method, although simple to implement, seldom yields high-quality paths (Bhattacharya & Gavrilova, 2008). The decomposition method discretizes the free space recursively, stopping when a cell is entirely in free-space or entirely inside an obstacle. The exact cell decomposition techniques are faster than the approximate one, but the path is not optimal. The decomposition method can yield near-optimal paths by increasing the grid resolution, but will also increase the computational burden.

The level set method (Agarwal & Lermusiaux, 2011) is used to solve the environmental influence problem for AUV path planning. The Anisotropic Fast Marching (AFM) method (Petres, et al., 2005) is applied to address similar problems but in an environment where relatively stronger currents exist. The AFM is an improved version of the FM method with higher computational efficiency than the level set method (Agarwal & Lermusiaux, 2011). Also, the optimal collision-free path generated by the AFM is able to provide the guaranteed convergence, which has been intuitively explained in (Konukoglu, et al., 2007) and mathematically proven in (Mirebeau, 2014). However, these studies have only been applied on AUV platforms, which has limited constraints than the USV environment. In the USV environment, additional constraints such as wind, tidal currents and COLREGS regulations, should be considered, for which the conventional AFM cannot implement. A multi-layered fast marching (MFM) method for USV path planning was proposed in (Song, et al., 2017) to address these problems. The MFM method took account of the sea current data and generated path in the collision-free space, which was constructed by potential field method. The potential method used an attractive/repulsive vector field to construct the collision-free space around the obstacles. However, the potential method may generate local minima.

The cell decomposition method or potential field method have been applied to yield the collision-free space in dealing with USV energy-efficient path planning problem, little work about roadmap based energy efficient path planning method has been done. The roadmap method attempts to capture the free-space connectivity with a graph. The roadmap method includes probabilistic roadmap method (Amato & Wu, 1996), rapidly-exploring random tree (Kuffner & Latombe, 2000), expansive space planner (Hsu, et al., 1997). Some other roadmap –based approaches are based on computational geometry, including Visibility graph and Voronoi diagram. Visibility graph has been used for yielding shortest path and Voronoi diagram is implemented for a maximum clearance path. The Voronoi diagram can be constructed in $O(nlog(n))$ time and generate a path with maximum clearance, where n is the number of the islands' vertices. However, the Voronoi path may be far from optimal and it usually has redundant turns. The Visibility graph can generate better path than then Voronoi diagram, however, it will consume $O(n^2)$ time to construct the collision-free roadmap, which makes it impractical for large spatial dataset.

In this research, a Voronoi-Visibility energy-efficient path planning algorithm is proposed. The proposed algorithm integrates the computational efficiency advantage of the Voronoi diagram and the optimal advantage of the Visibility graph. It improves the quality of the Voronoi path and keeps the same level of the computational efficiency as that of the Voronoi diagram method. The proposed algorithm not only minimize the energy consumption but also ensures the safety of the USV by keeping the USV a configurable clearance distance from the coastlines. The rest of this chapter is organized as follows: Section 4.3 presents the problem statement of developing the energy efficient path planning algorithm. The methodology is introduced in Section 4.4. The Voronoi-Visibility shortest path planning algorithm and the proposed energy efficient path planning algorithm are compared in Section 4.5. The summary is given in Section 4.6.

4.3 Problem Statement

The USV energy-efficient path planning algorithm for long range missions includes three challenges: Firstly, processing the large spatial dataset, it should be computationally efficient. Secondly, the generated path should keep the USV a minimum clearance distance from the island coastline, in order to ensure the safety of the USV due to map inaccuracy. Thirdly, the path planning approach should take into account the spatially variant sea current data and generate an energy-efficient path. The computational efficiency and clearance distance problems have been introduced in Section 3 and the energy saving problem will be introduced in this section.

The general path planning approach finds the shortest path while avoiding obstacles whereas sea currents have a direct effect on the energy cost of the USV. The USV usually has limited energy and saving energy will improve the endurance of the mission execution. Downstream current will reduce energy consumption, however, it may sacrifice the total distance. In some circumstances, it is also necessary to cross unflavoured surface current. Therefore, it is necessary to develop an energy efficient path planning approach to take account of the sea current data.

4.4 Methodology

This section is organized as follows: Section 4.4.1 describes the architecture of the proposed algorithm. Section 4.4.2 presents the environmental dataset used in this research, including the large spatial dataset and the sea current data. The Voronoi collision-free roadmap generation is described in Section 4.4.3 and the

Voronoi energy-efficient path generation is presented in Section 4.4.4. Visibility graph generation approach is described in Section 4.4.5. Finally, the Voronoi-Visibility energy-efficient path generation is introduced in Section 4.4.6.

4.4.1 Algorithm Architecture

The architecture of the proposed Voronoi-Visibility energy-efficient path planning algorithm is shown in Fig. 4. 1. This algorithm includes four parts: the collision-free Voronoi roadmap generation, the Voronoi energy-efficient path generation, the Visibility graph generation and Voronoi-Visibility energy efficient path generation.

In the collision-free Voronoi roadmap generation, the coastline dataset is processed by the coastline expanding algorithm and the expanded coastlines will be used for keeping the subsequently generated roadmap a clearance distance *c* from the original coastlines. Then the expanded coastline is processed by the Voronoi diagram algorithm, the unreachable edges are removed and finally the collision-free Voronoi roadmap is generated. In the Voronoi energy-efficient path generation section, the starting point and the destination are inserted into the Voronoi roadmap. All the nodes of the Voronoi roadmap are stored in a node matrix. Then the energy consumption model is used to predict the energy cost of each edge of the Voronoi roadmap, and the corresponding energy cost weight is stored in the node matrix. The Dijkstra’s search algorithm is applied to generate the Voronoi energy-efficient path. In the Visibility graph generation section, the Voronoi energy-efficient path is processed using Visibility graph and Visibility graph will provide more available paths into the Voronoi roadmap. Finally, the Dijkstra’s search algorithm is applied again to search for the Voronoi-Visibility energy-efficient path.

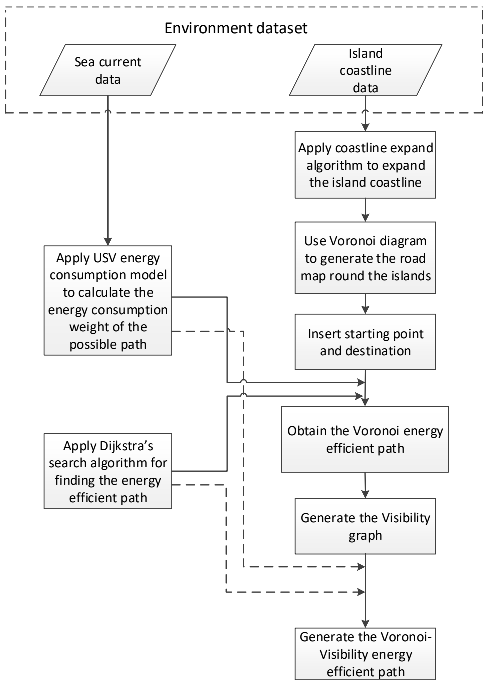

Environment dataset
Sea current data
Island coastline data
Apply coastline expand algorithm to expand the island coastline
Use Voronoi diagram to generate the road map round the islands
Insert starting point and destination
Apply USV energy consumption model to calculate the energy consumption weight of the possible path
Apply Dijkstra's search algorithm for finding the energy efficient path
Obtain the Voronoi energy efficient path
Generate the Visibility graph
Generate the Voronoi-Visibility energy efficient path

Fig. 4. 1. The architecture of the Voronoi-Visibility energy-efficient path planning algorithm

4.4.2 Environmental Dataset

The data used in this research include island coastlines data and sea current data.

4.4.2.1 Coastline Data

The Global Self-Consistent Hierarchical High-Resolution Shorelines (GSHHS) dataset is applied. The GSHHS dataset was created by Paul Wessel of the University of Hawaii and Walter H. F. Smith of the NOAA Geosciences Lab. The high-resolution coastline data used consist of points approximately 200 meters apart. The high-resolution data will provide a realistic environment for simulation.

4.4.2.2 Sea Current Data

The development of ocean science and satellite image processing techniques enable the ocean current state to be predicted. The sea current data was obtained from the company TideTech Ltd (Tidetech, 2014). The sea current data are compiled in grib files. The grib file format is a concise data format used to store historical and forecast weather data. The resolution, time step and forecast length of the sea current data are provided in Table. 4. 1. The combination of the sea current data and the coastline data in the Singapore Strait is shown in Fig. 4. 2.

Table. 4. 1. Sea current data specification

Region	Parameters	Resolution	Updating Time Step	Forecast Length
North Atlantic	Current	11 km	24 hrs	144 hrs
Gulf Stream	Current	11 km	24 hrs	144 hrs
English Channel	Current	2 km	15 min	48 hrs
North West Europe	Current	20 km	60 min	120 hrs
Singapore Strait	Tide Current	1.1 km	60min	48 hrs

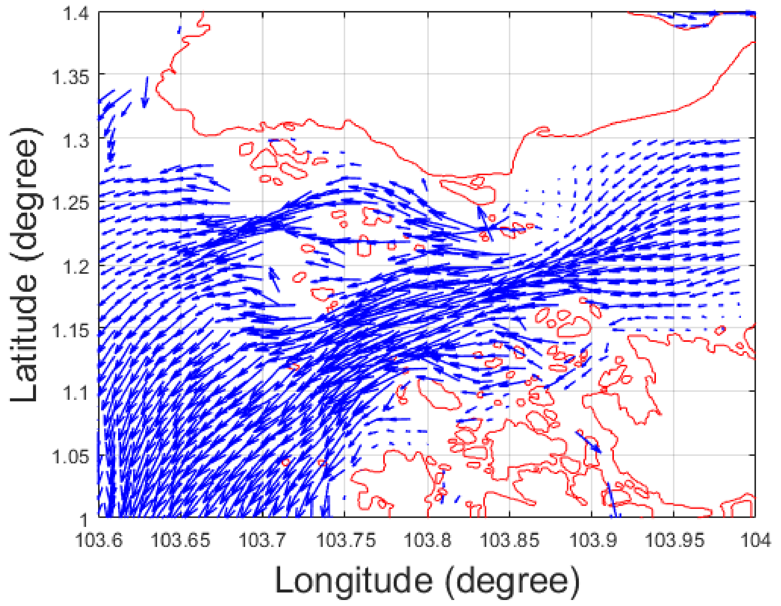


Fig. 4. 2. Sea currents of the Singapore Strait: the current state was recorded at 5:00 am, 11/06/2014. The red lines represent the coastlines of Singapore islands and the blue arrows show the sea current state.

4.4.3 Voronoi Roadmap Generation with Clearance c

The collision-free Voronoi roadmap generation includes three steps: expanding the coastlines, implementing the Voronoi diagram algorithm and removing unreachable paths, which are introduced in Section 3.5.3.

4.4.4 Voronoi Energy Efficient Path Generation

The Voronoi energy-efficient path generation includes three steps: Firstly, inserting the starting point and the destination point into the Voronoi roadmap; Secondly, applying the energy consumption model to calculate the path segments' weight; Thirdly, implementing Dijkstra's search algorithm to search the Voronoi energy-efficient path from the starting point to the destination.

4.4.4.1 Insertion of Starting Point and Destination

We insert the starting point and the destination point into the Voronoi roadmap by connecting the starting point and the destination point to the corresponding nearest Voronoi nodes. By querying the node positions, we can calculate the nearest Voronoi nodes to the starting point and the destination point. The nearest node to the starting point is denoted by S and the nearest node to the destination node is denoted by D. We assume there are N Voronoi nodes, then the node matrix F should be modified from a $N \times N$ matrix to a $(N+2) \times (N+2)$ matrix. We denote the starting node as $N+1$ node and the destination point as $N+2$ node, then the corresponding elements of F should be modified:$F(N+1, S) = 1, F(S, N+1) = 1, F(N+2, D) = 1, F(D, N+2) = 1$.

4.4.4.2 Energy Consumption Model

Assume a USV that is commanded to travel from waypoint N_i to waypoint N_{i+1}, the ground speed of the USV is denoted by $\overrightarrow{v_g}$, the sea current speed is denoted by $\overrightarrow{v_c}$, and the relative USV speed is denoted by $\overrightarrow{v_u}$. $\overrightarrow{v_g}$, $\overrightarrow{v_c}$ and $\overrightarrow{v_u}$ satisfy equation (4. 1).

$$\overrightarrow{v_g} = \overrightarrow{v_c} + \overrightarrow{v_u} \tag{4. 1}$$

The ship resistance is the most important term in determining the effective power of the given ship. Assuming the USV does not have the thrusters of the sway motion, only the surge speed relative to the water will be considered and the hydrodynamic drag F_d can be calculated by the equation (4. 2).

$$F_d = \frac{1}{2}\rho|v_u|^2 C_D A \tag{4. 2}$$

Where ρ is the mass density of the water, C_D is the drag coefficient, and A is the reference area. The USV energy consumption weight E can be calculated from the product of USV speed through water, the drag force it experiences, and the travel duration. Therefore, we get equation (4. 3) and (4. 4).

$$E = F_d \times |v_u| \times t \tag{4. 3}$$

$$t = \frac{|\overline{N_i N_{i+1}}|}{|v_g|} \tag{4. 4}$$

Where t is the time the USV consumed to travel from waypoint N_i to waypoint N_{i+1}. Then we can get equation (4. 5).

$$E = \alpha |v_u|^3 \frac{|\overline{N_i N_{i+1}}|}{|v_g|} \tag{4. 5}$$

Where α is a combination of the water density, drag coefficient and the reference area, as α is a constant value for each boat and we only analyse the proportion of the saved energy using the proposed algorithm comparing with the shortest algorithm, for simplicity, we assume α equals 1 in this research. Therefore, if the USV is commanded to travel at a constant ground speed $\overrightarrow{v_g}$, only the USV relative speed $\overrightarrow{v_u}$ and the waypoint segment length needs to be calculated. Note that if the distance between N_i and $N_{i+1}$is longer than the scale of the sea current, it will be divided into smaller segments before calculating the energy cost.

For the available Voronoi edges whose corresponding element in node matrix F equals 1, the energy consumption model is used to calculate the corresponding energy cost and the corresponding value of node matrix F will be updated.

4.4.4.3 Dijkstra's Search

We stored the positions and indices of the nodes into v, the cell information is stored into ce and the energy consumption information between the connected nodes is stored into F. The Dijkstra search algorithm is used for searching the most energy efficient path in the Voronoi roadmap. The pseudocode of the Dijkstra's search algorithm is given as follows:

```
Function Dijkstra (Roadmap, start point)
    For each vertex A in Roadmap:               //initialization
        Energycost[A] :=infinity                // Initial energy cost from start point to vertex A is set to infinite
        Previous[A] : = undefined               // Previous node in optimal path from start point
    Energycost[Start point] :=0                 // Energy cost from start point to start point
    $v$ := The set of all nodes in Roadmap      // All the nodes are stored in $v$
    While $v$ is not empty:                     // main loop
        U := node in $v$ with smallest Energycost[]
        Remove U from $v$
        For each neighbour A of U:              // where A has not yet been removed from $v$
            Alt := Energycost[U] + Energycost_between(U, A)
            if Alt < Energycost[A]
                Energycost[A] := Alt
                Previous[A] :=U
    Return previous[]
```

After implementing Dijkstra's search algorithm, the Voronoi energy-efficient path can be generated. The USV is commanded to travel from (103.95, 1.2) to (103.68, 1.3) at 05:00 am on 11/06/2014. The USV keeps a constant speed 1m/s. After the energy consumption model is implemented to calculate the corresponding energy consumption weight of node matrix F, Dijkstra's search algorithm is implemented to search the Voronoi energy efficient path, as shown in Fig. 4. 3. The green line represents the Voronoi energy efficient path and the blue lines represent the Voronoi collision-free roadmap. If we calculate the length of the connected nodes and implement Dijkstra's search algorithm, we can get the Voronoi shortest path. For comparison, the Voronoi shortest path is shown in Fig. 4. 4. The black line represents the Voronoi shortest path.

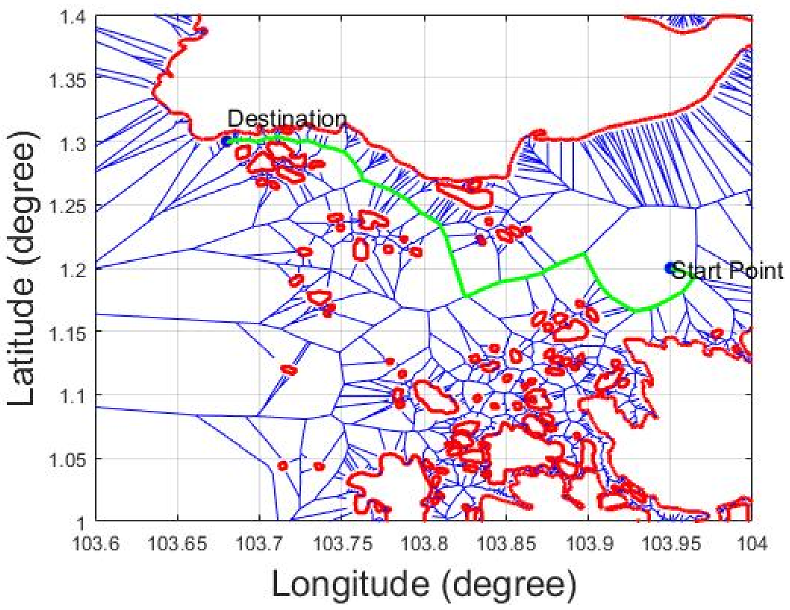


Fig. 4. 3. Voronoi energy efficient path

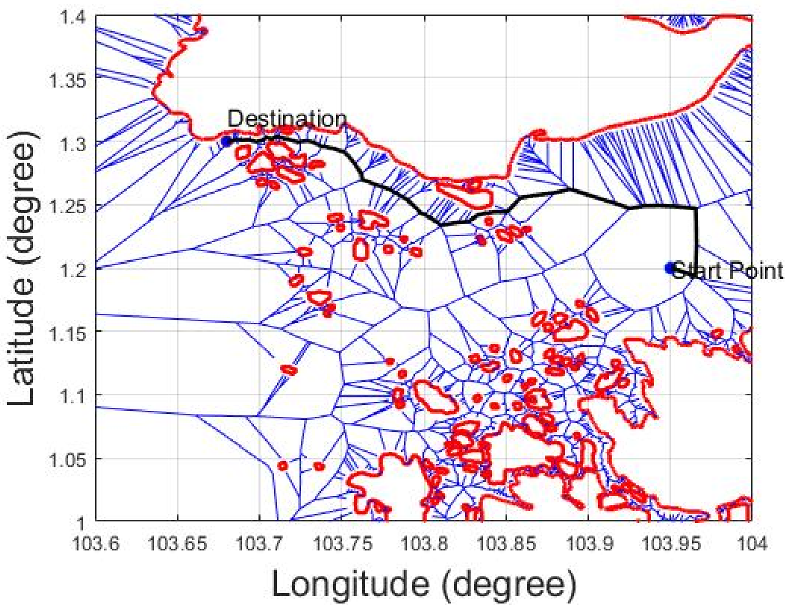


Fig. 4. 4. Voronoi shortest path

The Voronoi energy-efficient path and shortest path are combined in Fig. 4. 5. The sea current state is represented using blue arrows. The Voronoi energy-efficient path is represented by green line and the Voronoi shortest path is represented by black line. It can be seen that using the Voronoi energy-efficient path will make the USV encounter more favorable sea current than the shortest path. However, it can be seen that the Voronoi paths are far from optimal. Therefore, Visibility graph algorithm is implemented to optimize the Voronoi path.

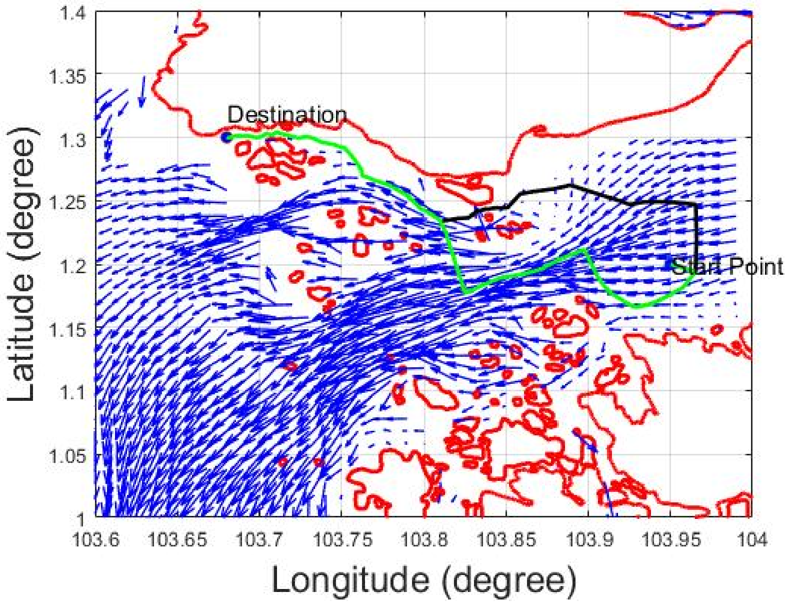


Fig. 4. 5. Voronoi energy-efficient path vs shortest path

4.4.5 Visibility Graph Generation

To refine the Voronoi energy-efficient path, the Visibility graph algorithm is applied to provide more candidate paths to the Voronoi roadmap. Assume for a waypoint w_i on the Voronoi energy efficient path ($i = \{1..m\}$), where m is the number of the waypoints on the Voronoi path, we check whether the line segment $w_i w_1$ has intersection with the expanded coastlines. If there is no intersection, using the energy consumption model, we update the node matrix node $F_{i,1} = E_{i,1}$, otherwise, we check the line segment $w_i w_2$ until the line segment $w_i w_m$. The pseudocode is given as follows:

```
Function Visibility graph (VP, F)
Assume $VP$ is the Voronoi path, which has $m$ waypoints.
For $i = 1{:}m$
  For $j = 1{:}m$
    Check whether line segment $VP(i,j)$ has intersection with the coastlines or not.
    If yes, continue.
    If no, update $F(i,j) = E(i,j)$.
 end
```

After applying the Visibility graph algorithm, we can get more possible paths, as shown in Fig. 4. 6. The Visibility graph is represented by the blue lines around the green line. Visibility graph has $O(m^2)$ edges and constructing the collision-free Visibility graph requires $O(m^2 log(n))$ time.

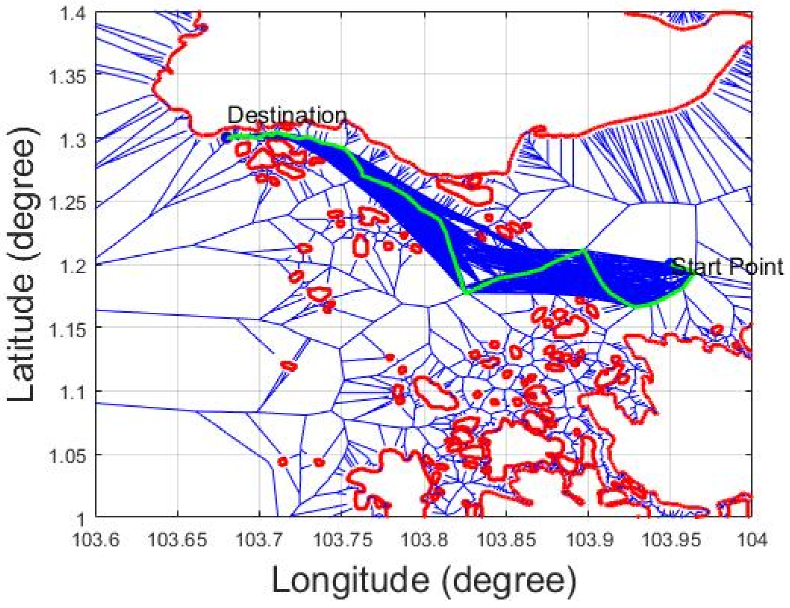


Fig. 4. 6. Voronoi-Visibility roadmap

4.4.6 Voronoi-Visibility Energy Efficient Path Generation

The Dijkstra's search algorithm is applied again to search the Voronoi-Visibility energy-efficient path from the Voronoi-Visibility roadmap, as shown in Fig. 4. 6. The generated Voronoi-Visibility energy-efficient path is shown in Fig. 4. 7. For comparison, we change the element of the node matrix F from the value of the energy consumption to the length of the nodes, then we applied the Dijkstra's search algorithm, we get the Voronoi-Visibility shortest path. The Voronoi-Visibility energy efficient path and Voronoi-Visibility shortest path are combined in Fig. 4. 8. The blue arrows represent the sea current state. The black line represents the Voronoi-Visibility shortest path and the green line represents the Voronoi-Visibility energy efficient path. It can be seen that the energy efficient path will save the USV more energy by making it follow the direction of the favourable flow. The energy consumption of the Voronoi-Visibility energy efficient path is 0.1831 and the energy consumption of the shortest path is 0.2517, therefore, the former path saves 27.26% energy than the later path.

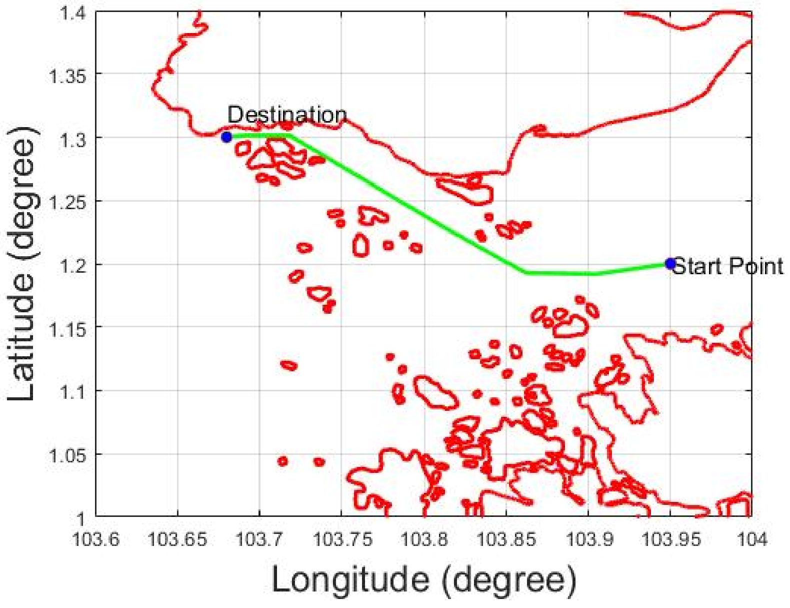


Fig. 4. 7. Voronoi-Visibility energy-efficient path

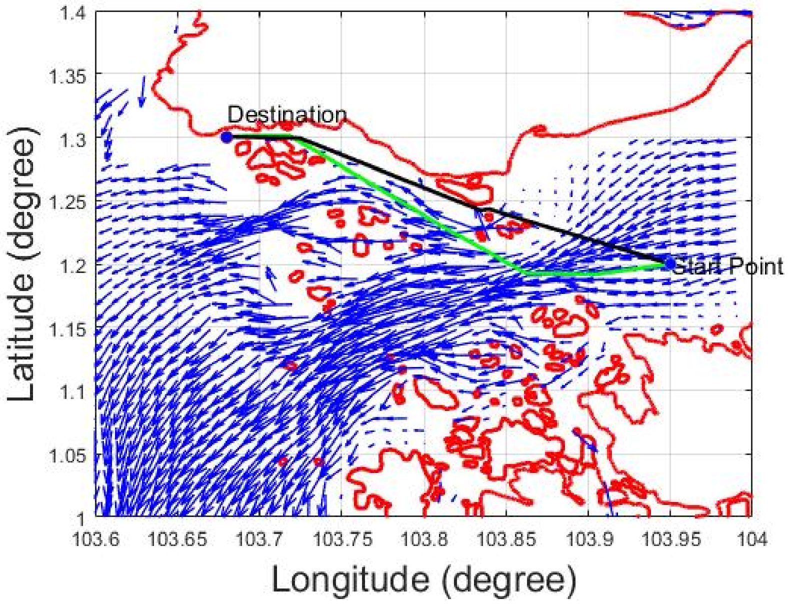


Fig. 4. 8. Voronoi-Visibility energy-efficient path vs shortest path

4.5 Numerical Simulation

In this section, the Voronoi-Visibility energy-efficient path planning algorithm is compared with the Voronoi-Visibility shortest path planning algorithm in ten Singapore Strait missions and five Croatian islands missions. The effects of mission time, USV traveling speed and sea current states on the results are investigated. The computational time of the proposed Voronoi-Visibility algorithm is also compared with that of the Voronoi energy-efficient path planning algorithm.

4.5.1 Voronoi-Visibility Shortest Path Planning Algorithm for Comparison

The Voronoi-Visibility shortest path planning algorithm is implemented to compare the energy consumption difference between the shortest path and the energy efficient path. The architecture of the Voronoi-Visibility shortest path is shown in Fig. 4. 9. The shortest path planning algorithm does not take account of the sea current data and the length between the adjacent Voronoi nodes is used as the weight of the node matrix. Dijkstra's search algorithm is first applied to search the Voronoi shortest path, after the Visibility graph algorithm is implemented to generate more possible paths, Dijkstra's search algorithm is used again to search the Voronoi-Visibility shortest path.

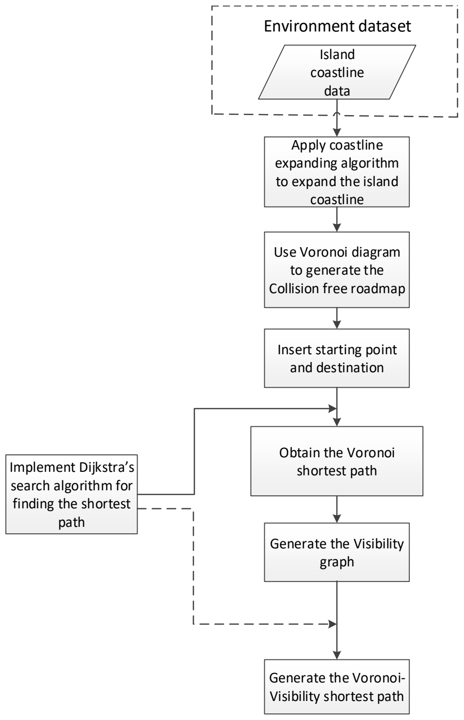

Environment dataset
Island coastline data
Apply coastline expanding algorithm to expand the island coastline
Use Voronoi diagram to generate the Collision free roadmap
Insert starting point and destination
Implement Dijkstra's search algorithm for finding the shortest path
Obtain the Voronoi shortest path
Generate the Visibility graph
Generate the Voronoi-Visibility shortest path

Fig. 4. 9. The architecture of the Voronoi-Visibility shortest path planning algorithm

4.5.2 Ten USV Mission Scenarios in Singapore

Ten USV mission scenarios are used to demonstrate the energy efficiency of the proposed algorithm. These ten USV missions differ in starting point, destination, mission time and USV speed. The USV keeps constant speed while it is traveling and it is assumed when the USV is traveling, the sea current is spatially variant and not temporally variant. The comparison of energy efficient path and shortest path is shown in Table. 4. 2.

Table. 4. 2. Comparison of energy efficient path and shortest path in ten USV missions of Singapore Strait

Mission No.	Starting Point	Destination	Mission time	Speed (m/s)	Efficient path length(km)	Shortest path length(km)	Efficient path energy consumption	Shortest path energy consumption	Energy saved
1	(103.95, 1.20)	(103.68, 1.30)	10:00am 11/06/2014	1	34.729	32.497	0.1157	0.2161	46.48%
2	(103.95, 1.20)	(103.68, 1.30)	6:00pm 11/06/2014	1	33.929	32.497	0.5068	0.5968	15.08%
3	(103.95, 1.20)	(103.75, 1.05)	6.00am 11/06/2014	1	31.910	28.647	0.0980	0.2079	52.84%
4	(103.95, 1.20)	(103.75, 1.05)	6.00am 11/06/2014	2	31.134	28.647	0.6868	0.9216	25.47%
5	(103.95, 1.20)	(103.75, 1.05)	6.00am 11/06/2014	3	31.052	28.647	1.8301	2.1503	14.89%
6	(103.95, 1.15)	(103.75, 1.12)	6.00am 11/06/2014	1	26.123	23.054	0.0867	0.1305	33.59%
7	(103.68, 1.30)	(103.95, 1.20)	6.00am 11/06/2014	2	32.805	32.497	1.2592	1.2754	1.27%
8	(103.74, 1.30)	(103.80, 1.08)	6.00am 11/06/2014	2	25.558	25.509	0.8779	0.8955	1.97%
9	(103.98, 1.20)	(103.75, 1.05)	6.00am 11/06/2014	2	34.509	31.026	0.7815	0.9939	21.38%
10	(103.90, 1.08)	(103.68, 1.30)	6.00am 11/06/2014	2	36.690	36.361	1.1560	1.1713	1.30 %

It can be seen that in these ten missions, the energy efficient paths save more energy than the shortest paths. The energy saving amount differs from 1.29% to 52.84%. The results of these ten USV missions differing in mission time, traveling speed as well as the sea current state are analyzed as follows.

4.5.2.1 USV Missions with Different Mission Times

Missions No.1 and No.2 have different mission time. The corresponding energy-efficient path and the shortest path are shown in Fig. 4. 10 and Fig. 4. 11. The green lines represent the energy efficient path and the black lines represent the shortest path. The blue arrows represent the direction and the magnitude of the sea current. Mission No.1 started at 10:00 am, from Fig. 4. 10, it can be seen that the sea current flowed from east to west and the USV also travelled from east to west. Both the first half part and the second half part of the energy efficient paths made the USV take advantage of the sea current state. The shortest path made the USV cross the sea current in the beginning without taking account of the sea current state, therefore, the energy efficient path saved more energy than the shortest path. In Mission No.2, the sea current state changed its direction and flowed from west to east, therefore, the energy efficient path was different with the path in Mission No.1. The first half of the energy efficient path made the USV cross the direction of the sea current instead of navigating the USV travel through the counter-flow directly like the shortest path.

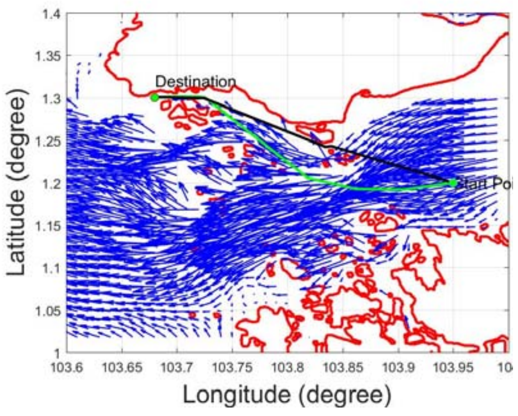


Fig. 4. 10. Energy efficient path and shortest path in Mission No.1

(mission time 10:00 am 11/06/2014)

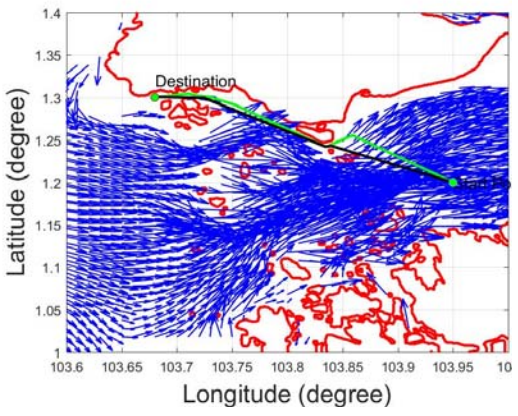


Fig. 4. 11. Energy efficient path and shortest path in Mission No.2

(mission time 18:00 pm 11/06/2014)

The energy-efficient path saved 46.48% energy than the shortest path in Mission No.1 and saved 15.08% in Mission No.2. It can be seen that it is easier for the USV to save more percentage of the total energy than the corresponding shortest path when it travels from the east to the west in the morning than in the evening.

4.5.2.2 USV Missions with Different Travelling Speeds

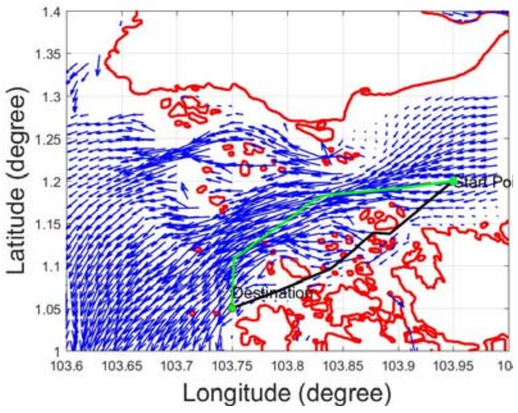


Fig. 4. 12. Energy efficient path and shortest path in Mission No.3 (USV speed is 1m/s)

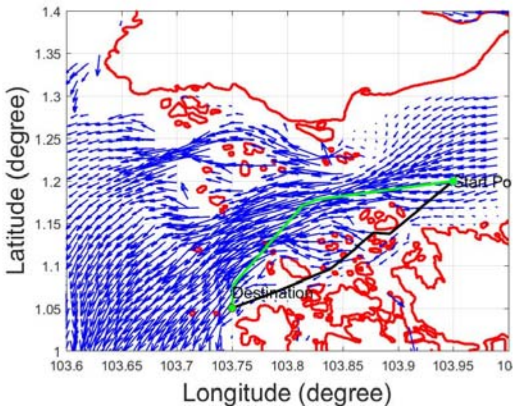


Fig. 4. 13. Energy efficient path and shortest path in Mission No.4 (USV speed is 2m/s)

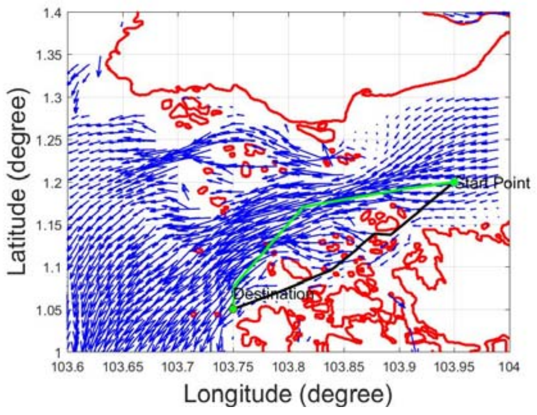


Fig. 4. 14. Energy efficient path and shortest path in Mission No.5 (USV speed is 3m/s)

Missions No.3, No.4 and No.5 have different traveling speed and the path planning results have been shown in Fig. 4. 12, Fig. 4. 13 and Fig. 4. 14. The results show that the higher speed the USV utilizes, the more energy it will consume. In Mission No.3, the USV traveled at 1m/s, it consumed 0.0980 energy. However, in Mission No.5, traveling at 3m/s, it consumed 1.8301 energy. This is because traveling at a high speed will cause high hydrodynamic drag. The results also show that traveling at a low speed on the energy efficient path will save more percentage of total energy compared with the corresponding shortest path than traveling at a high speed. In Mission No.3, it saved 52.84% of the total energy, and in Mission No.5, it just saved 14.89% of the total energy.

4.5.2.3 USV Missions in Favourable Flow, Cross-flow and Counter-flow

In Mission No.3, the USV encountered the favourable flow, as shown in Fig. 4. 15. In Mission No.7, the USV encountered the counter-flow, as shown in Fig. 4. 16.

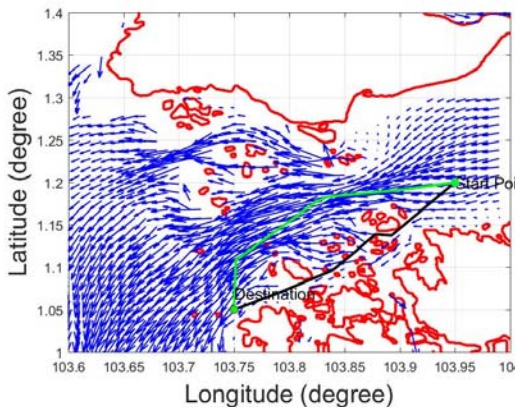


Fig. 4. 15. Energy efficient path and shortest path in favourable flow (In Mission No.3, sea current flows from east to west)

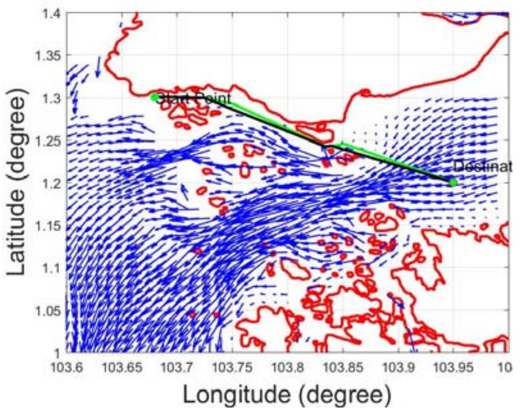


Fig. 4. 16. Energy efficient path and shortest path in counter-flow situation (In Mission No.7, sea current flows from east to west)

In Mission No.8 and No.10, the USV encountered cross flow, as shown in Fig. 4. 17 and Fig. 4. 18. In the favourable mission, the efficient path saved 52.84% energy. However, in the counter-flow and cross flow mission, the energy efficient paths are close to the corresponding shortest path and just saves small amount of energy comparing with the shortest path (saving 1.27% in Mission No.7, saving 1.97% in Mission No.8 and saving 1.30% in Mission No.10). This result indicates that in the favourable flow mission, it is easier for the proposed algorithm to save more percentage of total energy than the shortest path.

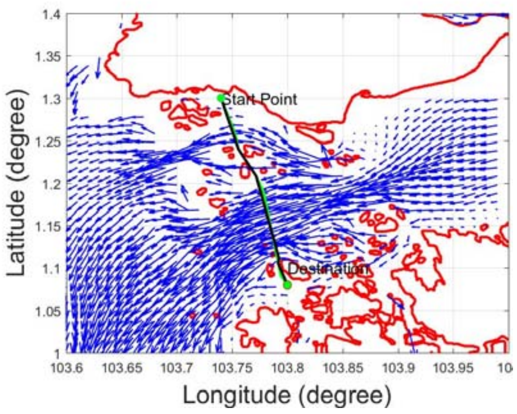


Fig. 4. 17. Energy efficient path and shortest path in cross-flow situation (Mission No.8, sea current flows from east to west)

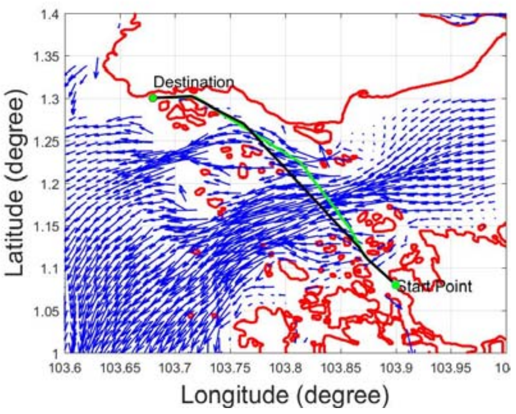


Fig. 4. 18. Energy efficient path and shortest path in cross-flow situation (Mission No.10, sea current flows from east to west)

4.5.2.4 Computational Time Comparison

The path planning based on the Voronoi diagram is computationally efficient but the result is far from optimal. The proposed Voronoi-Visibility algorithm intends to improve the quality of the Voronoi energy efficient path planning algorithm by using Visibility graph to generate more candidate paths. The Visibility graph can provide the optimal solution but the computational capability is not practical. The proposed algorithm intends to improve the quality of the Voronoi path and still keep the same level of the computational efficiency as that of the Voronoi energy efficient path planning algorithm.

The quality comparison of the Voronoi energy efficient path and the Voronoi-Visibility energy efficient path are shown in Fig. 4. 19 and Fig. 4. 20, respectively. The black line represents the Voronoi energy efficient path, the green line represents the Voronoi-Visibility energy efficient path and the blue arrows represent the sea current state. It can be seen that the Voronoi-Visibility path can navigate the USV to the area that has more favourable flow. Moreover, the Voronoi-Visibility path is more smooth than the Voronoi path.

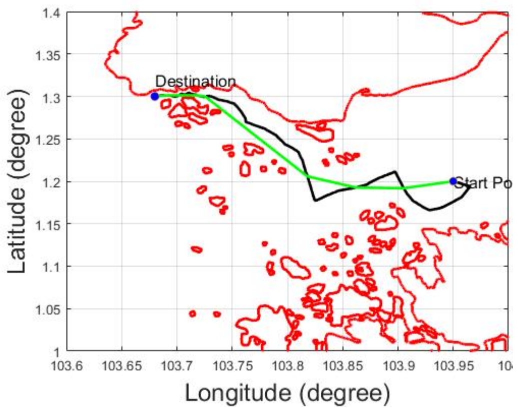


Fig. 4. 19. Voronoi algorithm vs VV algorithm in Mission No.1

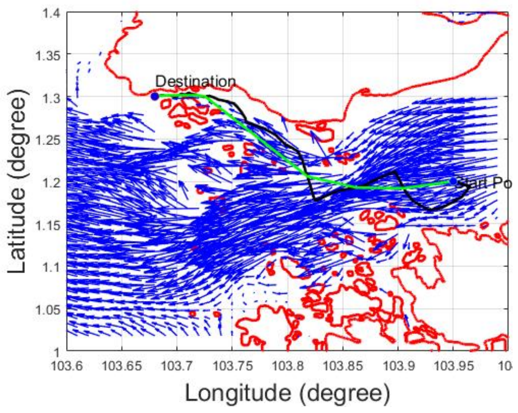


Fig. 4. 20. Voronoi algorithm vs VV algorithm with sea current state in Mission No.1

It is known that the Visibility graph can generate optimal result but will cost a significant computational time. Instead of applying Visibility graph algorithm to the islands' vertices of the entire map, the proposed algorithm applies the Visibility graph algorithm partially to the waypoints of the Voronoi energy efficient path. The computational time of planning the ten missions is recorded, as shown in Table. 4. 3. The proposed Voronoi-Visibility energy efficient path planning algorithm is divided into four parts: collision-free Voronoi roadmap generation (Part 1), Voronoi energy efficient path generation (Part 2), Visibility graph generation (Part 3) and Voronoi-Visibility energy efficient path generation (Part 4). t_V represents the total computational time of the Voronoi energy efficient path planning algorithm (Part 1 + Part 2), including collision-free Voronoi roadmap generation and Voronoi energy efficient path generation. t_{VV}represents the total computational time of the whole proposed Voronoi-Visibility energy efficient path planning algorithm (Part 1 + Part 2 + Part 3 + Part 4). The waypoints number of the Voronoi energy efficient path, which is used for generating the Visibility graph, is also recorded. Note that all the experiments were carried out on a 2.7 GHz Intel Core i7-6820HK processor with 16.0 GB. The program was implemented in Matlab R2016b.

From Table. 4. 3, we can find the computational time of Voronoi energy efficient path planning algorithm t_V is between 24 and 30 seconds. The additional time of yielding the Visibility graph is proportional to the number of the waypoints on the Voronoi path those are used for building the Visibility graph (Part 3), which is between 0.421 seconds and 3.036 seconds. Building the Visibility graph (Part 3) even costs much less time than building the Voronoi collision-free roadmap (Part 1). Comparing the computational time of the whole Voronoi-Visibility path planning algorithm t_{VV} and that of the Voronoi energy efficient path planning algorithm t_V, we can say the Voronoi-Visibility algorithm still keeps the same level of the computational efficiency as that of the Voronoi algorithm. The reason is that we applied Visibility graph to the Voronoi path, whose number of waypoints is between 34 and 115, instead of applying it to the entire map, whose number of vertices is 4128. In our proposed algorithm, the collision-free roadmap is constructed in $O((n+m^2)\log(n))$ time and the number of the edges is $O(n+m^2)$. According to (Bhattacharya & Gavrilova, 2008), when applying Visibility graph algorithm to the islands with 1866 vertices and only the length of the segment was computed, the computing time was longer than 1 minute. In our case, if the Visibility graph algorithm is implemented to the entire map (4128 vertices), the computational time should be at minutes-level because $O(n^2)$ edges will be queried whether they intersect with the coastlines or not and the energy consumption cost also needs to be calculated for each edge. Through the quality comparison and the computing time comparison $(t_{VV} - t_V)$, we can conclude that the proposed Voronoi-Visibility energy efficient path planning algorithm not only improves the quality of the Voronoi energy efficient path planning algorithm but also keeps the computational efficiency of the Voronoi energy efficient path planning algorithm, which makes it practical for processing large spatial dataset.

Table. 4. 3. The comparison of computational time in ten missions of Singapore Strait

Mission No.	Part 1 (seconds)	Part 2 (seconds)	Part 3 (seconds)	Part 4 (seconds)	t_V (seconds)	$t_{VV} - t_V$ (seconds)	Number of waypoints on Voronoi path
1	24.648	1.189	2.132	0.116	25.837	2.248	97
2	24.467	1.188	2.825	0.119	25.655	2.944	113
3	26.861	1.412	0.114	0.114	28.273	0.901	51
4	25.965	1.288	0.761	0.119	27.253	0.880	51
5	23.277	1.178	0.734	0.119	24.455	0.853	51
6	26.234	1.648	0.421	0.117	27.882	0.539	34
7	25.036	1.388	2.402	0.118	26.424	2.520	101
8	28.485	1.276	1.084	0.116	29.761	1.200	66
9	23.773	1.308	0.794	0.120	25.081	0.915	52
10	25.989	1.234	3.036	0.118	27.223	3.154	115

4.5.3 Five USV Mission Scenarios in Croatia

The proposed energy efficient algorithm and the shortest path planning algorithm are also compared in five mission scenarios of Croatian islands. The results are shown in Table. 4. 4. It can be seen that the energy efficient path saves more energy than the shortest path, however, the amount of the saved energy rate is less than those of the Singapore missions. We analyzed the Mission No.1 of Croatian islands and Mission No.4 of Singapore Strait, as shown in Fig. 4. 21 and Fig. 4. 22, respectively.

Table. 4. 4. Comparison of energy efficient path and shortest path in five missions of Croatian islands

Mission No.	Starting Point	Destination	Mission time	Speed (m/s)	Efficient path length(km)	Shortest path length(km)	Efficient path energy	Shortest path energy	Energy saved
1	(14.45, 45.20)	(14.50, 44.10)	3:00am 11/06/2014	1	136.477	136.229	1.0561	1.0785	2.07%
2	(14.50, 44.80)	(14.50, 44.10)	3:00am 11/06/2014	1	89.515	89.162	0.6338	0.6380	0.65%
3	(14.60, 44.90)	(14.50, 44.10)	3.00am 11/06/2014	1	97.633	97.121	0.7052	0.7148	1.35%
4	(14.45, 45.20)	(14.50, 44.10)	6.00am 11/06/2014	1	145.768	136.229	1.0094	1.0517	4.03%
5	(14.80, 44.50)	(14.80, 44.00)	6.00am 11/06/2014	1	71.490	64.848	0.5046	0.5189	2.75%

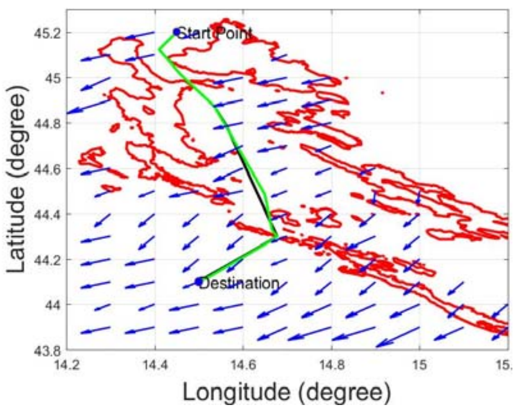


Fig. 4. 21. Energy efficient path and shortest path in Mission No.1 of Croatian islands

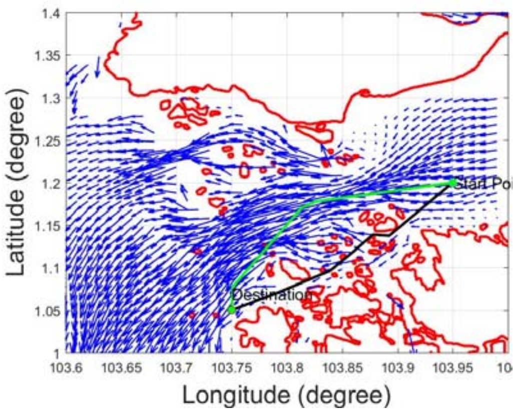


Fig. 4. 22. Energy efficient path and shortest path in Mission No.4 of Singapore Strait

From Fig. 4. 21 and Fig. 4. 22, it can be seen that although the Croatia area is larger than the Singapore area, the sea current of Croatia does not change as dramatically as Singapore area with the geographical variation. This leads to the small variation between the energy consumption of the energy efficient path and shortest path. However, the proposed energy efficient path planning algorithm can still provide energy efficient solutions in Croatia. In long-range missions, the energy saving amount cannot be ignored.

4.6 Summary

In this research, the Voronoi-Visibility (VV) energy efficient path planning algorithm integrates the Voronoi diagram, Visibility graph, Dijkstra's search algorithm and energy consumption function. Moreover, by applying the coastline expanding algorithm, the proposed algorithm also ensures the USV safety by keeping the USV a user-configurable clearance distance from the coastlines. By searching for the candidate path with the minimum energy cost, the proposed algorithm yields the energy efficient path. The proposed algorithm intends to improve the quality of the path generated from the Voronoi diagram and still keeps the computational efficiency of the Voronoi path planning algorithm. The proposed VV energy efficient path and the VV shortest path were compared in ten missions of Singapore Strait. It can be concluded that the VV energy efficient path cost less energy than the shortest path, which saved energy between 1.27% and 52.84%. By analysing the effects of the mission time, travelling speed and sea current states on the energy saving amount, we can conclude that in the favourable sea current and low speed situation, it is easier for the proposed VV energy efficient path to save more percentage of the total energy than the shortest path. Though the comparison of the quality and the computational time between the VV energy efficient path planning algorithm and Voronoi energy efficient path planning algorithm, we can find the proposed VV algorithm makes the path more smooth and enables the USV follow the favourable flow more closely, moreover, it still keep the same level of the computational efficiency as that of the Voronoi energy efficient path planning algorithm, which demonstrates that the proposed Voronoi-Visibility integrates the advantages of the Voronoi diagram and Visibility graph. In the simulation of Croatian missions, the flexibility of using the proposed algorithm in different geographical scenario is demonstrated. It also shows that the proposed algorithm will save more energy in the fast-changing sea current scenario than the regular scenario.

5 Energy Efficient Path Planning Algorithm in Spatially-Temporally Variant Environment

5.1 Abstract

Unmanned Surface Vehicles (USVs) are increasingly being used for persistent ocean missions. Typically, these USVs are deployed for a long period of time and operate with limited energy. As a result, there is increased interest in recent years on developing energy-efficient path for the marine vehicles. In this research, a novel energy efficient path planning algorithm is proposed for addressing the problem of the complex geographic dataset and the problem of spatially-temporally variant sea current. The proposed algorithm is realized by integrating the Voronoi roadmap and Genetic Algorithm (GA). The proposed approach also ensures that the USV remains at a user-configurable safety distance away from all islands and coastlines, hence maintaining the safety of USVs due to the uncertainties or inaccuracies in the map data. The proposed algorithm mainly includes generating collision-free roadmap and searching for energy-efficient path. The Global Self-Consistent Hierarchical High-Resolution Shorelines dataset and the historical sea current dataset are applied in this research to demonstrate the flexibility and the practicability of the proposed algorithm. The coastline expanding algorithm and Voronoi diagram are firstly applied to process the geographical data for generating the collision-free roadmap. Once the roadmap is generated, the proposed energy consumption function will be applied to calculate the energy consumption weights of the path segments. Through applying the Visibility graph and Dijkstra's search algorithm, the first generation of GA paths are generated. Using the first generation of GA paths and applying the proposed selection operator, crossover operator and mutation operators, the final energy-efficient path is generated. The energy efficiency and the practicability of the proposed algorithm were finally demonstrated by using three USV missions with complex geographic environment (98 islands) and spatially-temporally variant sea current. By comparing with the Voronoi-Visibility energy-efficient path planning algorithm, the total energy consumption of the proposed algorithm saves up to 27.68% of the total energy consumption in these three

missions, while keeping a user-configurable clearance distance from the coastlines.

5.2 Introduction

Path planning represents an important characteristic of autonomous systems. It represents the possibility for Unmanned Surface Vehicle (USV) and Unmanned Underwater Vehicle (UUV) deployment during a mission in presence of different disturbances and uncertainties, which is a complicated procedure due to existence of the unknown, inaccurate and varying information. Path planning for USVs and UUVs in general aims to optimize at least one of the following aspects of performance: traveling time, energy consumption and safety. Typically, the USVs and UUVs are deployed for long-term missions with limited energy, as a result, there is increasing interest in developing energy efficient path planning algorithm in recent years. In this research, we focus on optimizing the energy consumption and safety of the USV.

A level-set based approach for UUV path planning where the external flow was explicitly accounted for was described in the work of Lolla et al (Lolla, et al., 2012) (Lolla, et al., 2014) (Lolla, et al., 2015). In the level-set approach, time-optimal trajectories for single or multiple vessels are determined by employing a level set expansion of the flow field given the desired starting and goal positions for the vehicles. The underlying motion model behind the level-set method includes a velocity model, which is very often restrictive. Since the strategy requires full knowledge of the flow field and requires significant computational resources when performing the various level set expansions, the strategy is not amenable for real time planning purpose (Kularatne, et al., 2016).

The heuristic-grid searching approaches are another alternatives in dealing with NP-hard and multi-objective optimization problems. Carroll (Carroll, et al., 1992) represented the operational space with quad trees and applied A* to generate the efficient paths with minimum cost. An A* based energy efficient path planning algorithm is proposed by taking ocean current data into account (Garau, et al., 2014), in which a common situation on actual AUVs are considered that the thrust is assumed to be constant. An A* based path planning algorithm of AUVs is applied in the work of (Garau, et al., 2005). The energy optimal paths are calculated in a simulated ocean environment with high spatial variability in the form of different types of eddies. The effect of different heuristic functions on the results is analyzed. However, deterministic and heuristic methods like A* suffer from expensive computational cost and criticized for their weak performance in high-dimensional problems (MahmoudZadeh, et al., 2016).

Dynamic programming has been used as a graph-searching method when a cost is associated with the edge of a graph (Bryson & Ho, 1975). Although dynamic programming is able to produce the optimal solution, the computational time will increase geometrically with the dimension of the solution space. A path planning method based on mixed integer linear programming (MILP) is presented for handling multiple AUVs (Yilmaz, et al., 2008). However, the computational time of this approach will increase exponentially with the problem size, this approach has limitations in real-sized applications (Eichhorn, 2015). In the case of USV path planning in a spatially variant and temporally variant sea current environment, dynamic programming and MILP may not be computationally feasible.

The application to path planning of a search procedure based on the Darwinian theories of natural selection and survival has been recognized in the work of Goldberg & Holland (Goldberg & Holland, 1988). In these theories, called Genetic Algorithms (GAs), a population of possible paths is maintained and the paths are iteratively transformed by genetic operators such as crossover and mutation (Alvarez, et al., 2004) (Zadeh, et al., 2016). GAs have been applied in AUV path planning algorithms (Alvarez, et al., 2004) (Zadeh, et al., 2016). Unlike dynamic programming, the computational complexity of GAs increases linearly with the dimension of the solution space (Alvarez, et al., 2004), which is ideal for searching the high dimensional dataset. Previously proposed GAs based algorithm (Zadeh, et al., 2016) (Alvarez, et al., 2004) have applied GAs in AUV path planning algorithm and used grid map or cell map to represent the collision-free space and applied random walk to generate the initial population. However, in the USV mission scenario, the mission operational area may be complex spatial environment, for example, there are 98 islands in Singapore. In the complex spatial environment, the random walk method is inefficient to find the feasible paths from the starting point to the destination (Zhang, et al., 2008), which will be used as the initial population of the GA.

The previous GA based energy efficient path planning algorithms have applied grid maps to represent the collision-free space. However, very little work of integrating the Voronoi roadmap and GA method has been done to deal with the spatially-temporally variant sea current problem. The Voronoi roadmap can be used as the environment model that is more computationally efficient and better adapted to the context of mobile robots than regular grids (Benavides, et al., 2011) and Voronoi diagrams are computationally efficient in two-dimensional environments (Pehlivanoglu, 2012). The Voronoi-Visibility roadmap and Dijkstra's search algorithm are integrated for generating the energy efficient path in (Niu, et al., 2016). The computational efficiency and energy efficiency of the algorithm have been validated by comparing with the Voronoi-Visibility shortest path in (Niu, et al., 2016). The proposed algorithm can also ensure the user-configurable clearance distance from the coastlines. However, this algorithm just takes account of the spatially variant sea current and does not address the temporally variant sea current problem.

In this research, we combine the ideas presented in (Benavides, et al., 2011) and (Niu, et al., 2016). In (Benavides, et al., 2011), a shortest path planning algorithm is proposed by integrating the Voronoi roadmap and GA method. The motivation for this combination is to keep the flexibility and computational efficiency of the Voronoi diagram algorithm and the computational efficiency of the GA algorithm in searching high dimensional dataset so as to generate the energy efficient path in the spatially-temporally variant environment. The proposed algorithm was finally simulated and compared with the Voronoi-Visibility algorithm in three USV missions with a complex geographic environment (98 islands) and spatially-temporally variant sea current scenario.

The rest of this chapter is organized as follows: The problem statement is given in Section 5.3. The methodology is presented in Section 5.4. The Voronoi-Visibility energy efficient path planning algorithm and the proposed algorithm are simulated and compared in Section 5.5. The summary is given in Section 5.6.

5.3 Problem Statement

Besides the large geographic dataset and clearance distance problem, in long term USV mission, the sea current will change with time. The sea current state changes of Singapore Strait from 0:00am to 05:00am on 11/06/2014 are shown from Fig. 5. 1 to Fig. 5. 6.

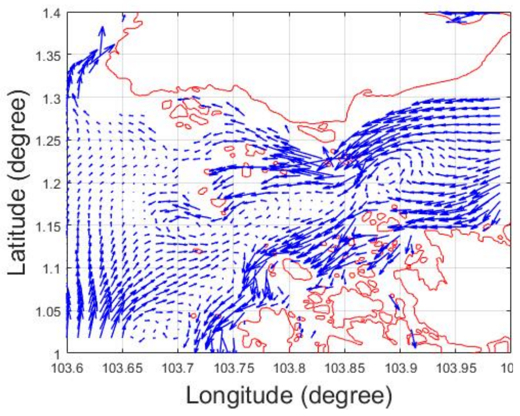


Fig. 5. 1. Sea current state at 00:00am on 11/06/2014

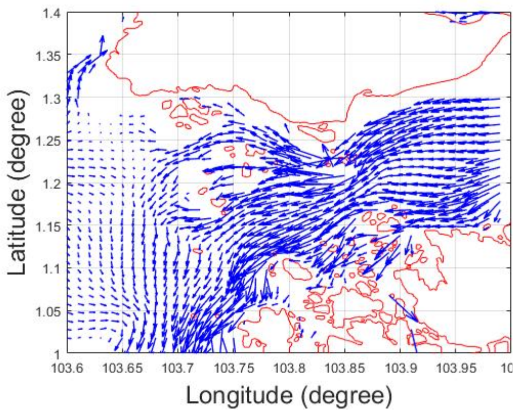


Fig. 5. 2. Sea current state at 01:00am on 11/06/2014

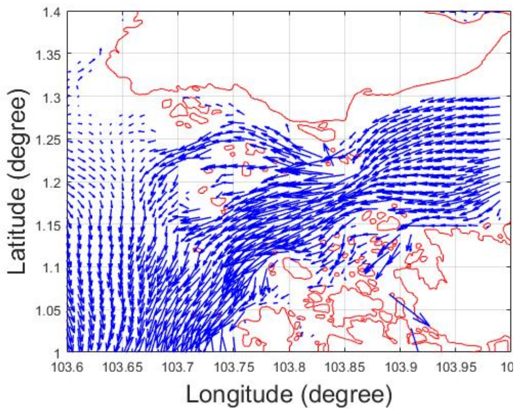


Fig. 5. 3. Sea current state at 02:00am on 11/06/2014

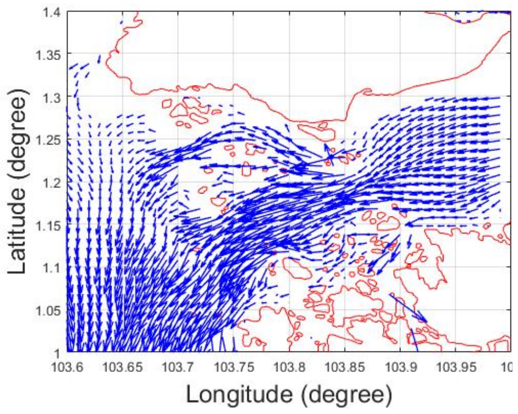


Fig. 5. 4. Sea current state at 03:00am on 11/06/2014

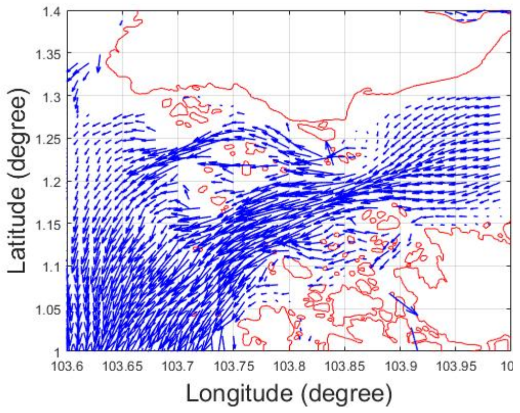


Fig. 5. 5. Sea current state at 04:00am on 11/06/2014

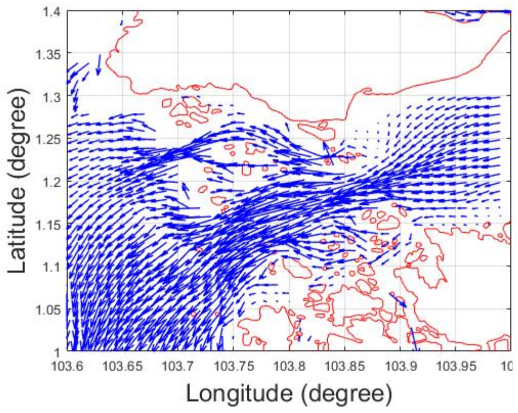


Fig. 5. 6. Sea current state at 05:00am on 11/06/2014

From Fig. 5. 1 to Fig. 5. 6, it can be seen that the tide began to rise at 0:00 am and the water came into the Singapore Strait from the East. At 2:00 am, more and more water came into the west part of Singapore Strait. The tide on the west side began to rise. Comparing Fig. 5. 1 and Fig. 5. 6, the large difference of the sea state between 0:00 am and 5:00 am is illustrated. If the USV encounters dramatically temporally variant sea current state, it will be necessary for an USV path planning algorithm to take account of not just the spatial variance of the sea current but also the temporal variance, which will save the total energy consumption during the entire USV mission.

5.4 Methodology

Taking account of the spatially-temporally variant sea current, an energy efficient path planning algorithm based on Voronoi roadmap and GA is proposed. The algorithm includes two main steps: Firstly, the collision-free roadmap is generated. Secondly, GA is applied to search for the collision-free and energy efficient path. This section is organized as follow: The algorithm architecture is presented in Section 5.4.1; In Section 5.4.2, the environmental dataset used in this research will be introduced; The Voronoi collision-free roadmap generation and GA implementation will be presented in Section 5.4.3 and Section 5.4.4, respectively.

5.4.1 Algorithm Architecture

The architecture of the Voronoi roadmap generation is shown in Fig. 5. **7**. In the Voronoi roadmap generation, the coastline expanding algorithm is applied firstly to expand the island coastlines, which will keep a user-configurable clearance distance from the coastlines. Then Voronoi diagram is applied to generate roadmap around the islands. However, not all the roads of the roadmap are reachable and the unreachable paths are deleted by using the proposed principle of removing the unreachable paths. The GA architecture is shown in Fig. 5. 8. The Dijkstra's search algorithm and Visibility graph algorithm are implemented to generate the first generation of the GA. The objective pursued with the implementation of the GA is to find a collision-free and energy efficient path from the starting point to the destination. A fitness function is proposed for the above optimization criterion and the fitness function will be applied for evaluating all the individuals of each generation. In the GA implementation, the selection, crossover, mutation operators are applied. The mutation operators include dividing, smoothing and exchanging operators. By using the selection operator, the individuals those have the best fitness value will be selected and stored. The crossover operator shuffles the existing population solutions and searches for a better solution space. And the mutation operator is used for the diversity of the population. The evaluation, selection, crossover and mutation are applied in an iterative loop. Until the termination condition is satisfied, the iteration will stop and the best individual will be selected.

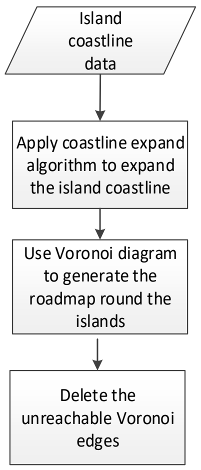

Island coastline data
Apply coastline expand algorithm to expand the island coastline
Use Voronoi diagram to generate the roadmap round the islands
Delete the unreachable Voronoi edges

Fig. 5. 7**. The architecture of the Voronoi collision-free roadmap generation**

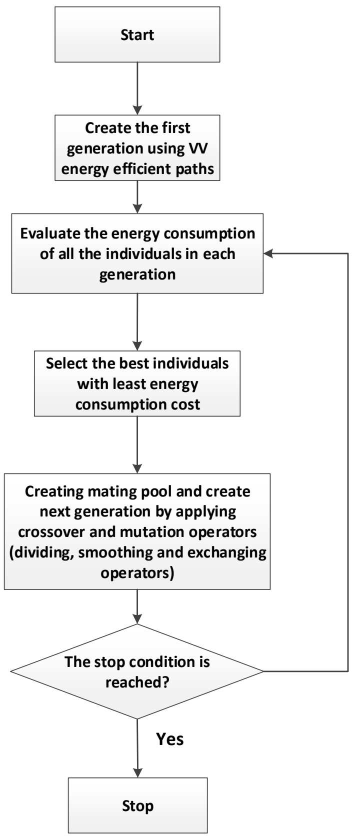

Start
Create the first generation using VV energy efficient paths
Evaluate the energy consumption of all the individuals in each generation
Select the best individuals with least energy consumption cost
Creating mating pool and create next generation by applying crossover and mutation operators (dividing, smoothing and exchanging operators)
The stop condition is reached?
Yes
Stop

Fig. 5. 8. The architecture of Genetic Algorithm

5.4.2 Environmental Dataset

The environmental dataset includes the island coastlines dataset and the sea current dataset. The dataset of the coastlines and sea current have been introduced in Section 4.4.2. The updates of the sea current during the mission will also be considered in this section.

5.4.3 Voronoi Collision-Free Roadmap Generation

The collision-free roadmap generation includes the coastline expanding algorithm implementation, Voronoi diagram algorithm implementation and removing the unreachable paths. The generation of collision-free roadmap with clearance distance c has been explained in detail in Section 3.5.3 (Niu, et al., 2016).

5.4.4 GA Implementation

5.4.4.1 Initialization

The first generation of the feasible paths are generated by using Voronoi-Visibility energy efficient path planning algorithm (Niu, et al., 2016). In (Niu, et al., 2016), after the Voronoi collision-free roadmap is generated, the energy cost function will be used to evaluate the energy consumption weight of each Voronoi edge and then Dijkstra's search algorithm is applied to search for the energy efficient Voronoi path. However, Voronoi path has redundant waypoints and the Visibility graph is applied to optimize the path. Finally, the Voronoi-Visibility energy efficient path is generated. The Voronoi-Visibility energy efficient path planning algorithm is suitable for the temporally-invariant sea current, therefore, we estimate the mission duration and apply the sea current data of each time updating interval for the generation of the first generation of paths. Note that the mission duration can be estimated by calculating the length of the first path and the travelling duration from the starting point to the destination. The mission duration estimation is not necessary accurate. For example, in Singapore Strait, the sea current data will be updated each hour, if the USV needs T hours for travelling, then the sea current data of each hour can be used for generating T feasible paths. These T feasible paths will be used for generating N feasible paths as the first generation by applying the mutation operator and the crossover operator.

5.4.4.2 Evaluation

The quality of the path is evaluated by the fitness function. The fitness function measures the energy cost of each path in the population. This is carried out by computing and adding up the energy required to overcome the drag generated by the sea current in each segment. There are two steps of calculating the energy cost in the spatially-temporally variant sea current: Firstly, we need to divide the path into equidistant segments based on the USV travelling distance of one hour. Assume the USV is commanded to travel from (103.68, 1.3) to (103.9, 1.08) at

0:00am on 11/06/2014 in Singapore Strait and we have a candidate path, as shown in Fig. 5. 9, the updating time interval of the sea current data is one hour (Tidetech, 2014) and the USV is commanded to travel at 1m/s, then we can divide the path based on the travelling distance of one hour, as shown in Fig. 5. 10, the path is divided by the red points, which is called one-hour break point. The second step is to calculate the energy cost of each equidistance path. When the USV is travelling on each equidistance path, we assume the sea current keeps constant in the corresponding time interval. For example, when the USV is travelling between one-hour break point 3 and 4, we will use the sea current data of 3:00am to calculate the corresponding energy cost. The illustrations are shown in Fig. 5. 11 and Fig. 5. 12. In In Fig. 5. 11, the path from point 3 to point 4 is selected in the red rectangle area and the corresponding sea current data at 3:00am is queried. As shown in Fig. 5. 12, the sea current state is represented by the blue arrows.

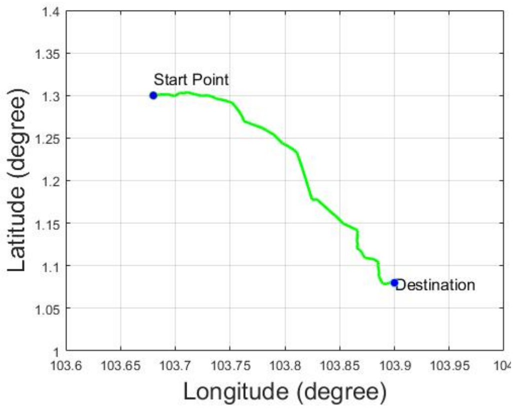


Fig. 5. 9. Candidate path

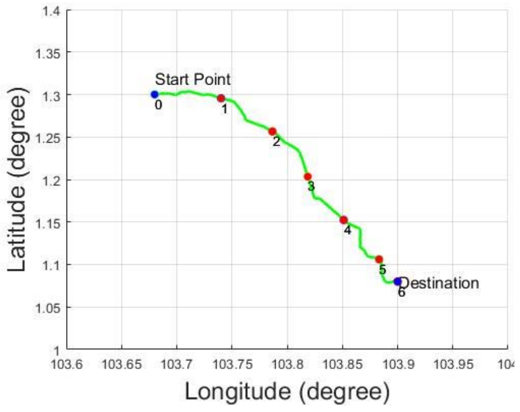


Fig. 5. 10. Candidate path with one hour break point

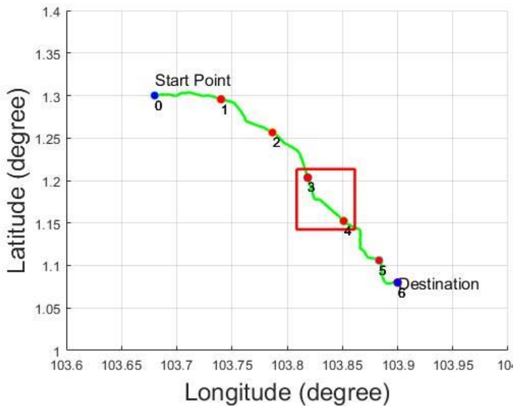


Fig. 5. 11. Candidate path with highlight area

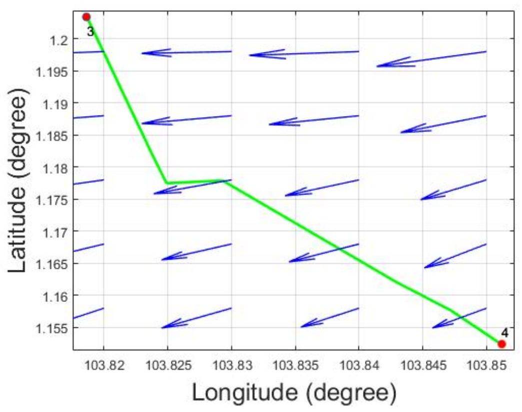


Fig. 5. 12. Sea current state from point 3 to point 4

The function of calculating the energy cost of the equidistant path is given in equation (4.11).

The total energy cost of the designated path is the sum of the energy cost of all the corresponding equidistant paths and the total energy cost is set to be the fitness function. Note that if the path has intersection with the expanded coastlines, it means this path is unavailable and the fitness value of this path will be set as a very large value.

5.4.4.3 Crossover

After evaluating the fitness value of N feasible paths, top 40% of the best feasible paths will be selected as parents for generating new individuals by applying crossover operator. The crossover operator is implemented by exchanging the genetic information of the parents. Each path consists a sequence of waypoints, which have been indexed. The index of the waypoint is used as the gene of the path. Assume we have two parents p_{w1} and p_{w2} randomly selected, and the waypoint index of these two parents are shown in (5. 1) and (5. 2).

$$p_{w1} = \{p_{w1,1}, \dots p_{w1,i}, \dots p_{w1,j}, \dots p_{w1,m}\} \tag{5. 1}$$

$$p_{w2} = \{p_{w2,1}, \cdots p_{w2,i}, \cdots p_{w2,j}, \cdots p_{w2,n}\} \tag{5. 2}$$

We randomly choose two integers i and j those are smaller than m and n, respectively. Then a new pair of offspring can be generated, as shown in (5. 3) and (5. 4). The crossover operator shuffles the existing populations and search for a better solution space.

$$p_{wo1} = \{[p_{w1,1}, \cdots p_{w1,i}], [p_{w2,j+1} \cdots p_{w2,n}]\} \tag{5. 3}$$

$$p_{wo2} = \{[p_{w2,1}, \cdots p_{w2,j}], [p_{w1,i+1} \cdots p_{w1,m}]\} \tag{5. 4}$$

5.4.4.4 Mutation

The mutation operator is used for the diversity of the population and the mutation ensures that the solutions do not converge prematurely to a stable local minimum (Alvarez, et al., 2004). There are three kinds of mutation operators used in this research: dividing operator, smoothing operator and exchanging operator.

(a) Dividing operator

Firstly, if a candidate path is generated, each line segment of this path will be queried, if the line segment is longer than a specified value L, this path will be divided to several equidistant line segments by inserting dividing waypoints and equidistance equals L, as shown in Fig. 5. 13 and Fig. 5. 14. In Fig. 5. 13, we assume two connected line segments of one individual path is AB and BC. We set L to be the distance of 10 minutes' travelling, which is set to be 1 in this demonstration, then we divide the AB and BC using the green equidistant points. The inserted waypoints are also indexed and they will be the new genes of the candidate path. The combination of the dividing operator and the crossover operator makes it possible for the USV to change direction in short time interval. This operator will be applied on all the existing and newly generated candidate paths.

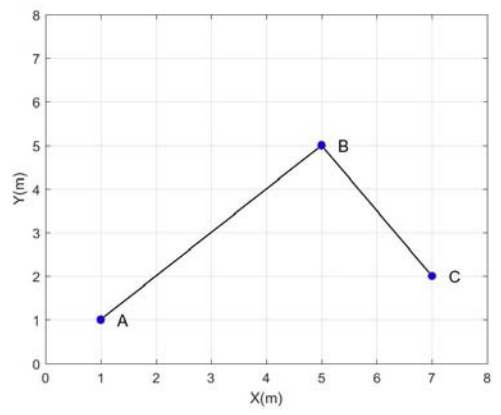


Fig. 5. 13.The path before applying dividing operator

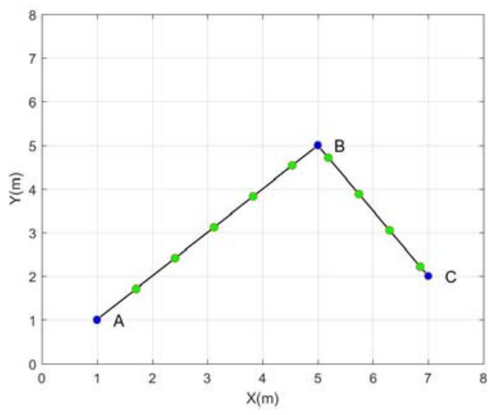


Fig. 5. 14. The path after applying dividing operator

(b) Smoothing operator

The smoothing operator is used for removing a waypoint from a candidate path randomly, as shown in Fig. 5. 15 and Fig. 5. 16. In Fig. 5. 15, assume we choose a point B randomly from a candidate path, there are two points connected to it, which are A and C. After we deleted B, we get the path AC, as shown in Fig. 5. 16. This operator can remove redundant waypoints from the candidate paths. This operator can be used for generating the smooth and energy efficient path.

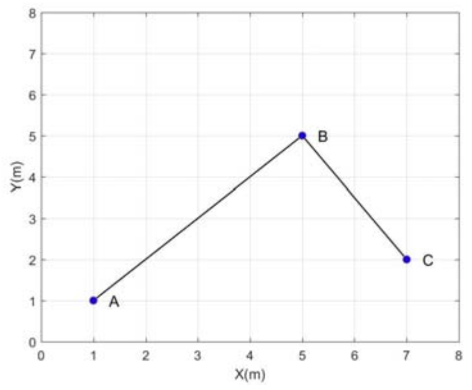


Fig. 5. 15. The path before applying smoothing operator

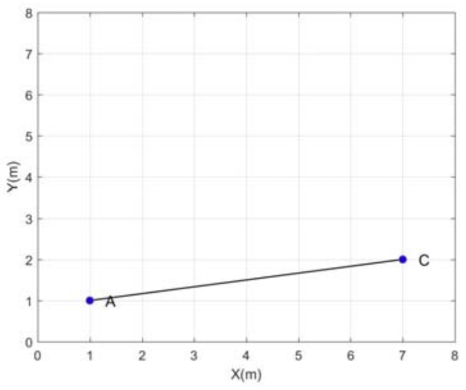


Fig. 5. 16. The path after applying smoothing operator

(c) Exchanging operator

The exchanging operator is applied by randomly exchanging a waypoint to the waypoint it connected. Assume we have two connected line segments AB and BC, and a waypoint D is connected to B, as shown in Fig. 5. 17, after we applied exchanging operator, our new path will be ADC, as shown in Fig. 5. 18. The smoothing operator or the exchanging operator will be used on half of the selected paths randomly.

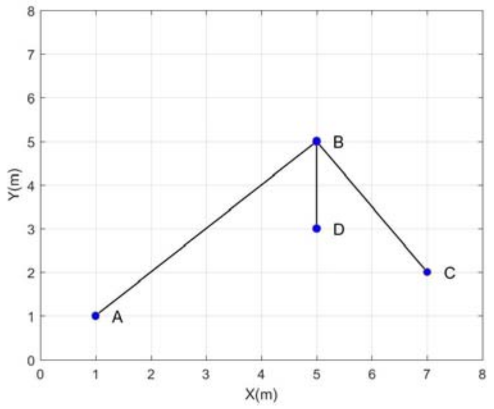


Fig. 5. 17. The path before using exchanging operator

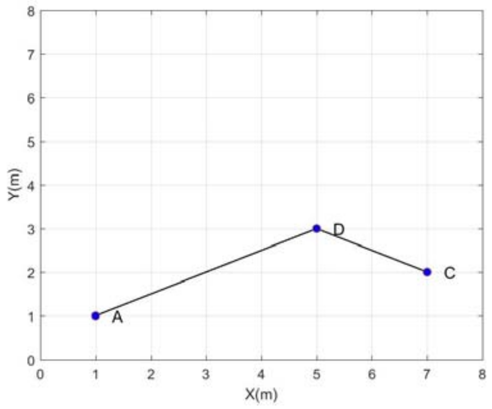


Fig. 5. 18. The path after applying exchanging operator

5.4.4.5 Selection and Iteration

The initial T paths are used to generate N paths by applying crossover and mutation operator. These paths are evaluated by using the fitness function. The feasible energy efficient path will have a low fitness value and the unfeasible path will have a high fitness value. The 40% N number of the paths, which have the lowest fitness value, will be selected to generate 40% N number of new individuals by using crossover operator. Half of the selected paths, which are 20% N number of paths, will be applied smoothing operator or exchanging operator randomly. In each iteration, top 40% of the paths are selected and these paths are stored so that the genes of the best paths are kept, together with the 40% N number of new individuals, and 20% N number of mutation paths, we have N number of paths in the end of the iteration. The iteration will stop until the termination condition is met, in this research, the termination condition is set as 20 times of iteration. The proportion of the individuals using each operator is given in Table. 5. 1.

Table. 5. 1. The proportion of the individuals using each operator

Number of individuals	Selecting the best individuals (%)	Crossover operator (%)	Smoothing or exchanging operator (%)	Dividing operator (%)
N	40	40	20	100

5.5 Numerical Simulation

To simulate and validate the proposed algorithm, a fast-changing sea current environment is simulated by integrating the discontinuous historical sea current data of the Singapore Strait. The relation between the simulated data and the historical data is shown in Table. 5. 2.

Table. 5. 2. The relation between the simulated data and the historical data

Simulated time interval	The time of the corresponding historical data
t_1	9:00am on 11/06/2014
t_2	12:00 (noon) on 11/06/2014
t_3	5:00pm on 11/06/2014
t_4	6:00pm on 11/06/2014
t_5	7:00pm on 11/06/2014

The Table. 5. 2 can be expressed as follow: In the first hour of the simulation, the historical data of 9:00am on 11/06/2014 will be used; In the second hour, the historical data of 12:00 (noon) on 11/06/2014 will be used. The rest of the table can be expressed similarly. We assume the sea current keeps constant in each hour interval in this research. Note that this assumption is for ease of analysis and are not a restriction of the approach. When the updating frequency of the data is higher, the proposed algorithm will still be suitable.

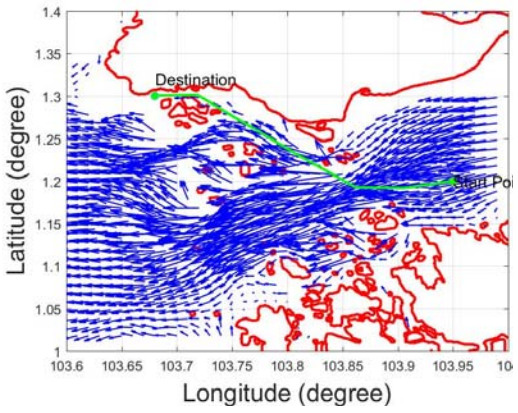


Fig. 5. 19. The energy efficient path using the data of the first hour

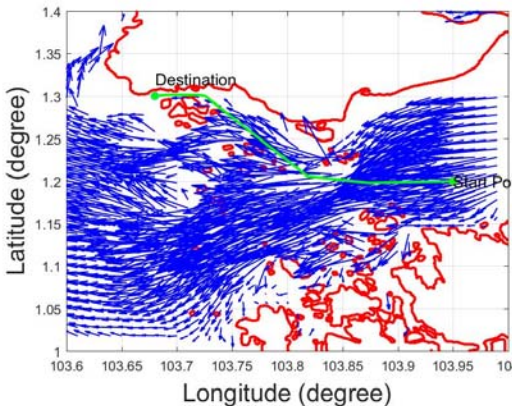


Fig. 5. 20. The energy efficient path using the data of the second hour

Assume the USV is commanded to travel from (103.95, 1.2) to (103.68, 1.3) and the travelling speed is kept as 2.5m/s. By using the sea current data of the first hour, we can get the corresponding energy efficient path through applying the Voronoi-Visibility energy efficient path planning algorithm. The path is shown in Fig. 5. 19. The energy efficient path is represented by using green line and the blue arrows represent the sea current state. The length of the path and the travelling duration can be calculated. The travelling duration is 3.79 hours. Then the data of the second hour, third hour and fourth hour are used for calculating another three energy efficient paths, as shown in Fig. 5. 20, Fig. 5. 21 and Fig. 5. 22. We can see from these four figures, during the first and second hour, the sea current directions are from east to west; During the third and fourth hour, the sea current directions are from west to east. The first and second energy efficient path will save energy in the first two hours, however, will waste energy in the latter two hours. In terms of the third and fourth energy efficient path, they can save energy in the latter two hours, however, will waste energy in the first two hours. It can be concluded that none of these paths can save energy in the entire travelling duration.

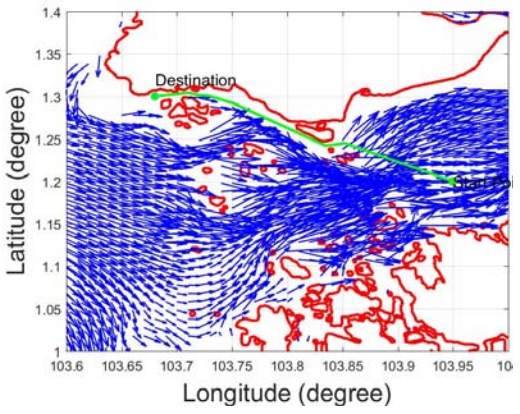


Fig. 5. 21. The energy efficient path using the data of the third hour

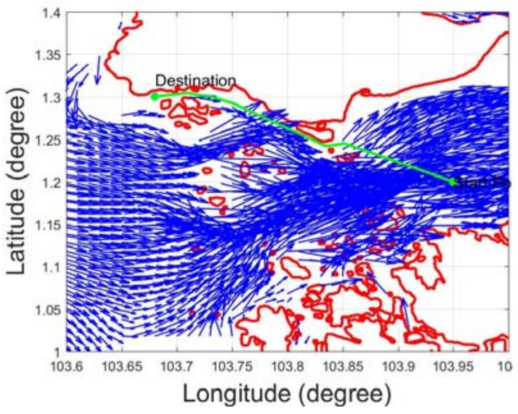


Fig. 5. 22. The energy efficient path using the data of the fourth hour

These four initial paths are used for generating 300 paths using crossover and mutation operators. Then 300 paths are evaluated and 120 best paths are selected as the parents of the next generation. These 120 best paths generate 120 new individuals by using crossover operators and generate 60 mutational individuals by using mutation operators. These 300 paths will be evaluated and selected again in the next iteration. When 20 times of iterations are finished, we choose the best individuals. For comparison, we combine the final energy efficient path with the four initial paths in Fig. 5. 23. In Fig. 5. 23, the green line represents the final energy efficient path generated by GA. The blue solid line and blue ‘--’ line represent the energy efficient paths of the first hour and the second hour respectively. The black solid line and black ‘--’ line represent the paths of the third hour and fourth hour respectively and they overlap with each other in this situation.

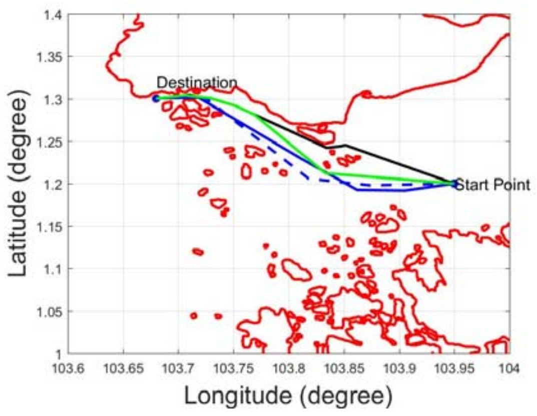


Fig. 5. 23. Final energy efficient path vs four initial energy efficient paths in Mission No.1

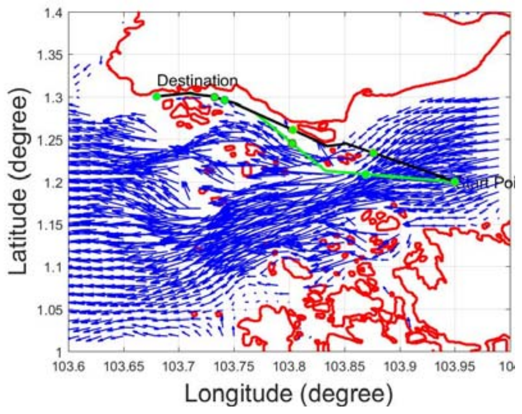


Fig. 5. 24. GA energy efficient path and the fourth initial energy efficient path in the sea current of the first hour

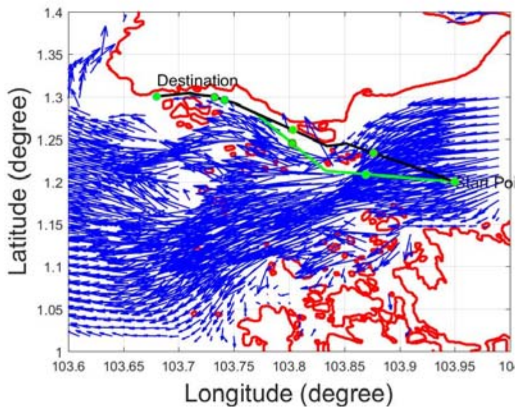


Fig. 5. 25. GA energy efficient path and the fourth initial energy efficient path in the sea current of the second hour

For demonstration, we combine the GA energy efficient path and the fourth initial energy efficient path in Fig. 5. 24 and Fig. 5. 25. In Fig. 5. 24, the GA path is represented by the green path and the fourth initial energy efficient path is represented by the black line. Both paths are divided by the one hour break points. The sea current states of the first hour and the second hour are represented by the blue arrows in Fig. 5. 24 and Fig. 5. 25. By comparison, it is illustrated that in the first two hours, the direction of the sea current is from east to west, the GA path enables the USV to take advantage of the favourable flow instead of crossing flow like the fourth initial path does, so GA path saves more energy than the fourth initial path in the first two hours.

As the first initial path and second initial path are similar, here we compare the GA path with the first initial path using the sea current data of the latter two hours. The GA path and the first initial path are compared in Fig. 5. 26 and Fig. 5. 27. The GA path is represented by the green line and the first initial path is represented by the black line. It can be seen, in the latter two hours, the direction of the sea current is from west to east, the GA path enables the USV travel to the north where the sea current is small instead of making the USV encounter counter-flow like the first initial path does.

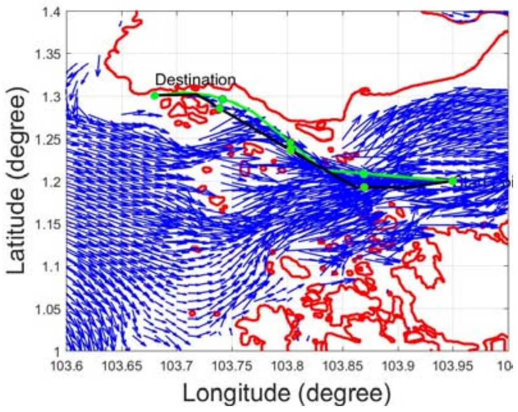


Fig. 5. 26. GA energy efficient path and the first initial energy efficient path in the sea current of the third hour

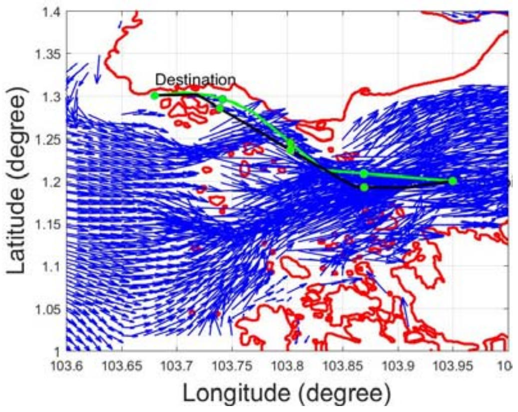


Fig. 5. 27. GA energy efficient path and the first initial energy efficient path in the sea current of the fourth hour

The energy consumption of the GA path and four initial paths during each hour of the Mission No.1 is shown in Table. 5. 3. It can be found that in the first two hours, the GA path consumes less energy than the third path and the fourth path, but consumes more energy than the first path and the second path. In the latter two hours, the GA path consumes more energy than the third path and the fourth path, but consumes less energy than the first path and the second path. Finally, it turns out the GA path requires the minimum energy in the entire mission duration and saves 10.27% energy than the second path, which is the most energy efficient one in the four initial paths.

Table. 5. 3. The energy consumption of the GA path and four initial paths during each hour in Mission No.1

	First hour	Second hour	Third hour	Fourth hour	Total
First path	0.3023	0.2347	0.7221	0.4690	1.7281
Second path	0.3129	0.1993	0.7153	0.4957	1.7231
Third path	0.3874	0.4789	0.5193	0.3440	1.7296
Fourth Path	0.3874	0.4789	0.5193	0.3440	1.7296
GA path	0.3312	0.2261	0.5848	0.4041	1.5462

Another two missions are simulated to test the performance of the proposed algorithm. In the Mission No.2, the USV is commanded to travel from (103.95, 1.2) to (103.7, 1.23). In the Mission No.3, the USV is commanded to travel from (103.95, 1.15) to (103.7, 1.25). The sea current data is the same with the data we used in the Mission No.1. In both missions, the USV should travel at 2.5 m/s speed. By calculating the first initial path, we can estimate both of these two missions can be finished in 4 hours, therefore, four initial paths are generated by applying the sea current data of four time intervals. In Fig. 5. 28 and Fig. 5. 29, the blue solid line, blue '--' line, black solid line and black '--' line represent the first initial path, the second initial path, the third initial path and the fourth initial path, respectively. The final GA paths are represented by the green solid line. The energy cost of each hour of four initial paths and the GA path are shown in Table. 5. 4 and Table. 5. 5. In Mission No.2, the GA path saves 27.68% of the total energy than the second initial path. In the Mission No.3, the GA path saves 20.18% of the total energy than the second initial path, which is the most energy efficient path generated by using Voronoi-Visibility energy efficient path planning algorithm. Therefore, we can conclude that our proposed path planning algorithm improves the performance of the Voronoi-Visibility energy efficient path planning algorithm in addressing the temporally variant sea current problem.

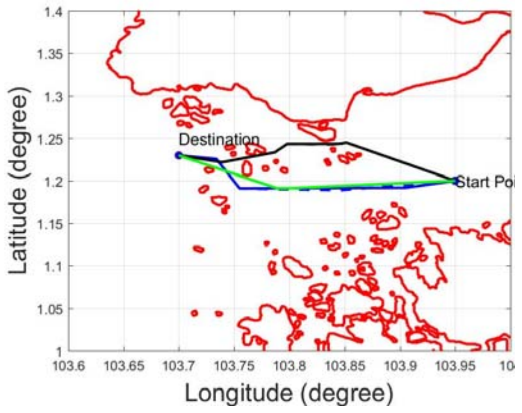


Fig. 5. 28. Final energy efficient path vs four initial energy efficient paths in Mission No.2

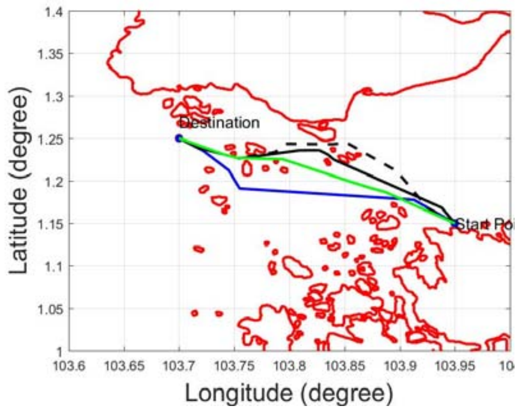


Fig. 5. 29. Final energy efficient path vs four initial energy efficient paths in Mission No.3

Table. 5. 4. The energy consumption of the GA path and four initial paths during each hour in Mission No.2

	First hour	Second hour	Third hour	Fourth hour	Total
First path	0.3017	0.1448	0.8050	0.3683	1.6198
Second path	0.2975	0.1416	0.8037	0.3665	1.6093
Third path	0.3874	0.4643	0.5968	0.3038	1.7522
Fourth Path	0.3874	0.4644	0.5967	0.3043	1.7528
GA path	0.3083	0.1265	0.5448	0.1842	1.1639

Table. 5. 5. The energy consumption of the GA path and four initial paths during each hour in Mission No.3

	First hour	Second hour	Third hour	Fourth hour	Total
First path	0.3624	0.1550	0.8156	0.4565	1.7894
Second path	0.3624	0.1550	0.8156	0.4565	1.7894
Third path	0.3894	0.4195	0.5886	0.4055	1.8030
Fourth Path	0.4376	0.4663	0.5420	0.4924	1.9383
GA path	0.3802	0.2166	0.5903	0.2413	1.4283

5.6 Summary

In this research, we proposed a spatially-temporally energy efficient path planning algorithm by integrating the Voronoi roadmap and GA algorithm. The proposed algorithm improves the performance of the Voronoi-Visibility energy efficient path planning algorithm (Niu, et al., 2016) by taking account of the temporal variation of the sea current. The proposed algorithm keeps the flexibility and computational efficiency of the Voronoi roadmap in addressing the complex geographic environment problem and also take advantage of the computational efficiency of the GA algorithm in searching for the high dimensional dataset. The proposed algorithm and the Voronoi-Visibility energy efficient path planning algorithm have been compared in three mission scenarios in the spatially-temporally variant sea current state. It is demonstrated that the proposed algorithm can save more energy than the Voronoi-Visibility energy efficient path planning algorithm in addressing the spatially-temporally variant sea current problem and the amount of saving energy varies between 10.27% and 27.68% in three missions, while also remaining at a user-configurable safety distance away from the island.

6 Development of USV Collision Avoidance Algorithm

6.1 Abstract

In recent years, Unmanned Surface Vehicles (USVs) have grown in popularity, with an increasing number of applications in the civil, military and research domains. When the USVs are operating in the same area with other marine vessels, the collision avoidance algorithm is the key to keep the safety of both the ownship vehicle and the encountered vessels. This chapter presents the development and validation of the geometric collision avoidance algorithm based on the International Regulations for Preventing Collisions at Sea (COLREGS). The proposed algorithm allows the USV to avoid single and multiple dynamic and static vessels while obeying the COLREGS and keeping the USV a constant clearance distance from the collision risk. The proposed algorithm includes collision detection algorithm, decision-making, collision resolution algorithm and resolution guidance algorithm. The collision detection algorithm describes the process of identifying the intruders as a collision. The decision-making algorithm will take account of the COLREGS and the traffic information to make the decision between keep-way and give-way. If there is a collision risk and the USV should give-way, the collision resolution waypoint will be generated by the collision resolution algorithm. Then the resolution guidance algorithm will be implemented to ensure the safety of the USV while the USV is avoiding the intruders. In the end, the proposed algorithm was validated in the single intruder scenario, multiple dynamic intruders scenario and the multiple static-dynamic mixed intruders scenario.

6.2 Introduction

Unmanned Surface Vehicles (USVs) have attracted a huge amount of interest in recent years because of their ease of use and low operational costs. USVs are ideal for deployment in missions that are tedious or dangerous for humans. Developing a high-level autonomous system, which can operate in unpredictable or unstructured environment is still a challenging task, since it requires robust guidance and control strategies to ensure the safety of the ownship and also any encountered vessel. Fig. 6. 1 shows the collision of two ships, the collisions not only cause huge financial loss to the operating company but most importantly the risk created for the ship operating staff, see (BEN LINE AGENCIES, 2014). When the USVs are operating in the same area with manned vessels, the collision avoidance algorithms are the key to keep the safety of both the ownship vehicles and any encountered vehicles. Instead of developing new traffic rules for the unmanned systems, USVs are expected to obey the existing rules, see (Cockcroft & Lameijer, 2011).

Fig. 6. 1. Ships collision (BEN LINE AGENCIES, 2014)

In recent years, different approaches have been proposed for solving the collision avoidance problem. Collision avoidance using a potential field method has been investigated in (Khatib, 1987). In essence, this method uses an artificial potential field which governs vehicle kinematics, but cannot always guarantee that the relative distance is greater than a minimum safe separation distance due to the difficulty of predicting the minimum relative distance. A two-tiered approach consisting a deliberative or far-field model and a reactive or near-field model have been developed (Larson, et al., 2007). The USV was equipped with monocular vision, stereo vision, radar and AIS. The proposed algorithm was validated during sea trials and the USV demonstrated the ability to avoid dynamic and static obstacles. The work carried out by Almeida (Almeida, et al., 2009) used a radar to detect collisions. The proposed algorithm was integrated into an USV and tested in different weather conditions to investigate the impact on radar performance. The tests were performed under different weather conditions ranging from heavy rain to clear sky to assess the impact on radar performance. One collision avoidance strategy (Naeem, et al., 2012) integrated Line-of-Sight with the manual biasing scheme. A collision avoidance algorithm based on virtual target path following guidance algorithm was applied (Marco, et al., 2012) and this algorithm was developed for USV multi-agent frameworks. In this investigation, Marco et al focused on the cooperative guidance algorithm of multiple USVs. An autonomous motion planning method was proposed based on COLREGS (Zhuang, et al., 2011). Loe (Loe, 2008) reviewed several collision avoidance approaches including both local methods and global methods. Local methods include Potential Field, Vector Field (VF) and Dynamic Window approach (DW). The global methods include Rapidly-Exploring Random Tree (RRT), A* and Constrained Nonlinear Optimization.

In the LEMUSV project, the geometric collision avoidance algorithm based on COLREGS was proposed and the demonstration of avoiding single collision and multiple collisions was made under the assumption that:

- Vehicle dynamics are represented by point mass in Cartesian coordinates.
- Intruders are non-manoeuvring during the collision avoidance manoeuvre.
- The USV ownship can obtain the deterministic position and velocity vectors of other intruders by using the onboard sensors, communication data link or estimator.

This implies that ownship USV can predict the trajectories and future state information using the current position and velocity vector and their linear projections. Note that these assumptions are for ease of analysis and are not a restriction of the approach. This chapter is organized as follows: Section 6.3 presents the architecture of the collision avoidance algorithm. The collision detection algorithm, decision-making algorithm, collision resolution algorithm and resolution guidance algorithm are described in Section 6.4, Section 6.5, Section 6.6 and Section 6.7, respectively. The single collision risk scenarios were simulated in Section 6.8 and the multiple collision risks scenarios were simulated in Section 6.9. The summary is given in Section 6.10.

6.3 Architecture of the Collision Avoidance Algorithm

The architecture of the collision avoidance algorithm is shown in Fig. 6. 2. The proposed algorithm includes four parts: collision detection, decision making, collision resolution and resolution guidance.

When the USV is following the predefined path, it keeps detecting and tracking the nearby traffic information. Assume that in any case, the distance between the intruder and the USV cannot be smaller than d_{min}. To ensure the strict safety of the USV, each intruder is assumed to have a safety radius d_m, in which case $d_m > d_{min}$. In any case, if the Closest Approach Distance d_r between the USV and the intruders is smaller than d_m, it is considered to be a collision risk. The USV is required to avoid the collision in time n, which is a configurable value defined by the users. Time to Closest Point of Approach (TCPA) is denoted by t_{tcpa}. The relations between n and t_{tcpa} are defined by the status flags, which defines the emergent level of the traffic. The decision-making system is defined according to the status flags and COLREGS to decide whether to keep on current heading or give way if there is a collision risk.

If the USV makes the decision to alter the course angle, the collision resolution waypoint will be calculated and the Proportional Navigation (PN) guidance algorithm will guide the USV to the new collision resolution waypoint. Once the collision risk is avoided, a new path to the destination will be plotted and the USV will follow the new path and check if there is any other potential collision.

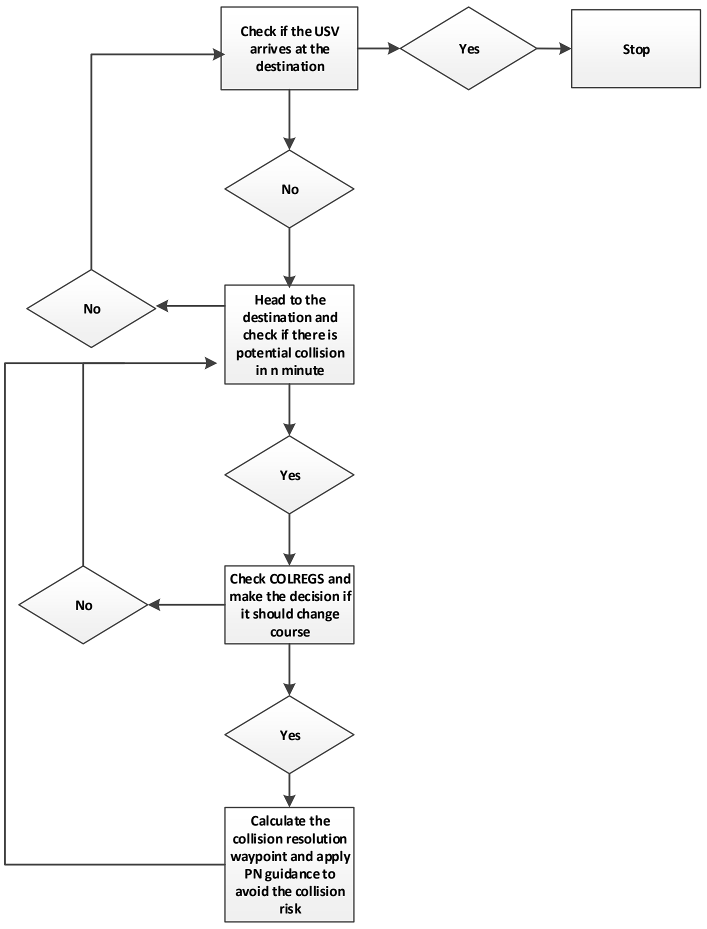

Check if the USV arrives at the destination
Yes
Stop
No
No
Head to the destination and check if there is potential collision in n minute
Yes
No
Check COLREGS and make the decision if it should change course
Yes
Calculate the collision resolution waypoint and apply PN guidance to avoid the collision risk

Fig. 6. 2. Collision avoidance algorithm logic

6.4 Collision Detection

The collision detection algorithm is based on the concept of collision geometry. If the distance between the USV with other vessels is smaller than the minimum separation distance d_m within a specific time, it is defined as a collision risk. The definition of collision detection algorithm is introduced by using the same concepts presented in (Dowek & Munoz, 2007) (Galdino, et al., 2007), such as the Closest Approach Distance (CAD), Closest Point Approach (CPA) and Time to Closest Point of Approach (TCPA).

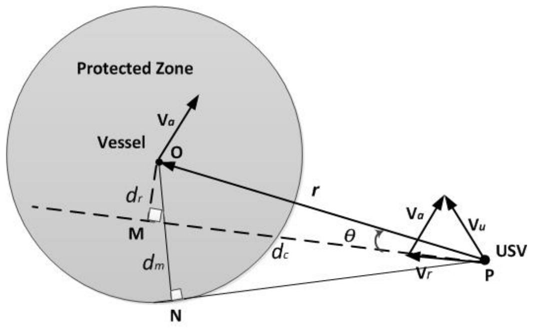


Fig. 6. 3. Collision detection

The USV speed $\overrightarrow{V_u}$, the vessel speed $\overrightarrow{V_a}$ and the relative speed $\overrightarrow{V_r}$ form the velocity triangle as illustrated in Fig. 6. 3. The relative speed can be calculated using (6. 1).

$$\overrightarrow{V_r} = \overrightarrow{V_u} - \overrightarrow{V_a} \tag{6. 1}$$

The angle between $\overrightarrow{V_r}$ and the line-of-sight from the USV to vessel is denoted by θ. The closest approach distance d_r and the distance r between the vessel and the USV satisfies (6. 2).

$$d_r = r \times \sin(\theta) \tag{6. 2}$$

When $d_r \le d_m$, a potential conflict is assumed to exist. The distance between USV and the Closest Point of Approach M is denoted by d_c, the Time to Closest Point of Approach is denoted by t_{tcpa}. t_{tcpa} can be calculated using (6. 3).

$$t_{tcpa} = \frac{d_c}{V_r} \tag{6. 3}$$

6.5 Decision Making

Status flags are used to define the danger level of the detected intruders based on t_{tcpa}. The status flags and COLREGS are applied together to define whether the USV should give way or stay on course (keep way). The descriptions of status flags and COLREGS are given as below.

6.5.1 Status Flag

To describe the USV status in different traffic situations, four collision status flags are defined:

Status flag 0: $t_{tcpa} > 2n\ minutes$, means there is no potential collision. Note that n is a time constraint defined by the user.

Status flag 1: $n < t_{tcpa} \leq 2n\ minutes$, means caution.

Status flag 2: $t_{tcpa} \leq n\ minutes$, means danger. The USV requires to analyse the COLREGS and makes the decision whether alter course or stay on course (keep way).

Status flag 3: $t_{tcpa} \leq n\ minutes$, the decision of the USV is to alter course based on COLREGS; The collision resolution and resolution guidance algorithm will be triggered to navigate the USV and avoid the collision.

6.5.2 COLREGS

The COLREGS is the abbreviation of International Regulations for Preventing Collisions at Sea. According to the COLREGS (Cockcroft & Lameijer, 2011), Head-on situation, Crossing situation, and Overtaking situation are defined to describe the situation of traffic.

6.5.2.1 Head-on Situation

According to Rule 14, when two power-driven vessels are meeting head-on both must change course to starboard so that they pass on the port side of the other. In Fig. 6. 4, both vessel A and vessel B should turn to starboard to give way to each other.

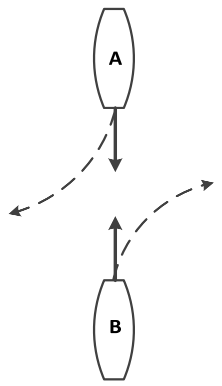


Fig. 6. 4. USV head-on situation

6.5.2.2 Crossing Situation

According to Rule 15, when two power-driven vessels are crossing, the vessel which has the other on the starboard side must give way and avoid crossing ahead of her. In Fig. 6. 5, the vessel A should give way to vessel B and vessel B should keep its way.

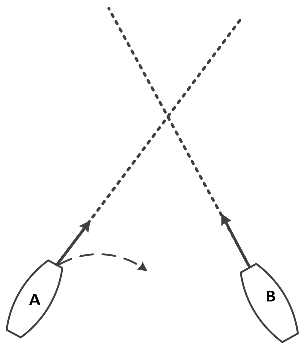


Fig. 6. 5. USV crossing situation

6.5.2.3 Overtaking Situation

According to Rule 13, an overtaking vessel must keep out of the way of the vessel being overtaken. Overtaking means approaching another vessel at more than 22.5 degrees abaft her beam. In Fig. 6. 6, vessel B is overtaking the way of vessel A and vessel B should give way to vessel A. Although COLREGS does not specify which side of the boat it must overtake, common practice on the water dictates that the overtaking boat should pass on the starboard side of the traffic boat (Kuwata, et al., 2011) (Kuwata, et al., 2014).

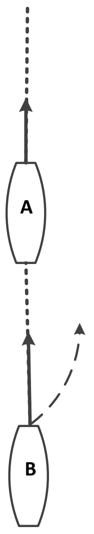


Fig. 6. 6. USV overtaking situation

6.5.3 Collision Avoidance Metrics

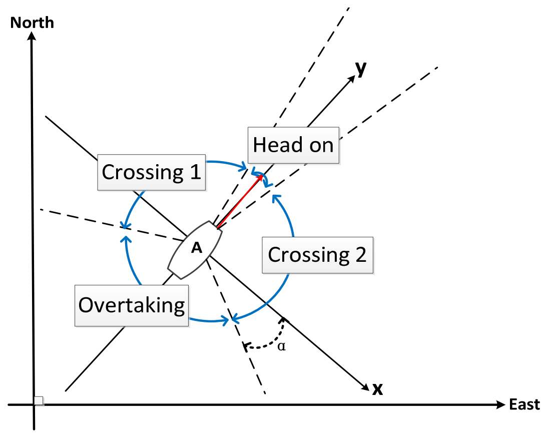


Fig. 6. 7. Collision avoidance metrics

According to the COLREGS, when the USV and the intruder have a potential collision, the USV will either give way to the intruder or keep way to wait for the intruder to alter course.

In the Fig. 6. 7, assume the USV is denoted by the vessel A and the vessel A is heading to the y direction. The relative position of the intruder with respect to vessel A can be divided into the head-on area, crossing 1 area, crossing 2 area and overtaking area. According to the COLREGS, $\alpha = 22.5^o$. The head-on angle range is assumed to be 20^0. When the vessel A is on a potential collision with the intruder, only when the intruder is in the head-on area and the crossing 2 area, the vessel A should alter course. Otherwise, vessel A should keep way. The altering course is realized by applying the collision resolution algorithm and the resolution guidance algorithm.

6.6 Collision Resolution

The proposed collision resolution algorithm can generate the collision resolution waypoint for one collision situation or multiple collisions risks.

6.6.1 Single Collision Scenario

When the USV detects that a potential collision will occur in n minutes and the USV is required to turn to starboard side to avoid the potential collision in accordance with COLREGS, the port side tangent point of the safety circle of the intruder vessel will be chosen as the collision resolution point. As shown in Fig. 6. 8, the USV is denoted by the vessel A and the intruder is represented by the vessel B, from A to the safety zone of vessel B there are two tangent points M and N, because N is on the port side of B, tangent point N will be chosen as the collision resolution waypoint.

The algorithm for calculating the position of the tangent point is given as follows. The positions of A and B are denoted by (x_A, y_A) and (x_B, y_B). The safety radius is represented by d_m. The positions of two tangent points are denoted by (x_M, y_M) and (x_N, y_N).

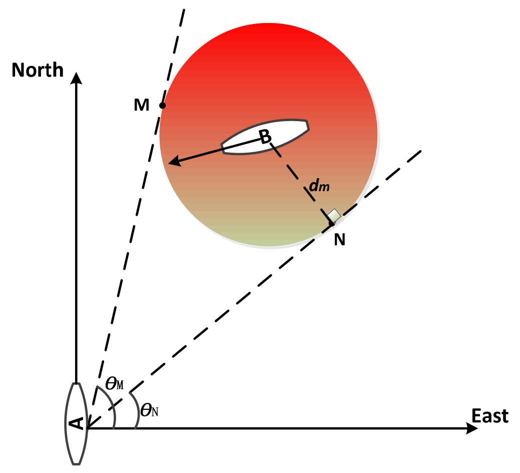


Fig. 6. 8. Collision Resolution (One Collision Situation)

The tangent point position (x, y) can be derived from (6. 4) and (6. 5).

$$(x - x_B)^2 + (y - y_B)^2 = {d_m}^2 \tag{6. 4}$$

$$(y - y_B) \times (y - y_A) = -(x - x_B) \times (x - x_A) \tag{6. 5}$$

Since these two equations will give us two solutions (x_M, y_M) and (x_N, y_N). Hence an approach for calculating the starboard side waypoint (x_s, y_s) is needed. The approach is given in (6. 6) and (6. 7).

$$\Delta\theta = \theta_M - \theta_N \tag{6. 6}$$

If $\Delta\theta$ satisfies the condition in (6. 7), (x_N, y_N) is the starboard side tangent point, otherwise, (x_M, y_M) is the starboard side tangent point.

$$0 < \Delta\theta < \pi \; or \; \Delta\theta < -\pi \tag{6. 7}$$

6.6.2 Multiple Collisions Scenario

In Fig. 6. 9, assume vessel A is the USV and vessel B, vessel C and vessel D are the intruders. Only when the intruders are in the head-on area or the starboard side crossing area, the ownship USV should turn to the starboard side. The boundary of the head-on area is denoted by AP and the boundary of the crossing area is denoted by AR.

According to the previous analysis, only the tangent point that is on the port side of the intruder safety circle will be selected as the collision resolution waypoint. The port side tangent points of the safety circles of the vessel B, vessel C, and vessel D are denoted by S, M and N, respectively. Taking into consideration the relative speed between the USV and the tangent point, we can use the method presented in (Shin, et al., 2008) to calculate the required heading angles of the USV to head to the point S, M and N, which are denoted by θ_S, θ_M and θ_N, respectively.

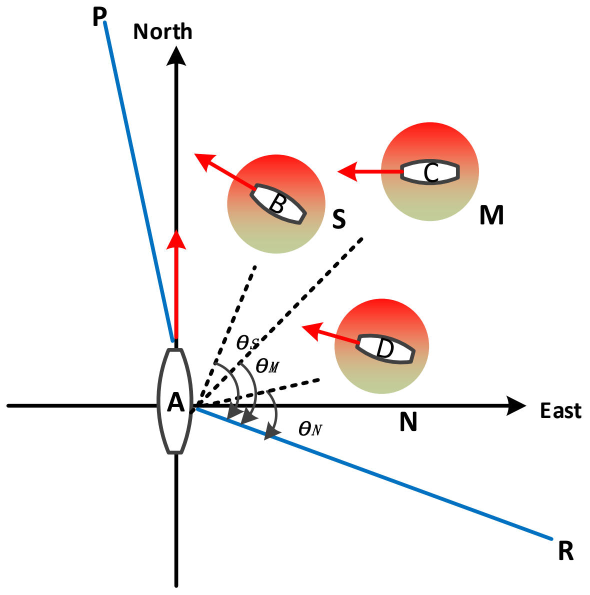


Fig. 6. 9. Collision Resolution (Multiple Collisions Resolution)

When USV encounters multiple potential collisions, multiple collision resolution waypoints will be generated. Thus, the approach of selecting one collision resolution waypoint to avoid all the intruders is needed. The approach is given as below:

$$\theta_X = \min\{\theta_S, \theta_M, \theta_N\}$$

In this case, $\theta_X = \theta_N$. The corresponding tangent point of safety circle D, which is N, is chosen as the collision resolution waypoint.

6.7 Resolution Guidance

Once the collision resolution waypoint is calculated, the Proportional Navigation (PN) waypoint guidance algorithm will be implemented to navigate the USV to travel to the collision resolution waypoint.

In Fig. 6. 10, assuming vessel A is the ownship USV and vessel B is the intruder, the collision resolution waypoint is the tangent point S. Point S is traveling at the same speed as vessel B, which is $\overrightarrow{V_B}$, while vessel A is travelling at $\overrightarrow{V_A}$ and the lateral acceleration is $\vec{a}$. The line-of-sight angle between A and tangent point S is denoted by θ_r and the changing rate is denoted by $\dot{\theta}_r$.

By changing the lateral acceleration of the ownship USV, we can change the speed direction of USV and navigate it to arrive at resolution waypoint S. The lateral acceleration can be calculated by (6. 8).

$$a = N \times \dot{\theta}_r \times V_r \tag{6. 8}$$

In the equation, *N* is a proportional constant that requires being regulated in the implementation. V_r is the relative speed between USV and vessel B. The angular velocity of USV $\dot{\theta}_A$ can be derived by (6. 9).

$$\dot{\theta_A} = \frac{a}{V_A} \tag{6. 9}$$

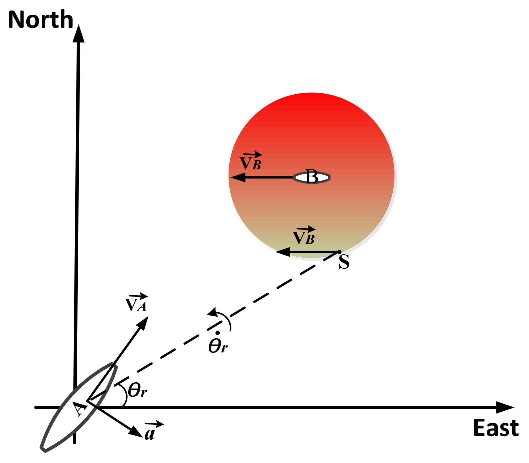


Fig. 6. 10. Proportional Navigation

Besides, to avoid chattering between the original waypoint guidance and collision resolution guidance, a decision-making algorithm is applied. When the USV is avoiding the intruders, the USV can check whether it will have any collision if it alters its course angle to be the LOS of USV and the destination. If there is no collision, the USV can travel back to the original destination.

6.8 Single Collision Avoidance Scenario Simulation

In this section, three scenarios were simulated, including head-on, crossing and overtaking. The kinematic of the USV is expressed as:

$$\dot{\eta} = R(\psi)\nu \tag{6. 10}$$

Where $\eta = [x, y, \psi\,]^T$ denotes the vector of position and orientation with coordinates in the earth-fixed reference frame, $\nu = [u, v, r\,]^T$ denotes the vector of linear and angular velocity with coordinates in the body-fixed reference frame, $R(\psi) = \begin{bmatrix} cos\psi & -\,sin\psi & 0 \\ sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$ denotes the transformation matrix between the earth-fixed reference frame and body-fixed reference frame. (x, y) and ψ are the position and orientation (yaw/heading angle) of the USV in the earth fixed reference frame. While u, v and r represent the velocity of surge, sway and yaw in the body-fixed reference frame, respectively.

6.8.1 Head-on Scenario

In this scenario, a head-on potential collision was simulated. The USV started from (0, -500) and aimed to travel to destination (0, 1000). The USV initial heading and speed is given as follows:

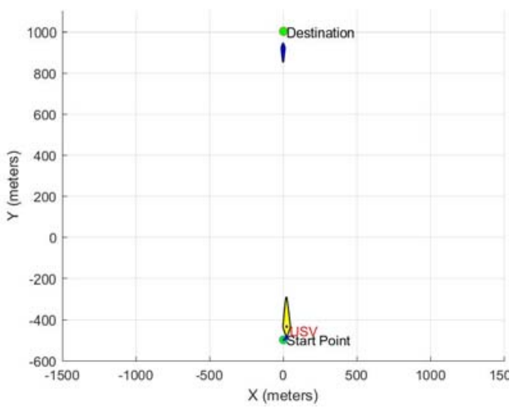
$$\begin{cases} USV_{\psi} = 0 \\ USV_{v} = 4 \end{cases}$$

The heading unit is radians, and the speed unit is meters per second. The head-on intruder was assumed to keep constant speed and constant heading. The intruder starting point, heading and speed is given as below:

$$\begin{cases} intruder(1)_x = 0 \\ intruder(1)_y = 1000 \\ intruder(1)_v = 4 \\ intruder(1)_{\psi} = 1.5\pi \end{cases}$$

The distance between the USV and the intruder should be controlled to be greater than 100 meters strictly, in this case, the safety radius was assumed to be 102 meters. It is configured such that if there is a potential collision in 1-minute time frame, the USV should take action to avoid the collision according to the COLREGS. The proportional constant N of PN guidance law was set to be 6.

The USV and the intruder started their missions, as shown in Fig. 6. 11. The USV was marked as yellow kayak shape and the intruder are marked as blue kayak shape. The USV starting point and destination were marked as green dots. In Fig. 6. 12, the USV detected the potential collision. According to COLREGS, the USV made the decision to turn right and then the collision resolution algorithm was triggered.



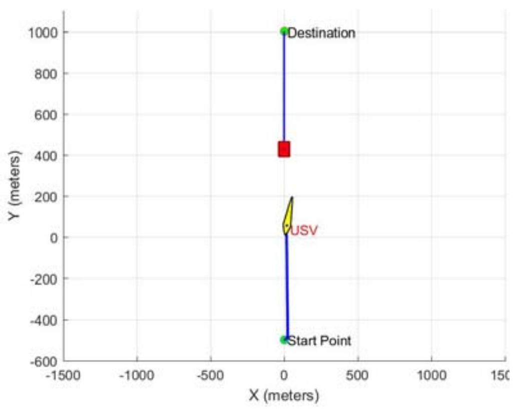
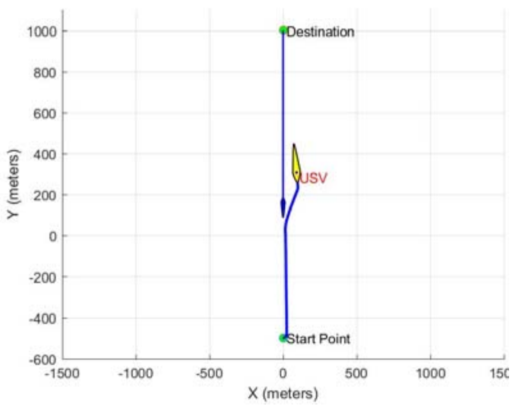


Fig. 6. 11. Head-on scenario (t = 20s) **Fig. 6. 12. Head-on scenario (t = 143s)**

In Fig. 6. 12, when the intruder was detected as potential collision risk, the intruder was marked as red square. The USV would keep turning to starboard side until the potential collision has passed. Once the collision is avoided, the USV head back to the destination, as shown in Fig. 6. 13. After the USV arrives at the destination, the USV will stop and the mission is completed. The final path of head-on scenario is shown in Fig. 6. 14.



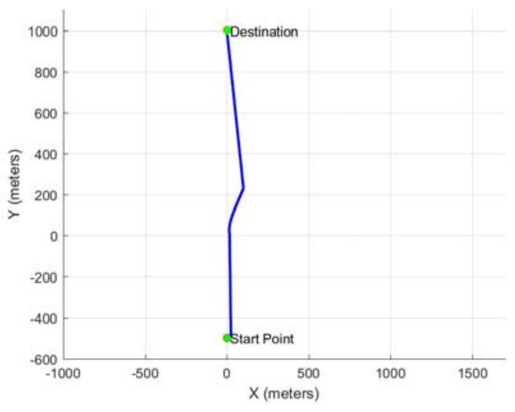


Fig. 6. 13. Head-on scenario (t = 210s) **Fig. 6. 14. Final path of head-on scenario**

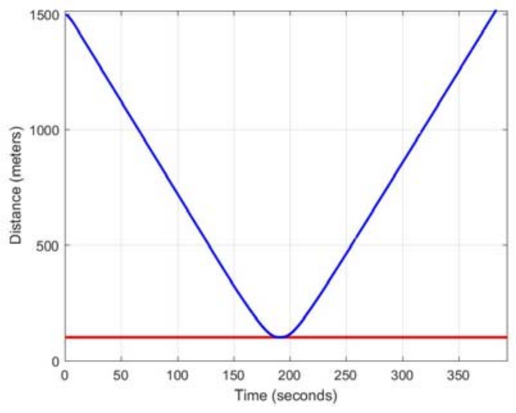


Fig. 6. 15. Distance between USV and intruder (Head-on scenario)

From the moment the intruder was detected until the intruder was passed, the distance between the USV and the intruder was recorded to demonstrate the performance of the proposed collision avoidance algorithm. As can be seen in Fig. 6. 15, the distance between the ownship and the intruder was kept greater than 100 meters, which will ensure the safety of the USV.

6.8.2 Crossing Scenario

In this section, a crossing collision scenario was simulated.

The USV started from (0, -500) and the destination was (0, 1000). The USV initial heading and speed is given as below:

$$\begin{cases} USV_{\psi} = 0 \\ USV_{v} = 4 \end{cases}$$

The intruder initial traffic information is given as below:

$$\begin{cases} intruder(1)_x = 700 \\ intruder(1)_y = 200 \\ intruder(1)_v = 4 \\ intruder(1)_{\psi} = \pi \end{cases}$$

The initial position and state of the USV and the intruder are shown in Fig. 6. 16. In Fig. 6. 17, when t = 120s, the intruder was detected as a potential collision.

According to the COLREGS, the USV should turn to its starboard side. As shown in Fig. 6. 17, the USV altered its original course and turned to the starboard side to avoid the intruder.

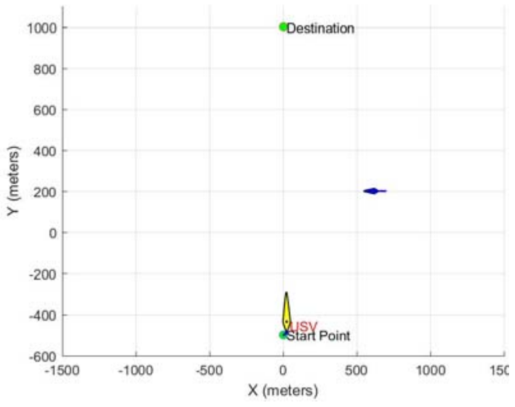


Fig. 6. 16. Crossing scenario (t = 20s)

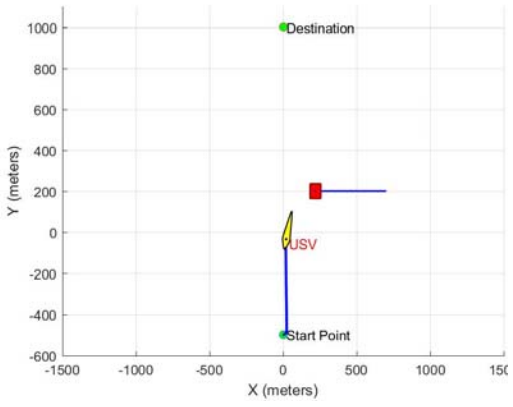


Fig. 6. 17. Crossing scenario (t = 120s)

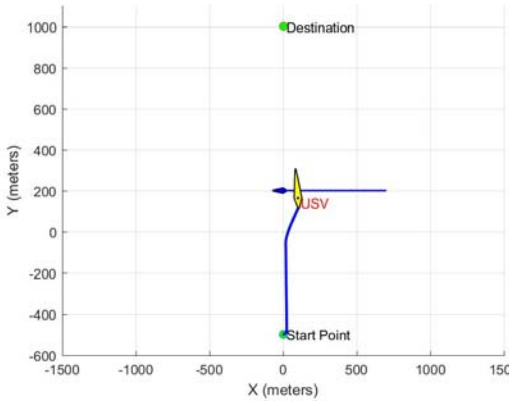


Fig. 6. 18. Crossing scenario (t = 175s)

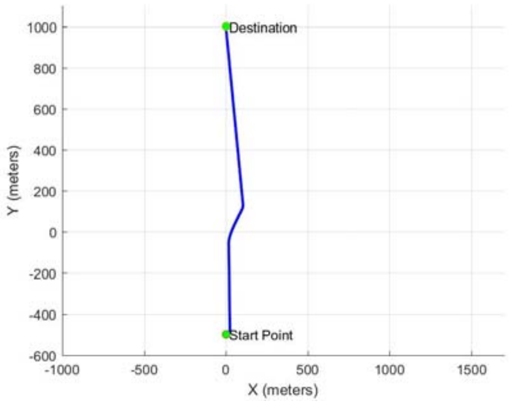


Fig. 6. 19. Final path of crossing scenario

After the USV avoided the intruder, the USV will head back to the destination, as shown in Fig. 6. 18. The final path of the crossing scenario is shown in Fig. 6. 19. The distance between the intruder and the USV was recorded and shown in Fig. 6. 20. It can be seen that the distance was kept greater than 100 meters, which proved the performance of the proposed collision avoidance algorithm in the crossing scenario to maintain a minimum safety distance from the intruder during the complete resolution maneuver.

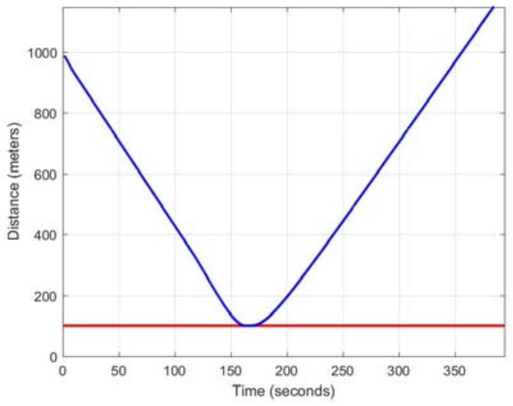


Fig. 6. 20. Distance between USV and intruder (Crossing scenario)

6.8.3 Overtaking Scenario

In this section, an overtaking scenario is simulated.

The USV started from (0, -500) and the destination was (0, 1000). The USV initial heading and speed is given as below:

$$\begin{cases} USV_\psi = 0 \\ USV_v = 4 \end{cases}$$

The intruder initial traffic information is given as below:

$$\begin{cases} intruder(1)_x = 0 \\ intruder(1)_y = 0 \\ intruder(1)_v = 1 \\ intruder(1)_\psi = 0.5\pi \end{cases}$$

As shown in Fig. 6. 21, the USV was heading to 0.5π direction and the intruder had the same course angle. However, the intruder has a lower speed in comparison with the USV, which results in the overtaking potential collision risk. In Fig. 6. 22, the USV detected the potential collision in the 1-minute time frame and started to turn to starboard side to avoid the collision.

In Fig. 6. 23, once the USV avoided the intruder, it headed back to the destination. The final path of overtaking scenario is shown in Fig. 6. 24. The safe performance of the proposed algorithm in overtaking scenario is shown in Fig. 6. 25, which illustrate that the distance between the USV and the intruder can be controlled to be greater than 100 meters during the complete maneuver.

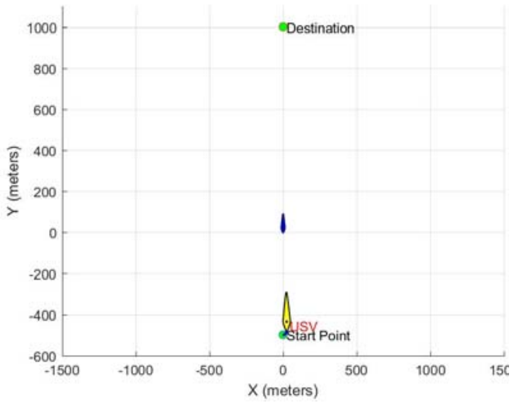


Fig. 6. 21. Overtaking scenario (t = 20s)

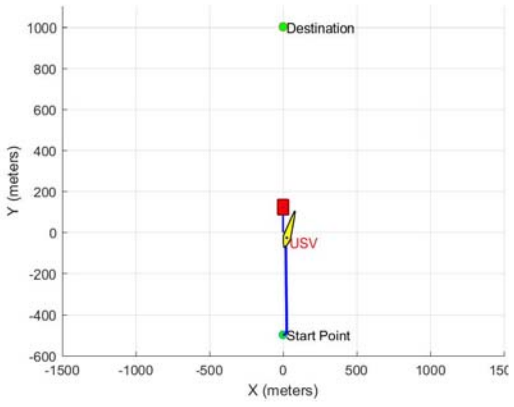


Fig. 6. 22. Overtaking scenario (t = 122s)

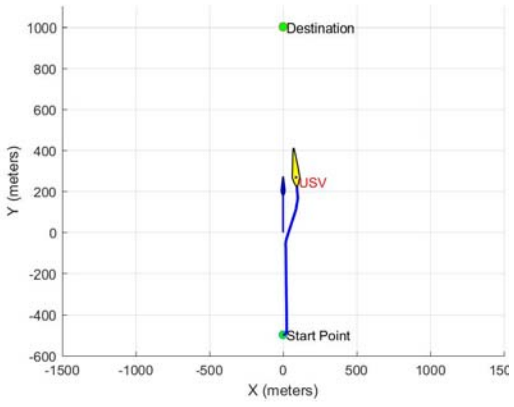


Fig. 6. 23. Overtaking scenario (t = 200s)

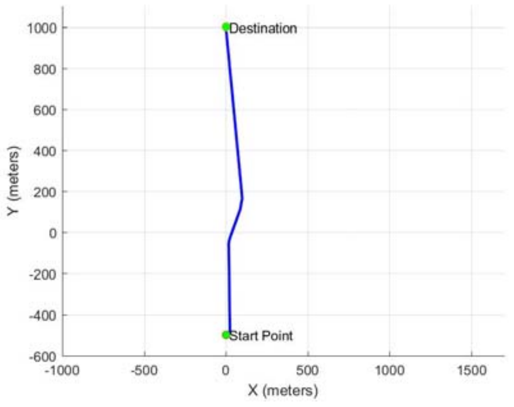


Fig. 6. 24. Final path of overtaking scenario

In conclusion, the proposed algorithm can ensure the safety of the USV in head-on, crossing and overtaking scenarios. The distance between the USV and the intruder can be greater than 100 meters during the complete maneuver.

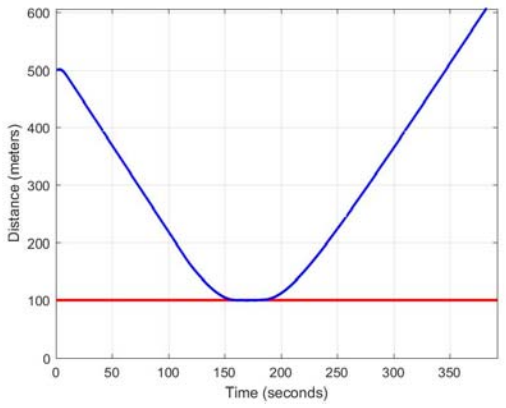


Fig. 6. 25. Distance between USV and intruder (Overtaking scenario)

6.9 Multiple Collision Avoidance Scenario Simulation

In this section, two scenarios were simulated, namely the multiple dynamic intruders scenario and multiple static-dynamic mixed intruders scenario.

6.9.1 Multiple Dynamic Intruders Scenario

In this scenario, six dynamic intruders were set up. The USV started from $(0, -500)$ with the aim to travel to destination $(0, 1000)$. The USV initial heading and speed is given below:

$$\begin{cases} USV_{\psi} = 0 \\ USV_{v} = 4 \end{cases}$$

The heading unit is radians, and the speed unit is meters per second. The six intruders were assumed to keep a constant speed and heading. The intruders' starting point, heading, and speed are given:

$$\begin{cases} intruder(1)_x = 1000 \\ intruder(1)_y = 300 \\ intruder(1)_v = 4 \\ intruder(1)_{\psi} = \pi \end{cases} \qquad \begin{cases} intruder(2)_x = -20 \\ intruder(2)_y = 450 \\ intruder(2)_v = 2 \\ intruder(2)_{\psi} = 1.5\pi \end{cases}$$

$$\begin{cases} intruder(3)_x = 1600 \\ intruder(3)_y = 700 \\ intruder(3)_v = 4 \\ intruder(3)_\psi = \pi \end{cases} \qquad \begin{cases} intruder(4)_x = 1200 \\ intruder(4)_y = 300 \\ intruder(4)_v = 4 \\ intruder(4)_\psi = \pi \end{cases}$$

$$\begin{cases} intruder(5)_x = 1000 \\ intruder(5)_y = 0 \\ intruder(5)_v = 2 \\ intruder(5)_\psi = \pi \end{cases} \qquad \begin{cases} intruder(6)_x = -800 \\ intruder(6)_y = -200 \\ intruder(6)_v = 2 \\ intruder(6)_\psi = 0 \end{cases}$$

The distance between the USV and the intruder should be controlled to be greater than 100 meters strictly, in this case, the safety radius was assumed to be 102 meters. It is assumed that if there is a potential collision in 1-minute time frame, the USV should take action to avoid the collision risk in accordance with the COLREGS. The proportional constant N of PN guidance law was set to be 6.

The ownship USV and the intruders started their missions as shown in Fig. 6. 26. The USV was marked as the large yellow triangle and the intruders were marked as small blue triangle. The USV starting point and final destination were marked as green dots. For clarity, the intruder numbers were also shown.

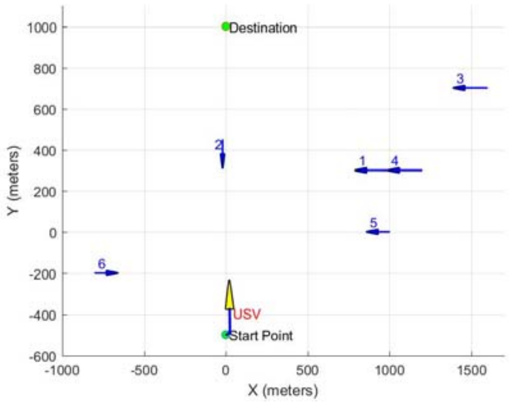


Fig. 6. 26. Multiple dynamic intruders scenario (t = 35s): the USV is marked as yellow triangle

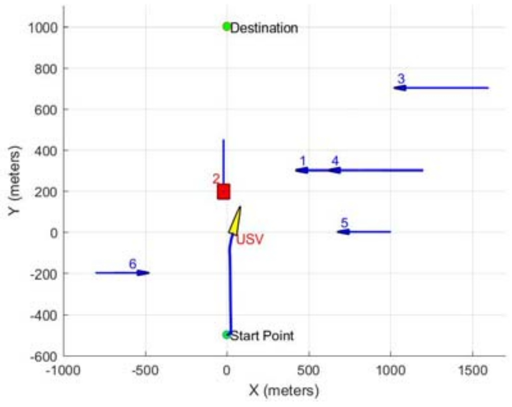


Fig. 6. 27. Multiple dynamic intruders scenario (t = 127s): the intruder being avoided is marked as red square

At 127 seconds, the USV encountered the intruder No.2 and calculated there is a potential collision risk and the traffic situation represents a head-on situation according to the COLREGS. Therefore, the USV altered course to the starboard side to avoid the collision, as shown in Fig. 6. 27. For clarity, the intruder that is being avoided, will be marked as red square.

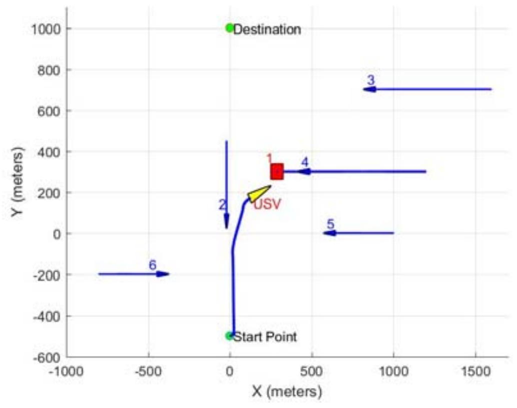


Fig. 6. 28. Multiple dynamic intruders scenario (t = 178s)

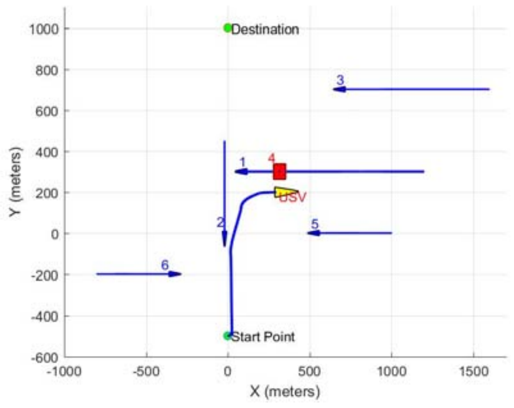


Fig. 6. 29. Multiple dynamic intruders scenario (t = 221s)

When the USV was avoiding the intruder No.2, it detected a potential collision with the intruder No.1 (crossing scenario). In this case, the multiple collision resolution algorithm will be triggered, and in this case the port side tangent point of the safety circle of intruder No.1 was chosen as the collision resolution waypoint, because when the intruder No.1 was avoided, the intruder No.2 will also be avoided. As shown in Fig. 6. 28, the intruder No.1 was marked as red square. However, due to the crowded situation, when the USV is avoiding the intruder No.1, it detected it will also have a collision risk with the intruder No.4 at the same time (crossing scenario) and it altered the course again as shown in Fig. 6. 29.

After intruder No.4 was avoided, the USV computed there was no collision in the next 1-minute time frame if it changes course to travel directly to the final destination waypoint. Therefore, it altered course to travel back to the final waypoint (destination) before it arrived at the destination, a new potential collision risk was calculated. The crossing intruder was intruder No.3 (crossing scenario) and the USV altered course once again to avoid the collision, as shown in Fig. 6. 30.

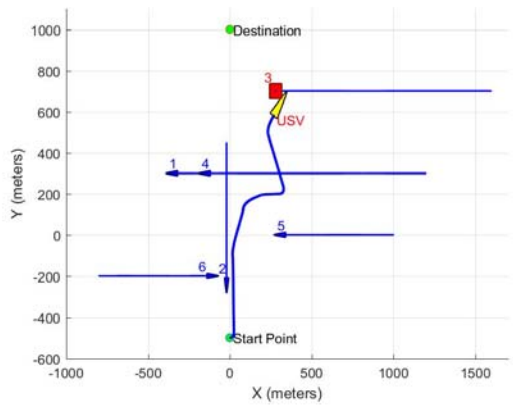


Fig. 6. 30. Multiple dynamic intruders scenario (t = 330s)

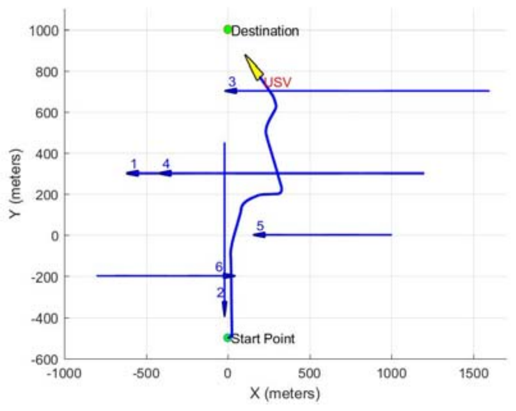


Fig. 6. 31. Multiple dynamic intruders scenario (t = 387s)

Once the intruder No.3 was avoided, the USV finally travelled to the destination, as shown in Fig. 6. 31. The Final path of the USV from the starting point to the destination is shown in Fig. 6. 32. For demonstrating the accuracy of the guidance algorithm, the distance between the USV and the intruder being avoided was also recorded, as shown in Fig. 6. 33.

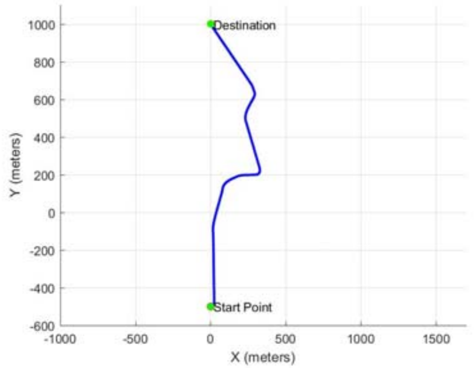


Fig. 6. 32. Final path of Multiple dynamic intruders scenario

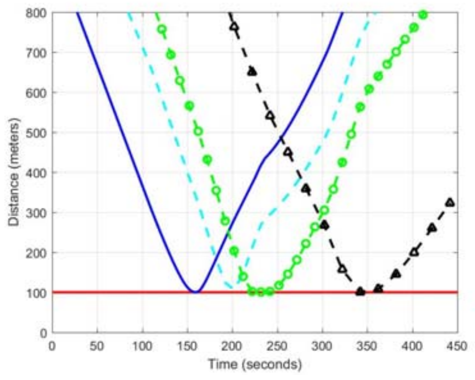


Fig. 6. 33. Distance between the USV and the intruders (The distances between the USV and the intruders No.2, No.1, No.4 and No.3 are represented by blue solid line, cyan dash line, green dash circle line and black dash triangle line, respectively)

The intruders that were avoided were No.2, No.1, No.4, and No.3, in time sequence, respectively. In Fig. 6. 33, the distances between the USV and the intruders No.2, No.1, No.4 and No.3 are represented by blue solid line, cyan dash line, green dash circle line and black dash triangle line, respectively. It can be seen that the distance between the USV and the intruder has been kept greater than 100 meters, which means the proposed algorithm can ensure the safety of the USV under multiple potential collisions scenarios while ensuring the USV obey the COLREGS.

6.9.2 Multiple Static-dynamic Mixed Intruders Scenario

In this scenario, multiple static intruders and multiple dynamic intruders were simulated to test the proposed algorithm, which is a more realistic situation of what might potentially be encountered in real life. The USV started from $(0, -500)$ and aimed to travel to destination $(0, 1000)$. The USV initial heading and speed is given as:

$$\begin{cases} USV_{\psi} = 0 \\ USV_{v} = 4 \end{cases}$$

Seven intruders initial traffic information is given as below:

$$\begin{cases} intruder(1)_x = 100 \\ intruder(1)_y = 400 \\ intruder(1)_v = 0 \\ intruder(1)_{\psi} = \pi \end{cases} \qquad \begin{cases} intruder(2)_x = -20 \\ intruder(2)_y = 0 \\ intruder(2)_v = 0 \\ intruder(2)_{\psi} = 1.5\pi \end{cases}$$

$$\begin{cases} intruder(3)_x = 1600 \\ intruder(3)_y = 700 \\ intruder(3)_v = 4 \\ intruder(3)_{\psi} = \pi \end{cases} \qquad \begin{cases} intruder(4)_x = 1200 \\ intruder(4)_y = 500 \\ intruder(4)_v = 4 \\ intruder(4)_{\psi} = \pi \end{cases}$$

$$\begin{cases} intruder(5)_x = 500 \\ intruder(5)_y = 0 \\ intruder(5)_v = 0 \\ intruder(5)_{\psi} = \pi \end{cases} \qquad \begin{cases} intruder(6)_x = -800 \\ intruder(6)_y = -200 \\ intruder(6)_v = 2 \\ intruder(6)_{\psi} = 0 \end{cases}$$

$$\begin{cases} intruder(7)_x = -1300 \\ intruder(7)_y = 500 \\ intruder(7)_v = 4 \\ intruder(7)_\psi = \pi \end{cases}$$

In this scenario, intruders No.1, No.2, and No.5 are the static vessels and the rest of the intruders are dynamic vessels. The dynamic vessels are represented by the small blue triangles and the static vessels are represented by the blue circles. In Fig. 6. 34, at 103 seconds, the USV detected the potential collision with static intruder No.2 and it altered course to avoid this collision. Once intruder No.2 was avoided, the USV detected potential collision with static intruder No.1, it altered course to avoid intruder No.1 and soon it detected another two potential collisions with intruders No.4 and No.7 (crossing scenarios). Therefore, the multiple collisions resolution algorithm was triggered and the tangent point of safety circle of intruder No.7 was chosen as the collision resolution waypoint. The moment when the USV was avoiding intruder No.1 is shown in Fig. 6. 35 and the instance, when the USV was avoiding intruder No.4 and No.7, is shown in Fig. 6. 36.

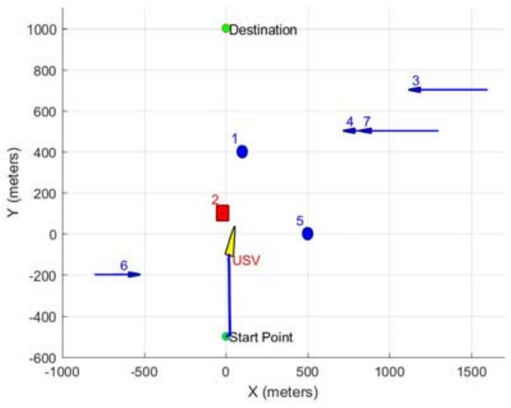


Fig. 6. 34. Multiple static-dynamic intruders scenario (t = 103s)

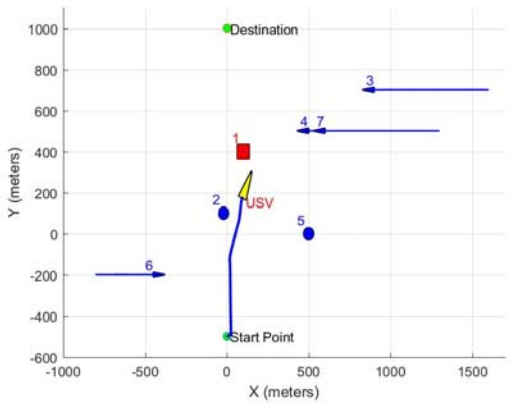


Fig. 6. 35. Multiple static-dynamic intruders scenario (t = 175s)

However, following the static intruder No.1 and two dynamic intruders (No.4 and No.7) were avoided, as the USV was planning to head back to the destination, a crossing potential collision with intruder No.3 was detected and the USV performed an emergent turn to avoid intruder No.3, as shown in Fig. 6. 37. The final path is shown in Fig. 6. 38 and the distance between the USV and the intruders is shown in Fig. 6. 39. It is worth noting that during this simulated scenario, not only the distance but also the collision risk within the 1-minute time frame was considered.

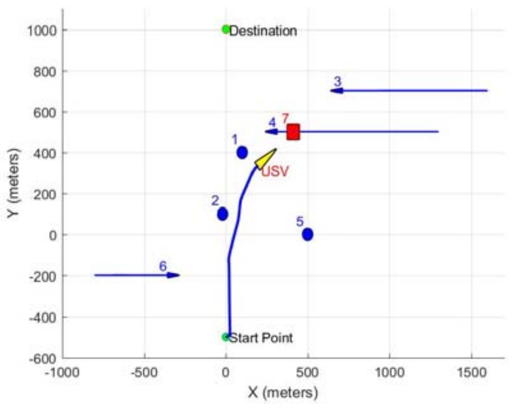


Fig. 6. 36. Multiple static-dynamic intruders scenario (t =222s)

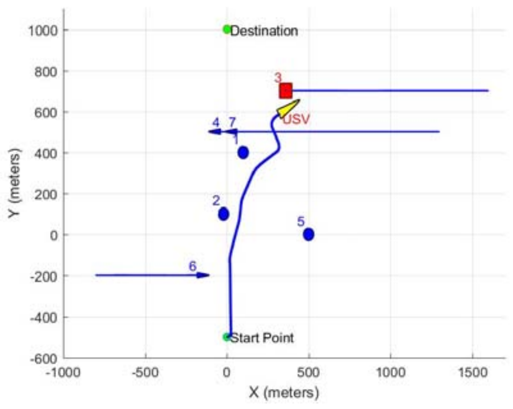


Fig. 6. 37. Multiple static-dynamic intruders scenario (t = 310s)

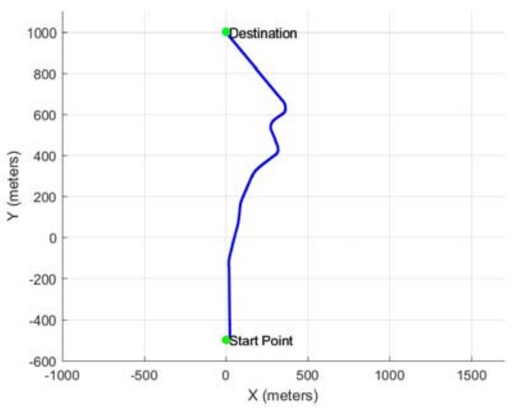


Fig. 6. 38. Final path of multiple static-dynamic intruders scenario

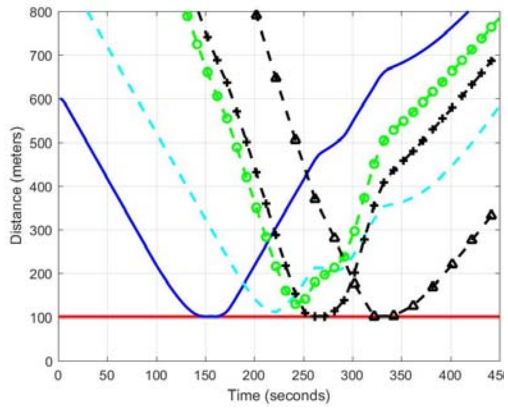


Fig. 6. 39. Distance between the USV and the intruders (The distances between the USV and the intruders No.2, No.1, No.4, No.7 and No.3 are represented by blue solid line, cyan dash line, green dash circle line, black dash plus line and black dash triangle line, respectively)

Fig. 6. 38 is a plot of the final path, where the time history of all the turn manoeuvres is shown, during which the intruders No.2, No.1, No.4, No.7 and No.3 were avoided in sequence. In Fig. 6. 39, the distances between the USV and the intruders No.2, No.1, No.4, No.7 and No.3 are represented by blue solid line, cyan dash line, green dash circle line, black dash plus line and black dash triangle line, respectively. The recorded distances between the USV and these intruders were kept greater than 100 meters, illustrating that the proposed algorithm can ensure the safety of the USV while obeying the COLREGS.

6.10 Summary

In this chapter, a geometric collision avoidance is proposed and validated in the complex multiple collisions scenarios. The proposed algorithm includes collision detection algorithm, decision-making algorithm, collision resolution algorithm and resolution guidance algorithm. The collision detection algorithm is applied for identifying the collision risk. The proposed algorithm takes account of the COLREGS by using the decision-making algorithm and the generated collision resolution obeys the regulations of the COLREGS. The resolution guidance algorithm ensures the USV a constant clearance from the potential collisions. Finally, the proposed algorithm was validated in the multiple dynamic intruders scenario and the multiple static-dynamic mixed intruders scenario.

7 Model Checking for Decision Making System of Long Endurance USV

7.1 Abstract

In this research, the feasibility of using model checking algorithm for verifying the USV decision-making system in dealing with multiple non-deterministic and concurrent environmental uncertainties has been demonstrated. Three USV case studies were implemented to verify the USV decision-making system. In the first scenario, multiple concurrent environmental uncertainties, including fault event, communication loss, and collision risk were modelled to verify the safety property of the decision-making system. The second scenario extended the first scenario by taking energy states into account. The second scenario was captured from the long endurance C-Enduro USV case, which utilizes solar panels, wind turbine and diesel generator as energy resources. The factors that affect the energy generation condition include the solar irradiance, wind condition and diesel generator state. The factors that affect the energy consumption include the USV behaviour and the sea current state. The interactive relations between these factors were modelled to verify the safety and long endurance properties of the USV decision-making system. In the third scenario, the long endurance USV decision-making system was improved by increasing the number of autonomous vehicles and building a mesh network. The purpose was to establish the fault tolerance capability and a reliable communication network. The final scenario dealt with the system comprised of a long endurance USV, a regular USV, a UAV, a GCS and a mesh network with safety, long endurance and fault-tolerant properties. In this research, the autonomous vehicles decision system, the GCS decision-making system, the wireless mesh network, the multiple environmental uncertainties and the states of the corresponding sensors were modelled using Kripke modelling method, the properties to be verified were expressed using CTL and finally the properties were verified using the model checker MCMAS. The verification results help retrospect the design of the system improved by considering additional environmental uncertainties, additional agents and behaviours during the three steps of scenario modification. This work also

represents the scalability and computational efficiency of the MCMAS which can verify the extended multi-agent system consisting of 35 Kripke models, including 15 communication-related models, 4 collision risk-related models, 7 fault event-related models, 5 energy-related models, and 4 autonomous systems-related models.

7.2 Model Checking Background

The collision avoidance algorithm (Savvaris, et al., 2014), path following algorithm (Niu, et al., 2016) and path planning algorithm (Niu, et al., 2016) are important parts of the Guidance, Navigation and Control (GNC) system, which play an important role in determining the level of USV autonomy. These algorithms should be considered as a submodule of the decision-making system for the USV. The correctness of the decision-making system for USVs is fundamentally critical. Simulation and formal verification are two common methods to verify the correctness of the decision-making system.

Simulation is commonly considered to be the most successful verification technique for analysing USV behaviour. In the simulation, every experiment is informative, and by applying statistical methods to a series of experiments, researchers may draw inferences concerning overall trends in behaviour. However, simulation alone is not sufficient to verify the requirements that are expressed in formal logic such as, “Will property P1 be true for all experiments?” or, “Is property P2 true whenever property P3 is true?”. There are two disadvantages of simulation which include the following: (1) Only a part of the available cases will be dealt with by software simulations as the analysis is not exhaustive; (2) The outcome is simple information which is infeasible to verify complex cases. The problem of analysing complex USV behaviour needs feasible approaches.

Formal verification is the state of the art technique in science and engineering to both detect errors and provide a high level of confidence in the correctness of a protocol, algorithm or system. Nowadays, formal verification may be an highly advantageous alternative technique to simulation. Formal verification is not only complete in logic and rigorous in mathematics, but flexible in the modelling and specification of complex behaviours. Formal verification includes deductive verification and model checking, which clarifies with a high degree of certainty that the system meets its requirements. Deductive verification is proof-based and is well recognized in computer science, significantly influencing software development. However, deductive verification is time-consuming and may be performed only by experts in logic and mathematics (Sirigineedi, et al., 2009). Model checking is a technique that may be used to analyse a system. Given a model of the system (Clarke, 1997), model checking allows for formal verification that the model satisfies a given property. Simulations or experiments may only test a subset of all possible execution states, while model checking may exhaustively and automatically check whether the model meets a given specification. Therefore, model checking may be applied to verify the properties of the system in the design phase to prevent the occurrence of errors. The system properties include reachability, safety, liveness and fairness properties. The details of these properties are follows:

- Reachability property means some particular property may be achieved.
- Safety property means that, under certain conditions, some event never occurs.
- Liveness property means that, under certain conditions, some event will ultimately occur.
- Fairness property means that, under certain conditions, some event will (or will not) occur infinitely often.

In order to apply model checking methods to a system at the design-level, the tasks to be performed are logically divided into three parts, which include the following:

- Modelling: In the first phase, the system design or mission scenario is converted to a formalism that is acceptable to the automated verification tools, known as model checkers. In some cases, the abstraction may be used to remove unimportant or unnecessary details of the system design.

- Specification: The next step is to state and express the necessary properties the system must satisfy. It is necessary to use a formalism that is acceptable to the model checker. In this phase, a temporal logic is usually used for hardware and software systems.
- Verification: The model checkers are used to verify the validity of the specification against the proposed model exhaustively. The most common mode of operation is to verify a system's state space for satisfaction of the specifications and to generate verification results. For negative results produced by the tool, the counter example has to be analysed and then the problem may be traced back either to the model or the wrong specification. After that, suitable amendment steps may be taken.

Formal modelling methods, formal specification (temporal logic) and model checkers are introduced in Section 7.2.1, Section 7.2.2 and Section 7.2.3, respectively. Section 7.2.4 presents a literature review of using the model checking method to verify autonomous systems.

7.2.1 Formal Modelling Methods

Formal modelling techniques originally evolved around the study of reactive systems. A reactive system is one that constantly maintains an interaction with, and is influenced by its environment. Therefore, its specification must consider its ongoing behaviours that change with time. This definition is closely relevant to moving robot applications, and many researchers have applied these techniques to study robot behaviours (Suresh, et al., 2006). To date, several formal modelling techniques such as Finite State Automata (FSA) (Béatrice, et al., 2001), Message Sequence Charts (MSCs) (Reniers, 1999), Petri nets (PNs) (Tadao, 1989), Kripke models and temporal logic have been proposed to specify, analyse, understand and verify the correctness of reactive systems.

FSA is the most common formal model and may be visualised as a state-transition graph. FSA may be defined as a model consisting of an initial state, a set of states, an input alphabet and a transition function that maps input symbols and current states to a next state. One state changes to new states depending on transition function. Although the FSA may express multiple-agent systems, the defined states need to be marked with channels and clocks in order to synchronize the whole system. However, the addition of channels and clocks makes automated systems complicated and large in size (Choi, 2012).

MSCs are a graphical and textual language developed for the specification and description of interactions between system components. Since the communication behaviour may be presented in a very intuitive and transparent manner, the MSC language is easy to learn, use and interpret. Therefore, the main application of MSCs is for the communication behaviour of real time systems. However, it is shown that MSCs are an incomplete specification method on their own (Ladkin & Leue, 1992). They need to be translated to automata and supplemented with explicit safety and liveness conditions in order to be a complete formal specification (Ladkin & Leue, 1992) (Leue, et al., 1998).

PNs is one of the mathematical modelling languages for the description of distributed systems. Petri nets is a directed bipartite graph consisting of places and transitions. Petri nets may be used to model process synchronisation, concurrent systems and conflicts. However, conventional Petri nets cannot model any form of uncertainty and needs to be fuzzy Petri nets. The reuse of Petri net models is also very restrictive and construction of Petri nets for large systems will scale up the model size considerably. When subjecting Petri nets for analysis, they need to be transformed into strengthened synchronised automata permitting the dynamic creation of parallel components (Suresh, et al., 2006).

Kripke model is a variation of the transition system, proposed by Saul Kripke in the 1950s and 1960s. The Kripke model formalism is widely used in highly complex and zero fault tolerance problems such as verification of real-time software and correctness of logic systems design (Huth & Ryan, 2004). A Kripke model is a graph with labelled nodes and edges. Such models provide semantics for various model logics including temporal logic, logic of program, logic of actions, and logic of obligation. All these logics have been studied intensively in philosophical and mathematical logic and in computer science, and have been applied increasingly in domains such as program semantics, artificial intelligence, and more recently in the semantic web (Gasquet, et al., 2014). The details of the Kripke model is given follows: Let AP be a set of atomic propositions, i.e. Boolean expressions over variables, constants and predicate symbols. In (Clarke, et al., 1999), a Kripke structure is defined over AP as a 4-tuple $M = (S, I, R, L)$ consisting of the following:

- A finite set of states S.
- A set of initial sets $I \subseteq S$.
- A transition relation $R \subseteq S \times S$.
- A labelling function $L: S \rightarrow 2^{AP}$ which is a function that labels each state with the set of atomic propositions true in that state.

For example, the Kripke model may be viewed as a directed graph as shown in Fig. 7. 1. Fig. 7. 1 illustrates a Kripke structure *M = (S, I, R, L),* where

- $S = \{S_1, S_2\}$ represents the states the system may stay at during operation.
- $I = \{S_1\}$ represents the initial state.
- $L = \{(S_1, \{p, q\}), (S_2, \{p\})\}$ represents the propositions those hold at corresponding states.
- $R = \{t_1, t_2\}$ represents the transition conditions between the states.

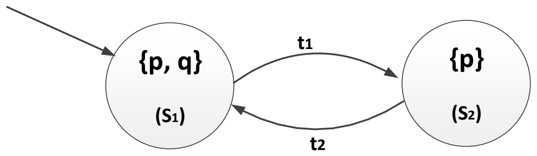


Fig. 7. 1. Graphical representation of Kripke model

The Kripke model is suitable for modelling unambiguous behaviours and representation of hybrid control approaches where discrete decision-making such as path following, collision avoidance, station keeping or increasing path following speed co-exist with reactive system operation. The main reasons for the success of the Kripke approach is its ability to represent real world uncertainty using a formal and intuitive model in the form of a directed graph. Although these models are abstract and simple, they are expressive enough to capture the dynamic and discrete temporal behaviours of the reactive systems.

7.2.2 Formal Specification

Temporal logic is usually used to specify the properties of the system. Temporal logic is a formalism for describing the sequences of state transitions in a reactive system. The “sequence of transition” is critical for real-time systems. As we want to capture the behaviour of the GCS and the USV over time, temporal logic is suitable for specifying requirements of the system. There are two fundamental types of temporal logic: Linear Temporal Logic (LTL) and Computational Tree Logic (CTL). LTL assumes a linear view of time. In LTL, the system state at any discrete point in time has a single successor state, depicted in Fig. 7. 2. CTL takes a branching view of time, that is, the system state at any discrete point in time may have many possible successors, all of which are reasoned about collectively. A CTL structure is shown in Fig. 7. 3.

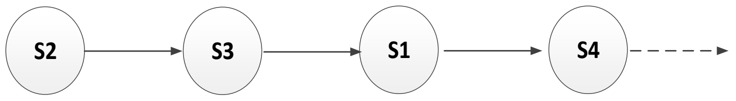


Fig. 7. 2. Representation for LTL

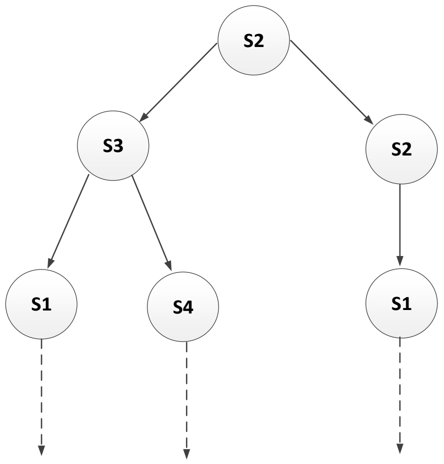


Fig. 7. 3. Representation for CTL

The LTL has four temporal operators which include: *X (next), G (globally), F (eventually),* and *U (until).* The CTL operators are *AX, EX, AG, EG, AU, EU, AF* and *EF*. These operators are pairwise operators. The first of the pair is either *A* or *E*. *A* means “for all paths” and *E* means “there exists a path”. Both A and E are quantifiers. The expressive power of CTL and LTL are not comparable because the properties expressed in the LTL cannot be expressed in the CTL, and vice versa. A third temporal logic is CTL*, which is a superset of CTL and LTL, combining path quantifiers and temporal operators freely.

7.2.3 Model Checker

Model checkers have been used for many years and may be used to deal with finite state concurrent systems, including verifying hardware and software design. The model checkers to be introduced in this section include the following: Simple Promela Interpreter (SPIN), Symbolic Model Verifier (SMV), Probabilistic Model Checker (PRISM), UPPAAL, KRONOS, Java Path Finder (JPF) and MCMAS.

SPIN and SMV are two common model checkers. SPIN is a popular open-source software verification tool developed at Bell Labs in 1980 that supports the LTL. The programming language of SPIN is called Process Meta Language (PROMELA) and the user needs to translate the model and specification into PROMELA. SMV, developed by K. L. McMillan at Carnegie Mellon University, performs (BDD-based) symbolic model checking of CTL. In the same way as SPIN, SMV translates the system model into its own input language.

PRISM is a probabilistic model checker which is a tool for formal modelling and analysis of systems that exhibit random or probabilistic behaviour. PRISM may build and analyse several kinds of probabilistic models including Discrete-Time Markov Chains (DTMCs), Continuous-Time Markov Chains (CTMCs), Markov Decision Processes (MDPs), Probabilistic Automata (PAs) and Probabilistic Timed Automata (PTAs).

UPPAAL is an integrated tool for modelling, validation and verification of real-time systems. UPPAAL may be used for automatic verification of safety and bounded liveness properties of real-time systems modelled as networks of timed automata. UPPAAL was developed in collaboration between Uppsala University in Sweden and Aalborg University in Denmark.

KRONOS (Nicollin, et al., 1992) is a model checker for Timed Computation Tree Logic (TCTL) properties of a timed (Buchi) automation. KRONOS is a tool assisting designers of real-time systems to develop projects meeting the specified requirements. The verification engine of KRONOS checks whether requirements are satisfied and provides diagnostic trials demonstrating why the property holds or does not hold.

JPF (Lerda & Visser, 2001) is a Virtual Machine (VM) for producing JAVA bytecode given to Java programs to execute. It is used to find defects in these programs. If the properties list is used as the input, JPF will output a report that states whether or not the properties hold.

MCMAS is an open-source model checker for the verification of Multi-Agent Systems (MAS) and uses its own language called Interpreted Systems Programming Language (ISPL). ISPL is an agent-based, modular language inspired by interpreted systems, a popular semantics in MAS. MCMAS supports CTL operators and basic fairness conditions. When the properties are violated, MCMAS will generate the counterexamples and witness executions for a wider range of specifications. MCMAS runs on multiple operating systems including Linux, Mac and Windows.

7.2.4 Literature Review

NASA has been developing formal verification techniques for multiple rovers and satellites (Brat & Jonsson, 2005) (Pecheur, 2000). In (Michael, et al., 2004), timed automata were applied to model multiple robot system, the properties expressed in Computational Tree Logic (CTL) and finally verified by the Uppaal model checker. The Kripke model of a single UAV performing a search mission was modelled, and the properties expressed in CTL verified using SMV in (Sirigineedi, et al., 2009); subsequently, the scenario was extended to a multiple UAV searching scenario in (Sirigineedi,, et al., 2011). A multiple UAV system monitoring road network was modelled and verified in (Sirigineedi, et al., 2010). A group of robots operating with minimalist communication and having no priori knowledge of the environment has been modelled using a Kripke model in (Suresh, et al., 2006), the desirable properties of co-operation were expressed using LTL and the properties were finally verified using SPIN. The integration of model checking techniques into human-automation mission planning systems was proposed in (Humphrey & Patzek, 2013) and enables the autonomy to make decisions based on the human’s intent and provides better feedback to the human when problems arise. Another USV mission plan verification for a “VIP escort” mission was presented in (Humphrey, 2012) and in this scenario, multiple UAVs monitor and safely guide a ground-based “VIP” vehicle across a road network. The model was built using PROMELA, the properties were expressed using LTL and verified using SPIN.

The model checker MCMAS is one of the more recently developed model checkers that may reason about time, knowledge and correct behaviours of agents. In (Molnar & Veres, 2009), MCMAS is used to verify the behaviours of the autonomous underwater vehicles. Kripke model and MCMAS are applied to verify the decision-making system of the team-level autonomous system (Choi, et al., 2010) (Choi, 2012) and the final scenario of (Choi, et al., 2015) comprises a ground control system, two unmanned aerial vehicles, and four unmanned ground vehicles with fault-tolerant and communication relay capabilities. It was demonstrated the computational power of the MCMAS by exploring 1.25809 million states which used 0.02% of the SMV model checker’s execution time. However, in (Choi, 2012), the external uncertainties were modelled into one environment model, which made it impossible to verify the system under multiple non-deterministic and concurrent uncertainties.

This research demonstrates the feasibility of implementing model checking algorithm for verifying the decision-making system of long endurance USV in dealing with multiple non-deterministic and concurrent environmental uncertainties. This chapter shows how to model multiple agents and multiple uncertainties and demonstrates how to use the counterexample to improve the design of the decision-making system. The model checking algorithm are implemented in three USV mission scenarios. Each scenario extends the previous scenario by adding additional uncertainties, agents and behaviours. The remainder of this chapter is organized as follows:

- Section 7.3 includes mission scenario description, modelling the concurrent uncertainties and the behaviours of the USV and GCS using Kripke modelling method, and using model checker MCMAS to verify the targeting properties. Section 7.4 and Section 7.5 have the similar structure with Section 7.3, but with extended mission scenarios.
- In Section 7.4, additional energy environmental uncertainties are taken account into the decision-making system of the long endurance USV, which has multiple energy resources. Both the safety property and the long endurance property are verified.
- In Section 7.5, a team of autonomous vehicles are deployed to improve the mission completion. The autonomous systems, including a long endurance USV, a regular USV, an UAV and a GCS, are modelled. A wireless mesh network is also constructed by using multiple autonomous systems as the communication nodes. Finally, the safety property, the long endurance property and the fault tolerance property are verified in Section 7.5.

7.3 USV Mission with Fault Event, Communication Loss and Collision Risk

This section is organised as follows: The USV mission scenario is explained in Section 7.3.1. Section 7.3.2 describes the Kripke models of the environmental uncertainties, Ground Control Station (GCS) decision-making system, and the USV decision-making system. In Section 7.3.3, the properties to be verified are introduced and the setup to run the MCMAS model checker for the verification is described. The verification result, the analysis and discussion are also presented.

7.3.1 Mission Scenario and Decision-Making System Introduction

Fig. 7. 4 depicts an overview of the USV mission scenario. In this scenario, the USV should go to the designated destination to execute its data collection mission. The GCS sends the mission plan (waypoint lists) to the USV and the USV should be launched after it receives the mission plan and the launch command. Once the USV is launched, it will follow the predefined path and it may encounter the collision risk, fault and communication loss problem. In the collision risk case, it will trigger the collision avoidance decision to avoid the obstacles. Once it avoids the obstacles, it will go back to the predefined path and continue the path following state. The communication states between the USV and the GCS includes good communication state and communication loss. When the communication is lost, the USV will trigger the station keeping behaviour, namely staying at the last reported position if it is not on a shipping lane, but if the last reported position is on the shipping lane, it will travel to a non-shipping lane area and stay there until the communication gets recovered. When the USV detects a fault, it will stay at standby state.

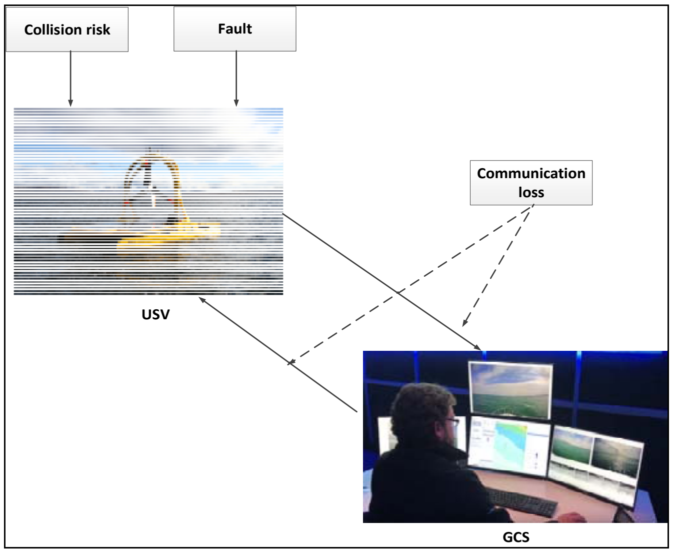


Fig. 7. 4. USV mission scenario

When the USV is travelling, it is assumed that possible problems, including collision risk, fault and communication loss, may be detected by the corresponding sensors, as shown in Fig. 7. 5. The state transitions of the sensors will trigger the transitions of the USV decision-making systems.

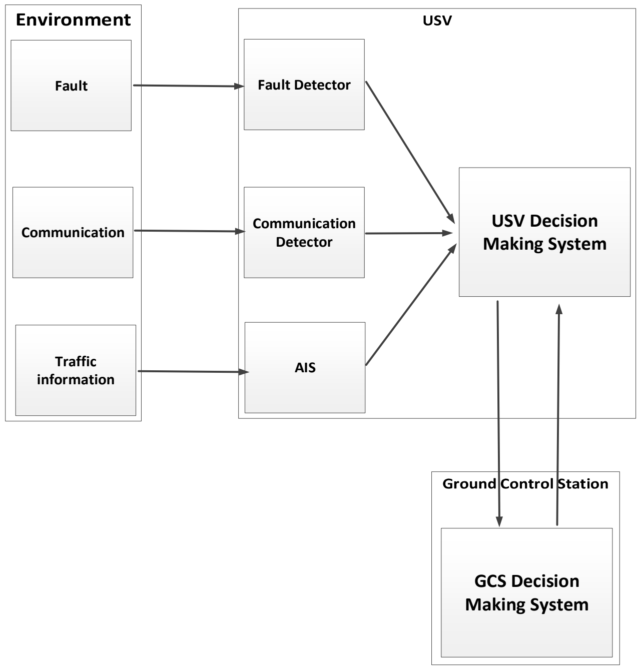


Fig. 7. 5. USV decision-making system

7.3.2 Kripke Modelling

In this section, the Kripke model of the environmental factor state and corresponding sensors' states are described first, namely , the Kripke models of the fault state and the fault detector state, the Kripke models of the communication channel state and the communication detector state, the Kripke models of the traffic information state and the AIS state. The Kripke models of the USV and the GCS decision-making systems are then presented.

7.3.2.1 Kripke model for Fault and Fault Detector

In Fig. 7. 6, The behaviours of fault event can be defined as fault state and non-fault state. The fault event is treated as a non-deterministic factor, which means each state may have several possible transitions. For example, if the state is fault state, it may have two transitions, namely t_1 and t_3, the next state may be fault state or non-fault state; Similarly, if the state is non-fault state, it may also have two transitions, namely t_2 and t_4, the next state may be fault or non-fault state. This non-deterministic model obeys the real case that the state at each specific moment cannot determine the exact state at the next moment.

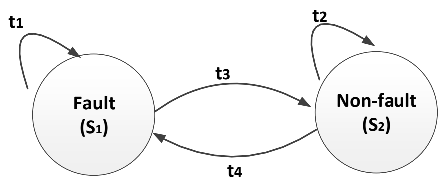


Fig. 7. 6. Kripke model for the fault event

The transitions of the fault event Kripke model are described as follows:

- $(S_1 \rightarrow S_1), (S_2 \rightarrow S_1)$: If the USV has fault event, these transitions happen.

$$t_1, t_4 = \begin{cases} True, & If\ the\ USV\ has\ fault\ event. \\ False, & Otherwise \end{cases}$$

- $(S_2 \rightarrow S_2), (S_1 \rightarrow S_2)$: If 'the USV has fault event' condition is false, these transitions happen.

$$t_2, t_3 = \begin{cases} True, & If\ 'the\ USV\ has\ no\ fault\ event'\ condition\ is\ false. \\ False, & Otherwise \end{cases}$$

We assume the vessel has the capability to detect its own malfunction. The Kripke model of the fault detector is shown in Fig. 7. 7. The states include *Fault detected* and *Non-fault detected*. The states of the fault detector are affected by the transitions of the fault event. Please note that although the Kripke models of fault event and fault detector are similar, the properties of the transitions are different, as the fault event is modelled as a non-deterministic system and the state of fault detector is determined by the transitions of fault event, and the ISPL code in Section 7.3.3.1 will explain the details of the transition conditions. In this work, to distinguish the autonomous agents (USV, UAV and GCS) from the environment-related agents, only the Kripke models of environment-related agents have self-loop.

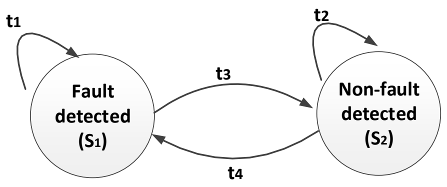


Fig. 7. 7. Kripke model for the fault detector

7.3.2.2 Kripke Model for Communication Channel and Communication Detector

In Fig. 7. 8, the behaviours of the communication channel between the USV and the GCS may be defined as communication state and communication loss state. The communication channel is also treated as a non-deterministic system.

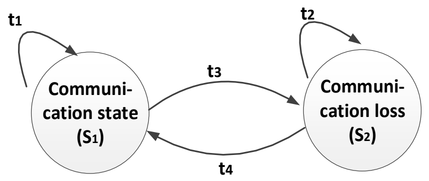


Fig. 7. 8. Kripke model for the communication channel

The transitions among the states may be described in detail as follows:

- $(S_1 \rightarrow S_1), (S_2 \rightarrow S_1)$: If the communication is normal, these transitions happen.

$$t_1, t_4 = \begin{cases} True, & If\ the\ communication\ is\ normal \\ False, & Otherwise \end{cases}$$

- $(S_2 \rightarrow S_2), (S_1 \rightarrow S_2)$: If 'the communication state is normal' condition is false, these transitions happen.

$$t_2, t_3 = \begin{cases} True, & If\ 'the\ communication\ state\ is\ normal'\ condition\ is\ false \\ False, & Otherwise \end{cases}$$

The communication state may be detected by checking the state of the data transition, here we name the communication detecting mechanism as communication detector. The behaviours of the communication detector are defined as communication state detected and communication loss detected. The Kripke model of the communication detector is depicted in Fig. 7. 9.

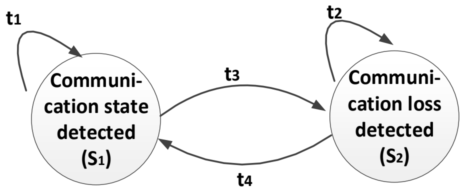

t1
t2
t3
t4
Communi-cation state detected (S1)
Communi-cation loss detected (S2)

Fig. 7. 9. Kripke model for the communication detector

7.3.2.3 Kripke Model for Traffic Information and AIS

When the USV is executing the mission, it may encounter collision risks. The collision risks may be classified into two kinds of risks based on COLREGS

(International Regulations for Preventing Collisions at Sea), namely give-way collision risk and the stand-on collision risk. The give-way collision risk represents the collision scenario that the USV should give way to the encountering vessel. The stand-on collision risk describes the collision scenario that the USV should keep on its way to avoid the collision risk. The USV will make the collision avoidance decision, namely to alter its course angle, when it encounters the give-way collision risk instead of the stand-on collision risk. Therefore, the traffic situation may be modelled using three states including no collision risk, give-way collision risk and stand-on collision risk. The traffic situation is also a non-deterministic system which means at each state, it may have three possible transitions, as shown in Fig. 7. 10. For example, when the traffic situation is at no collision risk state, it has three non-deterministic transitions including t_7, t_1 and t_5. Thus, next state may be no collision risk, give-way collision risk or stand-on collision risk. This non-deterministic model obeys the real traffic situation whose state at each specific moment cannot determine the exact state at the next moment.

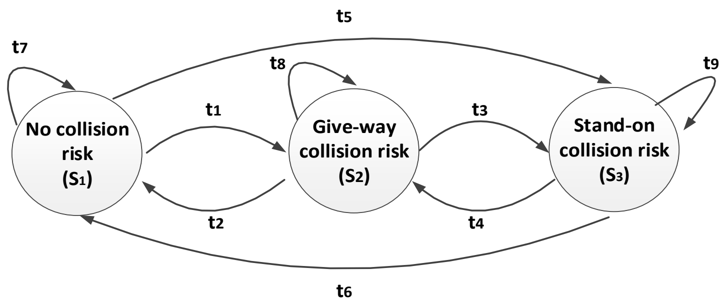


Fig. 7. 10. Kripke model for the traffic information

The transitions among these states are described as follows:

- $(S_1 \rightarrow S_1), (S_2 \rightarrow S_1), (S_3 \rightarrow S_1)$: If no collision risk appears, these transitions happen.

$$t_7, t_2, t_6 = \begin{cases} True, & If\ no\ collision\ risk\ appears. \\ False, & Otherwise \end{cases}$$

- $(S_2 \rightarrow S_2), (S_1 \rightarrow S_2), (S_3 \rightarrow S_2)$: If give-way collision risk appears, these transitions happen.

$$t_8, t_1, t_4 = \begin{cases} True, & If\ give - way\ collision\ risk\ appears. \\ False, & Otherwise \end{cases}$$

- $(S_3 \rightarrow S_3), (S_1 \rightarrow S_3), (S_2 \rightarrow S_3)$: If stand-on collision risk appears, these transitions happen.

$$t_9, t_5, t_3 = \begin{cases} True, & If\ stand - on\ collision\ risk\ appears. \\ False, & Otherwise \end{cases}$$

The traffic information may be detected by an Automatic Identification System (AIS) sensor. It is assumed that the AIS sensor may detect the traffic information correctly. The Kripke model of AIS is depicted in Fig. 7. 11, illustrating transition conditions similar to the Kripke model of the traffic information. The transitions of AIS are affected by the transitions of traffic information.

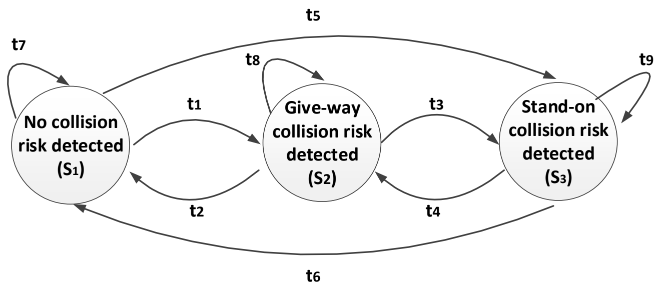

t5
t7
t8
t9
t1
t3
No collision risk detected (S1)
Give-way collision risk detected (S2)
Stand-on collision risk detected (S3)
t2
t4
t6

Fig. 7. 11. Kripke model for the AIS

7.3.2.4 Kripke Model for the USV

The behaviours of the USV are defined as follows: *Standby (SB), Ready (RE), Dispatched (DP), Path Following (PF), Collision Avoidance (CA), Station Keeping (SK)*, *Arrive (AR)* and *Fault (FA)*. The Kripke model of the USV behaviours is shown in Fig. 7. 12.

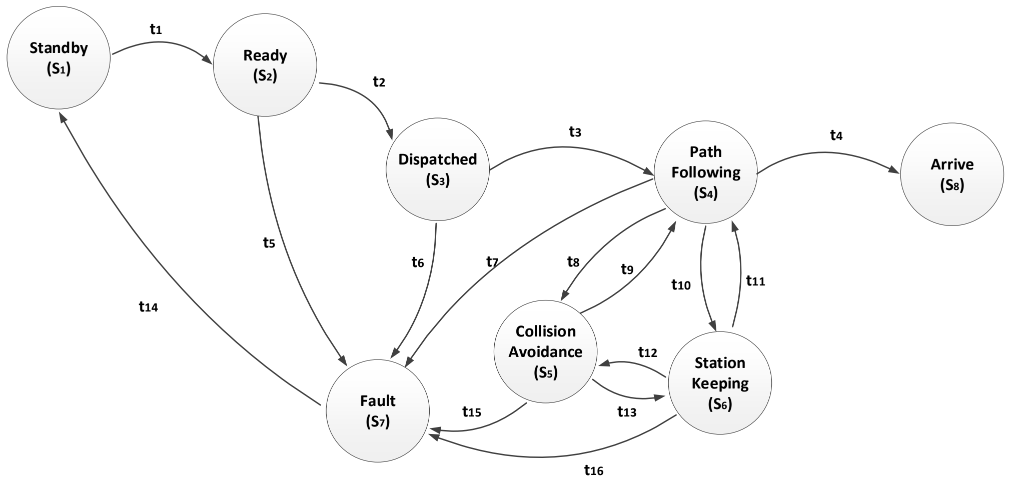


Fig. 7. 12. Kripke model for the USV

The accessibility relationships are described as follows:

- $(S_1 \rightarrow S_2)$: If the USV received the mission from the GCS and no fault is detected, this transition happens.

$$t_1 = \begin{cases} True, & If\ the\ USV\ received\ the\ mission\ \ and\ \ no\ fault\ detected \\ False, & Otherwise \end{cases}$$

- $(S_2 \rightarrow S_3)$: If the USV received the launching command from GCS and no fault is detected, this transition happens.

$$t_2 = \begin{cases} True, & If\ the\ USV\ received\ the\ mission \\ False, & Otherwise \end{cases}$$

- $(S_3 \rightarrow S_4)$: If the USV has been dispatched and no fault is detected, this transition happens.

$$t_3 = \begin{cases} True, & If\ the\ USV\ has\ been\ dispatched\ and\ no\ fault\ detected \\ False, & Otherwise \end{cases}$$

- $(S_4 \rightarrow S_8)$: If the USV arrived the destination, this transition happens.

$$t_4 = \begin{cases} True, & If\ the\ USV\ has\ arrived \\ False, & Otherwise \end{cases}$$

- $(S_5 \rightarrow S_4), (S_6 \rightarrow S_4)$: If the USV is in collision avoidance or station keeping state, no give-way collision risk is detected, communication channel is in good status and no fault is detected, these transitions happen.

$$t_9, t_{11} = \begin{cases} True, & If\ the\ USV\ is\ in\ collision\ avoidance \\ & or\ station\ keeping\ state, and\ no\ give-way\ collision \\ & risk\ found\ and\ communication\ channel\ is\ in\ good \\ & status\ and\ no\ fault\ is\ detected \\ False, & Otherwise \end{cases}$$

- $(S_4 \rightarrow S_5), (S_6 \rightarrow S_5)$: If the USV is in path following or station keeping state, a give-way collision risk is detected, communication channel is in good status and no fault is detected, these transitions happen.

$$t_8, t_{12} = \begin{cases} True, & If\ the\ USV\ in\ path\ following\ state\ or\ station \\ & keeping\ state, and\ a\ give-way\ collision \\ & risk\ appears\ and\ communication\ channel \\ & is\ in\ good\ status\ and\ no\ fault\ is\ detected. \\ False, & Otherwise \end{cases}$$

- $(S_4 \rightarrow S_6), (S_5 \rightarrow S_6)$: If the USV is in path following or collision avoidance state, no give-way collision risk appears, communication channel is lost and no fault is detected, these transitions happen.

$$t_{10}, t_{13} = \begin{cases} True, & If\ the\ USV\ in\ path\ following\ or\ collision\ avoidance \\ & state, and\ communication\ is\ lost\ and \\ & no\ give-way\ collision\ risk\ appears\ and\ no\ fault\ detected. \\ False, & Otherwise \end{cases}$$

- $(S_2 \rightarrow S_7), (S_3 \rightarrow S_7), (S_4 \rightarrow S_7), (S_5 \rightarrow S_7), (S_6 \rightarrow S_7)$: If the USV has detected fault, these transitions happen.

$$t_5, t_6, t_7, t_{15}, t_{16} = \begin{cases} True, & If\ the\ USV\ has\ detected\ fault. \\ False, & Otherwise \end{cases}$$

- $(S_7 \rightarrow S_1)$: If the USV has been in fault state, this transition happens.

$$t_{14} = \begin{cases} True, & If\ the\ USV\ has\ been\ in\ fault\ state \\ False, & Otherwise \end{cases}$$

7.3.2.5 Kripke Model for the GCS

The behaviours of the GCS are defined as follows: *Path planning (PP), Send waypoints (SW), Launch command (LC), Situation analysis (SiA), Path re-planning (PR)* and *Send new waypoint (SN)*. The Kripke model of the GCS behaviours is shown in Fig. 7. 13.

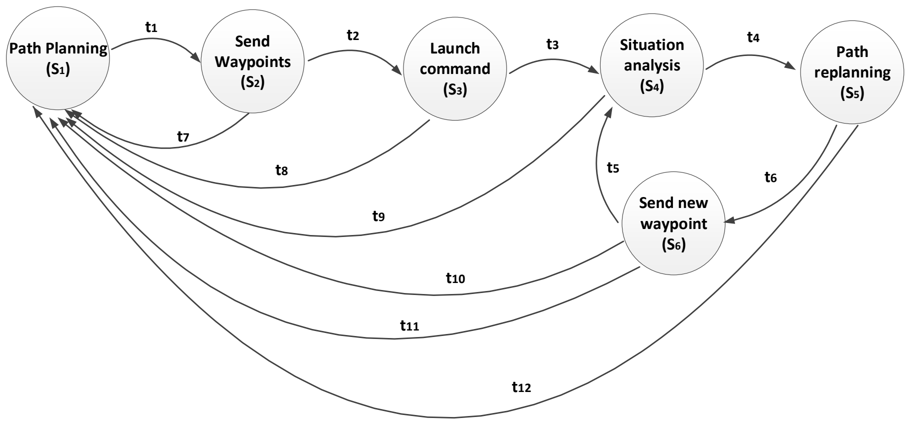


Fig. 7. 13. Kripke model for the GCS

The accessibility relationships are described as follows:

- $(S_1 \rightarrow S_2)$: If the GCS has planned the path and the USV has been in standby state, this transition happens.

$$t_1 = \begin{cases} True, & If\ the\ GCS\ has\ planned\ the\ path\ and\ the\ USV\ has\ been\ in \\ & standby\ state \\ False, & Otherwise \end{cases}$$

- $(S_2 \rightarrow S_3)$: If the GCS has sent waypoints and the USV has been in ready state, this transition happens.

$$t_2 = \begin{cases} True, & If\ the\ GCS\ has\ sent\ waypoints\ and\ the\ USV\ has\ been\ in \\ & dispatched\ state \\ False, & Otherwise \end{cases}$$

- $(S_3 \rightarrow S_4)$: If the GCS has sent launch command and the USV has been in path following state, this transition happens.

$$t_3 = \begin{cases} True, & If\ the\ GCS\ has\ sent\ launch\ command\ and\ the\ USV\ has\ been\ in \\ & path\ following\ state \\ False, & Otherwise \end{cases}$$

- $(S_4 \rightarrow S_5)$: If the GCS has been in the situation analysis state, the USV has been in station keeping state, and the communication state gets recovered, this transition happens.

$$t_4 = \begin{cases} True, & If\ the\ GCS\ has\ been\ in\ situation\ analysis, the\ USV\ has\ been \\ & in\ station\ keeping\ state, and\ the\ communication\ state\ gets\ recovered \\ False, & Otherwise \end{cases}$$

- $(S_5 \rightarrow S_6)$: If the GCS has planned the new path, this transition happens.

$$t_6 = \begin{cases} True, & If\ the\ GCS\ has\ planned\ the\ new\ path \\ False, & Otherwise \end{cases}$$

- $(S_6 \rightarrow S_4)$: If the GCS has sent new waypoints and the USV has been in path following mode, this transition happens.

$$t_5 = \begin{cases} True, & If\ the\ GCS\ has\ sent\ new\ waypoints\ and\ the\ USV\ has\ been \\ & in\ path\ following\ mode \\ False, & Otherwise \end{cases}$$

- $(S_2 \rightarrow S_1), (S_3 \rightarrow S_1), (S_4 \rightarrow S_1), (S_5 \rightarrow S_1), (S_6 \rightarrow S_1)$: If the USV has detected fault and the communication status is normal, these transitions happen.

$$t_7, t_8, t_9, t_{10}, t_{11}, t_{12} = \begin{cases} True, & If\ the\ USV\ has\ detected\ fault\ and \\ & the\ communication\ status\ is\ normal \\ False, & Otherwise \end{cases}$$

7.3.3 Model Checking with MCMAS

MCMAS uses its own language, ISPL, to describe the system model. ISPL has six essential parts including *Environment Agent, Agent, InitStates, Evaluation*, and *Formulae.* In the *Environment Agent* and *Agent* parts, the possible worlds, labelling function and transitions of the Kripke model may be parsed using state variables, actions, protocols and evolution. Possible worlds may be defined as state variables, labelling function corresponds to the protocol using actions, and the transitions may be represented by the evolution. The *InitStates* defines the initial states of all the agents. The atomic propositions of the properties to be verified are declared in the *Evaluation*. These propositions and CTL are used to describe how the behaviours of the system unfold over time. The properties we want to verify are expressed in the *Formulae* part.

7.3.3.1 MCMAS Model

The Kripke models of the decision-making system described in Section 7.3.2 are translated into ISPL. The ISPL code of the fault event and fault event detector, which are modelled in Section 7.3.2.1 (Fig. 7. 6 & Fig. 7. 7), is shown in Fig. 7. 14. The fault event is modelled as *Agent Fault*, and the fault event detector is modelled as *Agent FaultDetector*. The states of *Agent Fault* and *Agent FaultDetector* are described in Vars. Each agent is allowed to perform some actions, and these actions are visible by other agents. The actions correspond to the atomic propositions of the Kripke model. The Protocols in ISPL corresponds to the labelling function of the Kripke model. The Protocols describe which action may be performed in each state and correspond to which atomic proposition those hold in each state. The Evolution functions for an agent describes how the states transit as a result of the actions performed by all the other agents, corresponding to the transitions relation of the Kripke model that describe the condition of the state transitions.

In Fig. 7. 14, the *Agent Fault* shows two states: *Non-Fault state (NF)* and *Fault state (FA)*. The fault event is a non-deterministic model, so when *state* = *NF*, it has two possible actions, including: *Action* = *NF* and *Action* = *FA*, which are described in the Protocol. These two actions will be used as the conditions of the state transitions and they are described in the evolution part. That is, if *Action* = *NF*, *state* = *NF,* and if *!Action* = *NF*, *state* = *FA*. Therefore, if at a specific moment, the communication state is *NF*, then its next state may be *NF* or *FA*; If the communication state is *FA*, then its next state may also be *NF* or *FA*. The transition mechanism obeys the non-deterministic characteristic of the fault event in the real world, namely, the state at each moment may have several possible transitions.

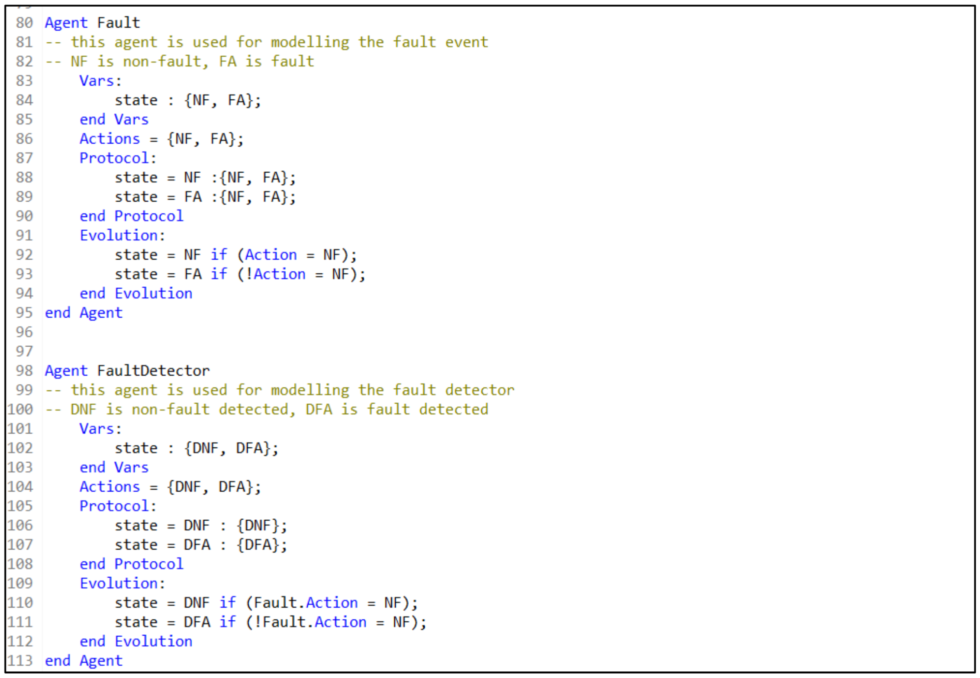
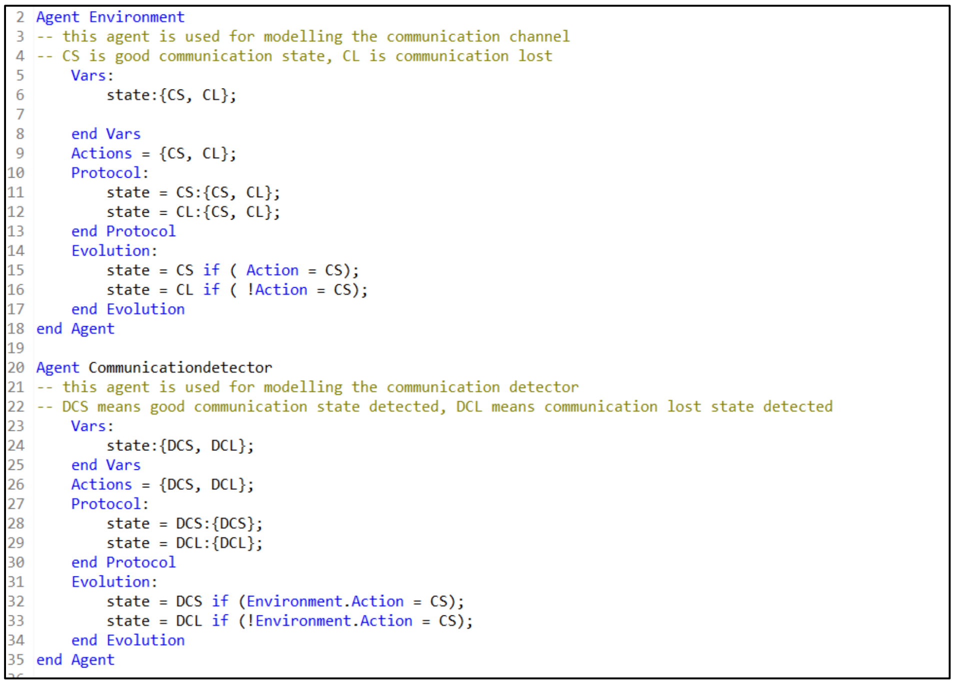

```
Agent Fault
-- this agent is used for modelling the fault event
-- NF is non-fault, FA is fault
    Vars:
        state : {NF, FA};
    end Vars
    Actions = {NF, FA};
    Protocol:
        state = NF :{NF, FA};
        state = FA :{NF, FA};
    end Protocol
    Evolution:
        state = NF if (Action = NF);
        state = FA if (!Action = NF);
    end Evolution
end Agent


Agent FaultDetector
-- this agent is used for modelling the fault detector
-- DNF is non-fault detected, DFA is fault detected
    Vars:
        state : {DNF, DFA};
    end Vars
    Actions = {DNF, DFA};
    Protocol:
        state = DNF : {DNF};
        state = DFA : {DFA};
    end Protocol
    Evolution:
        state = DNF if (Fault.Action = NF);
        state = DFA if (!Fault.Action = NF);
    end Evolution
end Agent
```

Fig. 7. 14. ISPL code of fault event and fault detector

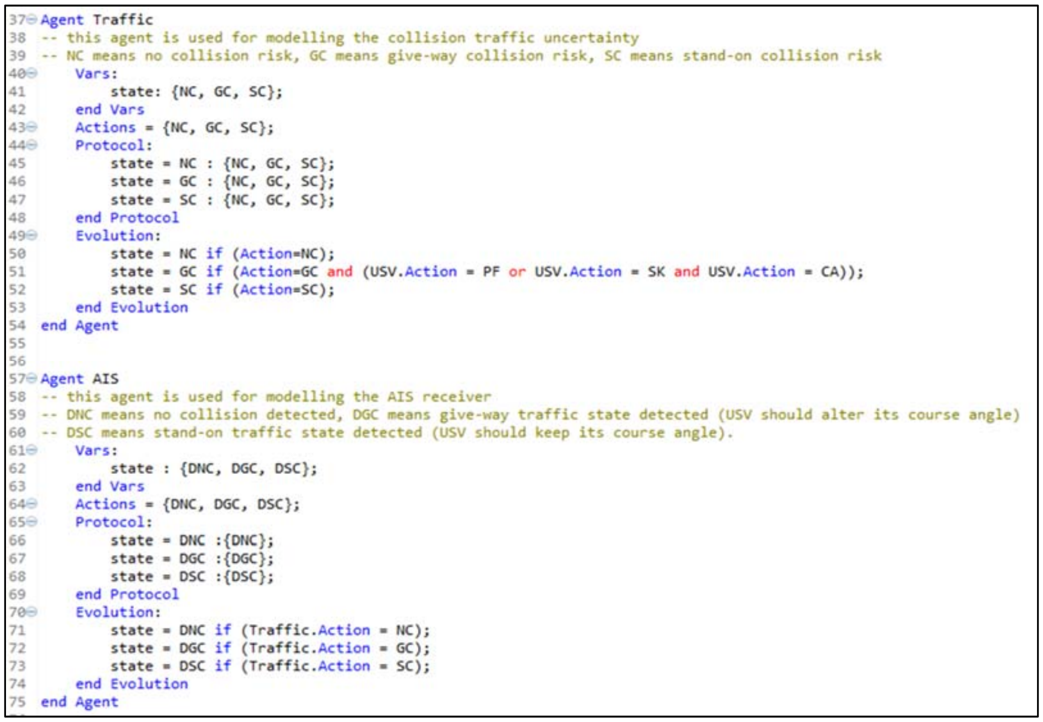
Unlike the fault event, the transitions condition of the *Agent FaultDetector* are deterministic and they are governed by the Actions of the fault event, if *Fault.Action* = *NF*, the state of the fault detector will be updated to *DNF* (non-fault state detected). The *DNF* state will have one action that is *FaultDetector.Action* = *DNF* and the action will be used as a transition condition of the *Agent USV*. These transition sequences obey the sequences of the real system. First, the *Action* of the fault event is generated. Second, the *State* of the fault detector will be updated according to the *Action* of the fault event. Finally, the newly updated state of the fault detector will have a new *Action* that will eventually be used as the transition condition of the *Agent USV*.

The ISPL code of the communication channel and communication detector is shown in Fig. 7. 15. The ISPL code of the traffic information and the AIS is shown in Fig. 7. 16. The ISPL code of the GCS is shown in Fig. 7. 17 and the ISPL code of the USV is shown in Fig. 7. 18.



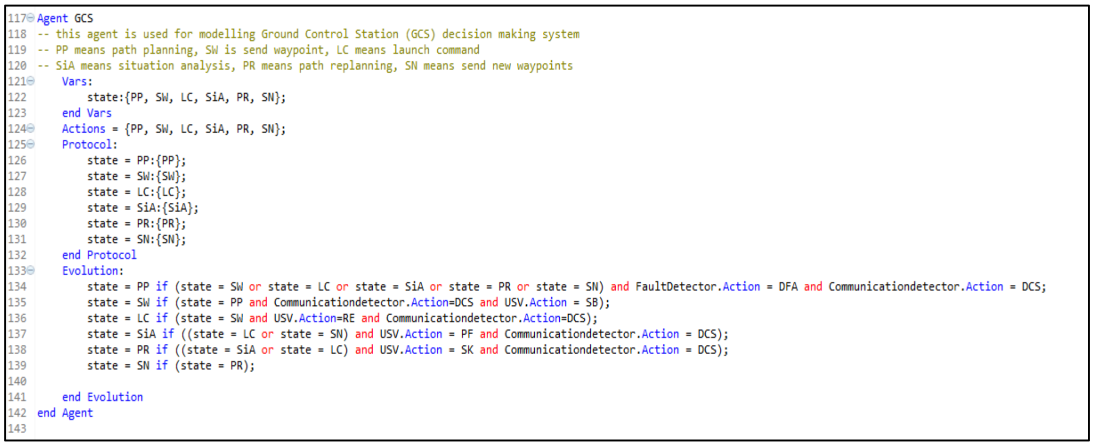
```
Agent Environment
-- this agent is used for modelling the communication channel
-- CS is good communication state, CL is communication lost
    Vars:
        state:{CS, CL};

    end Vars
    Actions = {CS, CL};
    Protocol:
        state = CS:{CS, CL};
        state = CL:{CS, CL};
    end Protocol
    Evolution:
        state = CS if ( Action = CS);
        state = CL if ( !Action = CS);
    end Evolution
end Agent

Agent Communicationdetector
-- this agent is used for modelling the communication detector
-- DCS means good communication state detected, DCL means communication lost state detected
    Vars:
        state:{DCS, DCL};
    end Vars
    Actions = {DCS, DCL};
    Protocol:
        state = DCS:{DCS};
        state = DCL:{DCL};
    end Protocol
    Evolution:
        state = DCS if (Environment.Action = CS);
        state = DCL if (!Environment.Action = CS);
    end Evolution
end Agent
```

Fig. 7. 15. ISPL code of communication channel and communication detector



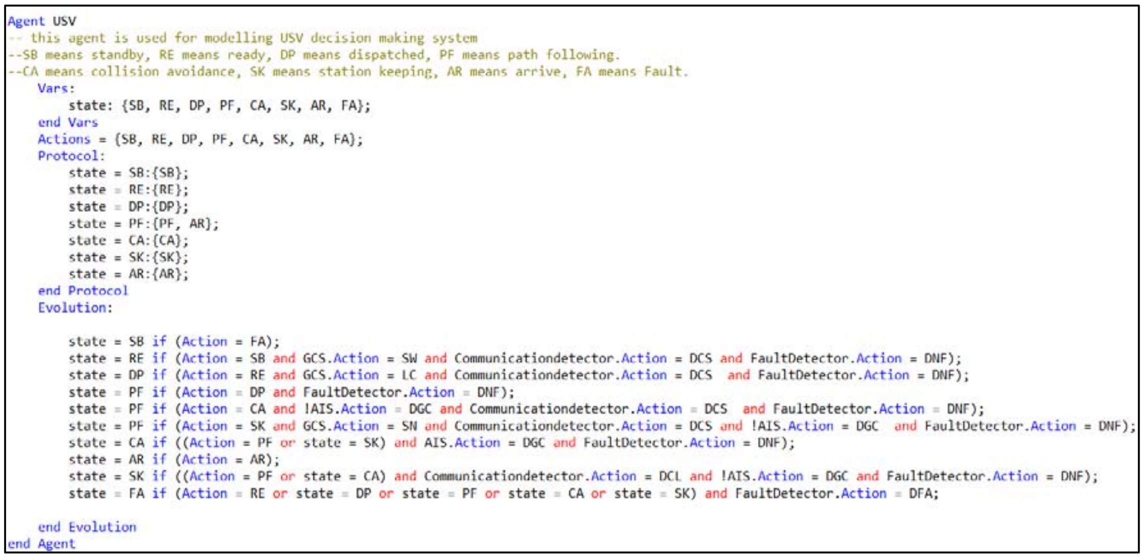
```
Agent Traffic
-- this agent is used for modelling the collision traffic uncertainty
-- NC means no collision risk, GC means give-way collision risk, SC means stand-on collision risk
    Vars:
        state: {NC, GC, SC};
    end Vars
    Actions = {NC, GC, SC};
    Protocol:
        state = NC : {NC, GC, SC};
        state = GC : {NC, GC, SC};
        state = SC : {NC, GC, SC};
    end Protocol
    Evolution:
        state = NC if (Action=NC);
        state = GC if (Action=GC and (USV.Action = PF or USV.Action = SK and USV.Action = CA));
        state = SC if (Action=SC);
    end Evolution
end Agent


Agent AIS
-- this agent is used for modelling the AIS receiver
-- DNC means no collision detected, DGC means give-way traffic state detected (USV should alter its course angle)
-- DSC means stand-on traffic state detected (USV should keep its course angle).
    Vars:
        state : {DNC, DGC, DSC};
    end Vars
    Actions = {DNC, DGC, DSC};
    Protocol:
        state = DNC :{DNC};
        state = DGC :{DGC};
        state = DSC :{DSC};
    end Protocol
    Evolution:
        state = DNC if (Traffic.Action = NC);
        state = DGC if (Traffic.Action = GC);
        state = DSC if (Traffic.Action = SC);
    end Evolution
end Agent
```

Fig. 7. 16. ISPL code of Traffic and AIS



```
Agent GCS
-- this agent is used for modelling Ground Control Station (GCS) decision making system
-- PP means path planning, SW is send waypoint, LC means launch command
-- SiA means situation analysis, PR means path replanning, SN means send new waypoints
    Vars:
        state:{PP, SW, LC, SiA, PR, SN};
    end Vars
    Actions = {PP, SW, LC, SiA, PR, SN};
    Protocol:
        state = PP:{PP};
        state = SW:{SW};
        state = LC:{LC};
        state = SiA:{SiA};
        state = PR:{PR};
        state = SN:{SN};
    end Protocol
    Evolution:
        state = PP if (state = SW or state = LC or state = SiA or state = PR or state = SN) and FaultDetector.Action = DFA and Communicationdetector.Action = DCS;
        state = SW if (state = PP and Communicationdetector.Action=DCS and USV.Action = SB);
        state = LC if (state = SW and USV.Action=RE and Communicationdetector.Action=DCS);
        state = SiA if ((state = LC or state = SN) and USV.Action = PF and Communicationdetector.Action = DCS);
        state = PR if ((state = SiA or state = LC) and USV.Action = SK and Communicationdetector.Action = DCS);
        state = SN if (state = PR);

    end Evolution
end Agent

```

Fig. 7. 17. ISPL code of GCS



```
Agent USV
-- this agent is used for modelling USV decision making system
--SB means standby, RE means ready, DP means dispatched, PF means path following.
--CA means collision avoidance, SK means station keeping, AR means arrive, FA means Fault.
    Vars:
        state: {SB, RE, DP, PF, CA, SK, AR, FA};
    end Vars
    Actions = {SB, RE, DP, PF, CA, SK, AR, FA};
    Protocol:
        state = SB:{SB};
        state = RE:{RE};
        state = DP:{DP};
        state = PF:{PF, AR};
        state = CA:{CA};
        state = SK:{SK};
        state = AR:{AR};
    end Protocol
    Evolution:

        state = SB if (Action = FA);
        state = RE if (Action = SB and GCS.Action = SW and Communicationdetector.Action = DCS and FaultDetector.Action = DNF);
        state = DP if (Action = RE and GCS.Action = LC and Communicationdetector.Action = DCS  and FaultDetector.Action = DNF);
        state = PF if (Action = DP and FaultDetector.Action = DNF);
        state = PF if (Action = CA and !AIS.Action = DGC and Communicationdetector.Action = DCS  and FaultDetector.Action = DNF);
        state = PF if (Action = SK and GCS.Action = SN and Communicationdetector.Action = DCS and !AIS.Action = DGC  and FaultDetector.Action = DNF);
        state = CA if ((Action = PF or state = SK) and AIS.Action = DGC and FaultDetector.Action = DNF);
        state = AR if (Action = AR);
        state = SK if ((Action = PF or state = CA) and Communicationdetector.Action = DCL and !AIS.Action = DGC and FaultDetector.Action = DNF);
        state = FA if (Action = RE or state = DP or state = PF or state = CA or state = SK) and FaultDetector.Action = DFA;

    end Evolution
end Agent
```

Fig. 7. 18. ISPL code of USV

7.3.3.2 Modelling of Properties to be Verified

There are twelve properties to be verified expressed by the CTL formula. The abbreviation for the states of the fault detector, the communication detector, the AIS, the USV and the GCS are described in the Table. 7. 1.

Table. 7. 1. Abbreviations and descriptions of the agents' behaviours

Agent	Abbreviation	Description
Communication detector	*DCS*	Good communication state is detected
	DCL	Communication loss is detected
Fault detector	*DNF*	No fault is detected
	DFA	Fault is detected
AIS	*DNC*	No collision risk is detected
	DGC	Give-way collision risk is detected
	DSC	Stand-on collision risk is detected
USV	*SB*	Standby
	RE	Ready
	DP	Dispatched
	PF	Path following
	CA	Collision avoidance
	SK	Station keeping
	AR	Arrive
	FA	Fault
GCS	*PP*	Path planning
	SW	Send waypoints
	LC	Command to launch
	SiA	Situation analysis
	PR	Path re-planning
	SN	Send new waypoints

The twelve properties and their corresponding CTL formulas are list as follows:

1, 'After the USV received the mission (ready state), if the communication state is good and no fault is detected by the USV, then the GCS sends the launching command, the USV will be dispatched'.

$$\begin{aligned} AG((USV.state = RE \wedge Communicationdetector.state \\ = DCS \wedge Faultdetector.state = DNF \wedge GCS.state = LC) \\ \rightarrow AX(USV.state = DP)) \end{aligned}$$

2, 'When the USV is in the path following state and no fault is detected, then a give-way collision risk is detected, in the next moment, the USV will either change way to avoid the collision or arrive at the destination'.

$$AG((USV.state = PF \land Faultdetector.state = DNF \land AIS.state = DGC) \rightarrow AX(USV.state = CA \lor USV.state = AR))$$

3, 'When a stand-on collision risk appears, the USV will not change its way to avoid the collision'.

$$AG(USV.state = DSC \rightarrow AX(!USV.state = CA))$$

4, 'If the give-way collision does not appear, the USV will never alter its way to avoid the collision'.

$$AG(!AIS.state = DGC) \rightarrow AF(!USV.state = CA)$$

5, 'After the USV avoided the collision risk, if the communication state is good, no give-way collision risk is detected and no fault is detected, the USV will always continue to follow the path'.

$$AG((USV.state = CA \land Communicationdetector.state = DCS \land !AIS.state = DGS \land Faultdetector.state = DNF) \rightarrow AX(USV.state = PF))$$

6, 'When the USV is in the station keeping state and no fault is detected and the GCS is in the situation analysis state, if good communication state is detected, the GCS will always re-plan the path'.

$$AG((USV.state = SK \land Faultdetector.state = DNF \land GCS.state = SiA \land Communicationdetector.state = DCS) \rightarrow AX(GCS.state = PR))$$

7, 'After the GCS re-planned the path, it will send new waypoints to the USV'.

$$AG(GCS.state = PR \rightarrow AX(GCS.state = SN))$$

8, 'When the USV is in path following state, and there is no give-way collision risk detected and no fault event is detected, if the communication is lost, the USV will either change to station keeping state or arrive at the destination'.

$$AG((USV.state = PF \wedge Faultdetector.state = DNF \wedge !AIS.state = DGC \wedge Communicationdetector.state = DCL) \rightarrow AX(USV.state = SK \vee USV.state = AR))$$

9, 'When the USV is in station keeping state, if the communication state is good and no fault is detected, after the GCS sends the new waypoints, the USV will change to path following state'.

$$AG((USV.state = SK \wedge Communicationdetector.state = DCS \wedge Faultdetector.state = DNF \wedge GCS.state = SN) \rightarrow AX(USV.state = PF))$$

10, 'When the USV is in the path following state and no fault is detected, if the communication loss is detected, the USV will change to station keeping state'.

$$AG((USV.state = PF \wedge Faultdetector.state = DNF \wedge Communicationdetector.state = DCL) \rightarrow AX(USV.state = SK))$$

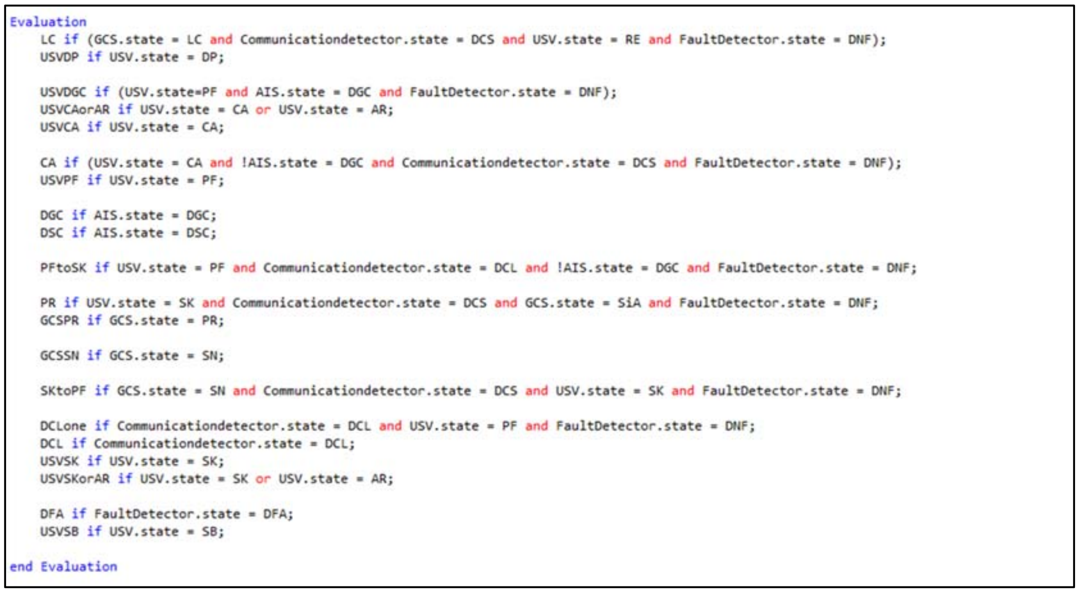
11, 'If the communication loss state is not detected, the USV will never change to station keeping state'.

$$AG(!Communicationdetector.state = DCL) \rightarrow AF(!USV.state = SK)$$

12, 'If the USV fault event is detected, the USV will eventually change to standby state'.

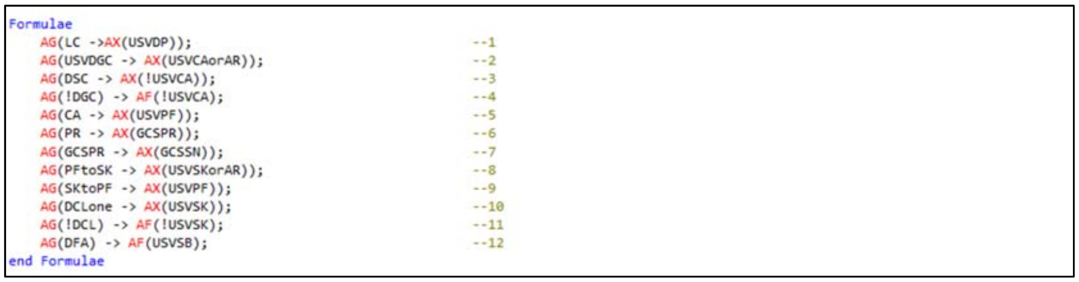
$$AG(Faultdetector.state = DNF) \rightarrow AF(USV.state = SB)$$

The properties are expressed in *Evaluation* and *Formulae* sections, as shown in Fig. 7. 19 and Fig. 7. 20.



```
Evaluation
    LC if (GCS.state = LC and Communicationdetector.state = DCS and USV.state = RE and FaultDetector.state = DNF);
    USVDP if USV.state = DP;

    USVDGC if (USV.state=PF and AIS.state = DGC and FaultDetector.state = DNF);
    USVCAorAR if USV.state = CA or USV.state = AR;
    USVCA if USV.state = CA;

    CA if (USV.state = CA and !AIS.state = DGC and Communicationdetector.state = DCS and FaultDetector.state = DNF);
    USVPF if USV.state = PF;

    DGC if AIS.state = DGC;
    DSC if AIS.state = DSC;

    PFtoSK if USV.state = PF and Communicationdetector.state = DCL and !AIS.state = DGC and FaultDetector.state = DNF;

    PR if USV.state = SK and Communicationdetector.state = DCS and GCS.state = SiA and FaultDetector.state = DNF;
    GCSPR if GCS.state = PR;

    GCSSN if GCS.state = SN;

    SKtoPF if GCS.state = SN and Communicationdetector.state = DCS and USV.state = SK and FaultDetector.state = DNF;

    DCLone if Communicationdetector.state = DCL and USV.state = PF and FaultDetector.state = DNF;
    DCL if Communicationdetector.state = DCL;
    USVSK if USV.state = SK;
    USVSKorAR if USV.state = SK or USV.state = AR;

    DFA if FaultDetector.state = DFA;
    USVSB if USV.state = SB;

end Evaluation
```

Fig. 7. 19. ISPL code for Evaluation

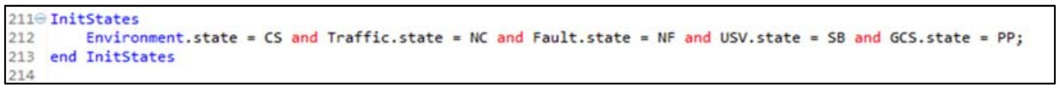


```
Formulae
    AG(LC ->AX(USVDP));                         --1
    AG(USVDGC -> AX(USVCAorAR));                --2
    AG(DSC -> AX(!USVCA));                      --3
    AG(!DGC) -> AF(!USVCA);                     --4
    AG(CA -> AX(USVPF));                        --5
    AG(PR -> AX(GCSPR));                        --6
    AG(GCSPR -> AX(GCSSN));                     --7
    AG(PFtoSK -> AX(USVSKorAR));                --8
    AG(SKtoPF -> AX(USVPF));                    --9
    AG(DCLone -> AX(USVSK));                    --10
    AG(!DCL) -> AF(!USVSK);                     --11
    AG(DFA) -> AF(USVSB);                       --12
end Formulae
```

Fig. 7. 20. ISPL code for Formulae

7.3.3.3 Verification Result

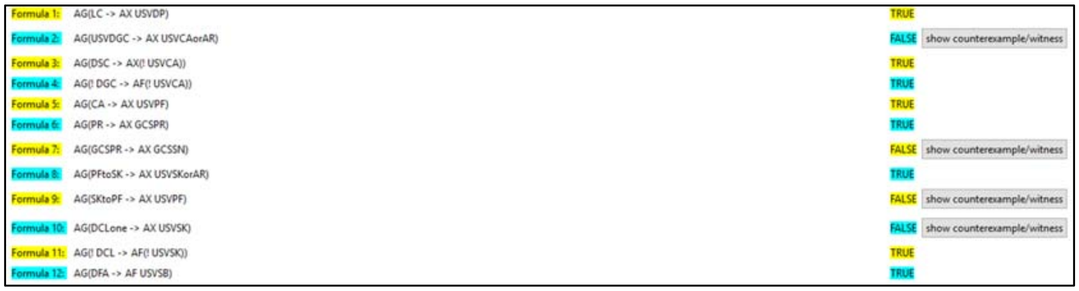
The *Environment Agent, Agent, Evaluation* and *Formulae* are presented in Section 7.3.3.1 and Section 7.3.3.2. Verification includes defining the *InitStates*. The *InitStates* used in this section is shown in Fig. 7. 21.



```
InitStates
    Environment.state = CS and Traffic.state = NC and Fault.state = NF and USV.state = SB and GCS.state = PP;
end InitStates
```

Fig. 7. 21. ISPL code for InitStates

The initial environmental factors are defined as good communication, no traffic risk is detected and no fault is detected. The USV initial state is Standby and the GCS initial state is Path Planning. The verification results are shown in Fig. 7. 22.



```
Formula 1:   AG(LC -> AX USVDP)                  TRUE
Formula 2:   AG(USVDGC -> AX USVCAorAR)          FALSE  show counterexample/witness
Formula 3:   AG(DSC -> AX(! USVCA))              TRUE
Formula 4:   AG(! DGC -> AF(! USVCA))            TRUE
Formula 5:   AG(CA -> AX USVPF)                  TRUE
Formula 6:   AG(PR -> AX GCSPR)                  TRUE
Formula 7:   AG(GCSPR -> AX GCSSN)               FALSE  show counterexample/witness
Formula 8:   AG(PFtoSK -> AX USVSKorAR)          TRUE
Formula 9:   AG(SKtoPF -> AX USVPF)              FALSE  show counterexample/witness
Formula 10:  AG(DCLone -> AX USVSK)              FALSE  show counterexample/witness
Formula 11:  AG(! DCL -> AF(! USVSK))            TRUE
Formula 12:  AG(DFA -> AF USVSB)                 TRUE
```

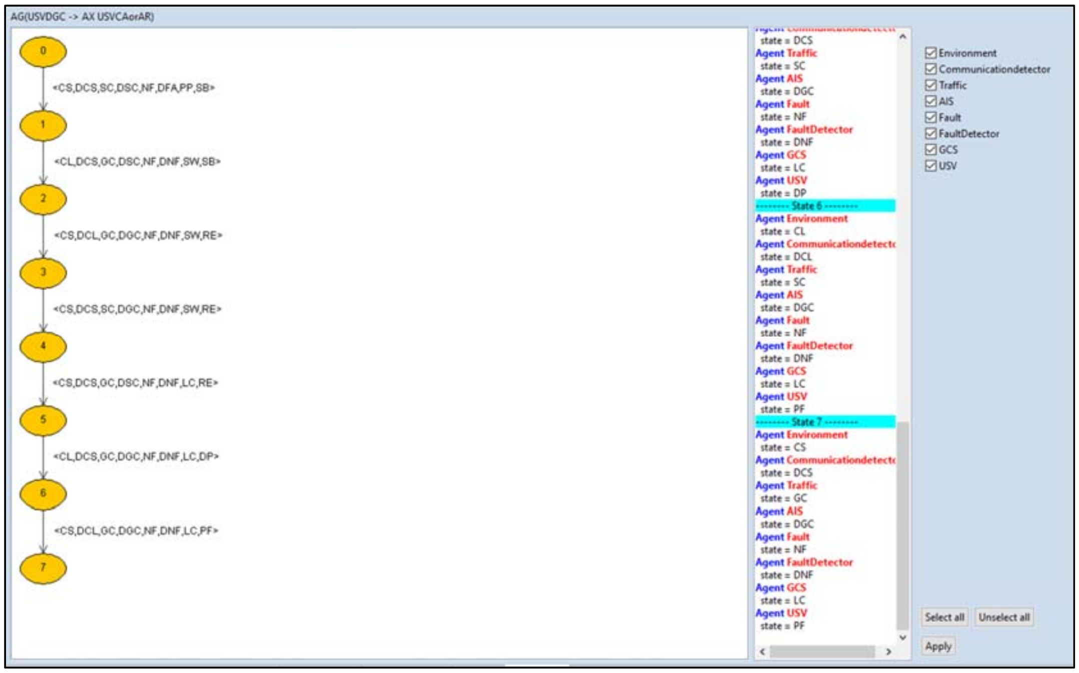
Fig. 7. 22. Verification result of MCMAS

The twelve properties described in Section 7.3.3.2 correspond to the twelve Formulas listed in Fig. 7. 22. Formula 1, Formula 3, Formula 4, Formula 5, Formula 6, Formula 8, Formula 11 and Formula 12 have '*TRUE'* results. Formula 2, Formula 7, Formula 9 and Formula 10 have '*FALSE'* results. The error trace may be acquired using the 'show counterexample/witness' option, as discussed in the next section.

7.3.3.4 Analysis and Discussion

Formula 1 acquires the result '*True'*, which shows the USV may be dispatched normally. Formulas 3, 4 and 5 acquire the result '*True'*, which shows the decision-making behaviours of the USV in relation to the collision avoidance performed well. The result of Formula 6 shows the GCS performs path re-planning behaviour well. The result of Formulas 8 and 11 shows the USV transited to the station keeping state properly under specific conditions. The result of Formula 12 shows that when the fault event is detected, the USV will standby, which obeys the safety design of the decision-making system.

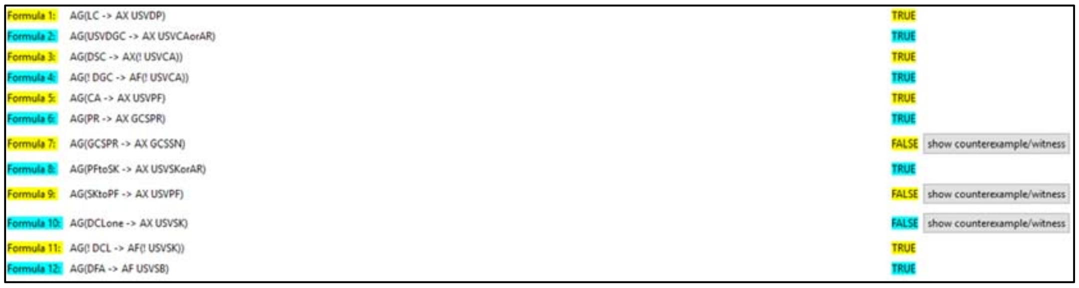
The property of Formula 2 reads, '*When the USV is in the path following state and no fault is detected, then a give-way collision risk is detected, in the next moment, the USV will either change way to avoid the collision or arrive the* *destination"*. However, Fig. 7. 23 illustrates that the USV remains at path following state instead of collision avoidance or arrive state. By checking the state transition condition of the USV in Fig. 7. 12, we found that the transition state from path following to collision avoidance is that '*If the USV is in path following or collision avoidance or station keeping state, and a give-way collision risk is detected and communication channel is in good status and no fault is detected, the USV will transit from path following to collision avoidance'.*


AG(USVDGC -> AX USVCAorAR)
0
<CS,DCS,SC,DSC,NF,DFA,PP,SB>
1
<CL,DCS,GC,DSC,NF,DNF,SW,SB>
2
<CS,DCL,GC,DGC,NF,DNF,SW,RE>
3
<CS,DCS,SC,DGC,NF,DNF,SW,RE>
4
<CS,DCS,GC,DSC,NF,DNF,LC,RE>
5
<CL,DCS,GC,DGC,NF,DNF,LC,DP>
6
<CS,DCL,GC,DGC,NF,DNF,LC,PF>
7
Environment
Communicationdetector
Traffic
AIS
Fault
FaultDetector
GCS
USV
Select all
Unselect all
Apply

```
 state = DCS
Agent Traffic
 state = SC
Agent AIS
 state = DGC
Agent Fault
 state = NF
Agent FaultDetector
 state = DNF
Agent GCS
 state = LC
Agent USV
 state = DP
-------- State 6 --------
Agent Environment
 state = CL
Agent Communicationdetectc
 state = DCL
Agent Traffic
 state = SC
Agent AIS
 state = DGC
Agent Fault
 state = NF
Agent FaultDetector
 state = DNF
Agent GCS
 state = LC
Agent USV
 state = PF
-------- State 7 --------
Agent Environment
 state = CS
Agent Communicationdetectc
 state = DCS
Agent Traffic
 state = GC
Agent AIS
 state = DGC
Agent Fault
 state = NF
Agent FaultDetector
 state = DNF
Agent GCS
 state = LC
Agent USV
 state = PF
```

Fig. 7. 23. The counterexample of Formula 2

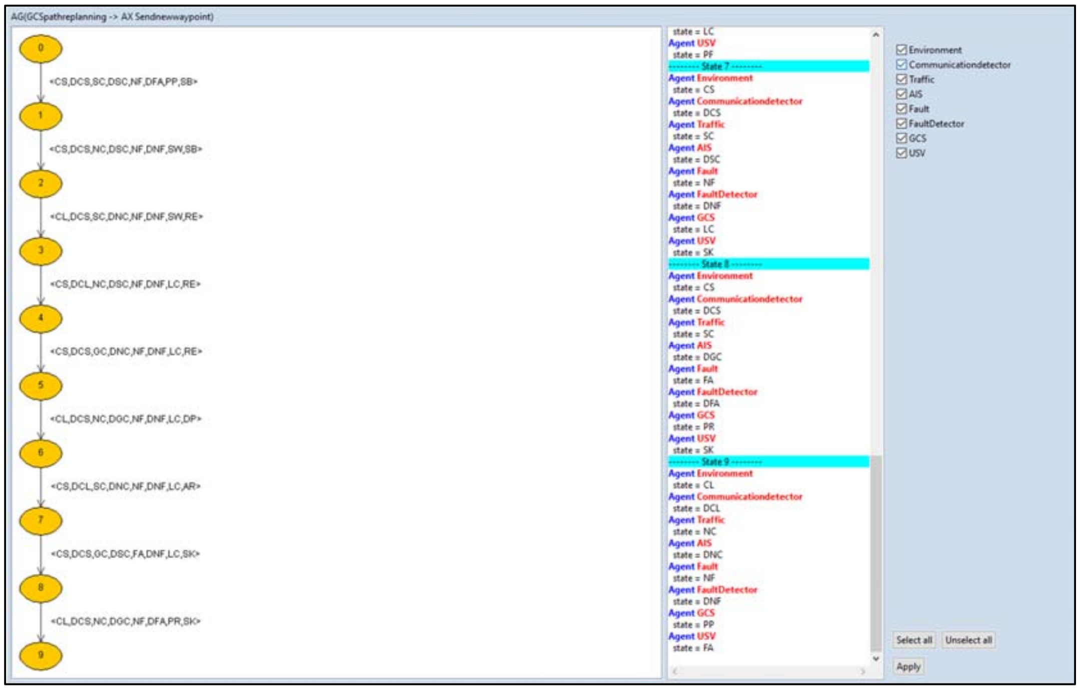
However, in Fig. 7. 23, we found that the communication loss and the give-way collision are detected at the same time, therefore, it doesn't satisfy the requirement of state transition from path following to collision avoidance, and it stays at path following state. In terms of safety, when the give-way collision risk is detected, we need the USV to avoid the collision even if the communication is lost, therefore we change the state transition condition as follows: '*If the USV is in path following or collision avoidance or station keeping state, and a give-way collision risk is detected and no fault is detected, the USV will transit from path following to collision avoidance'.* The corrected Kripke model is translated to ISPL code again, and we get the correct result for Formula 2, shown in Fig. 7. 24. The result of Formula 2 shows that the modelling method used in this research may help the USV system detect the design mistake under concurrent uncertainties.



```
Formula 1:   AG(LC -> AX USVDP)                TRUE
Formula 2:   AG(USVDGC -> AX USVCAorAR)        TRUE
Formula 3:   AG(DSC -> AX(! USVCA))            TRUE
Formula 4:   AG(! DGC -> AF(! USVCA))          TRUE
Formula 5:   AG(CA -> AX USVPF)                TRUE
Formula 6:   AG(PR -> AX GCSPR)                TRUE
Formula 7:   AG(GCSPR -> AX GCSSN)             FALSE  show counterexample/witness
Formula 8:   AG(PFtoSK -> AX USVSKorAR)        TRUE
Formula 9:   AG(SKtoPF -> AX USVPF)            FALSE  show counterexample/witness
Formula 10:  AG(DCLone -> AX USVSK)            FALSE  show counterexample/witness
Formula 11:  AG(! DCL -> AF(! USVSK))          TRUE
Formula 12:  AG(DFA -> AF USVSB)               TRUE
```

Fig. 7. 24. The verification result of the corrected model

The property of Formula 7 reads, '*After the GCS re-planned the path (PR), it will send new waypoints (SN) to the USV*'. Therefore, the state of GCS should change from *PR* to *SN*. However, Fig. 7. 25 illustrates that this property is violated in some branch of the computational tree.



AG(GCSpathreplanning -> AX Sendnewwaypoint)

```
0
  <CS,DCS,SC,DSC,NF,DFA,PP,SB>
1
  <CS,DCS,NC,DSC,NF,DNF,SW,SB>
2
  <CL,DCS,SC,DNC,NF,DNF,SW,RE>
3
  <CS,DCL,NC,DSC,NF,DNF,LC,RE>
4
  <CS,DCS,GC,DNC,NF,DNF,LC,RE>
5
  <CL,DCS,NC,DGC,NF,DNF,LC,DP>
6
  <CS,DCL,SC,DNC,NF,DNF,LC,AR>
7
  <CS,DCS,GC,DSC,FA,DNF,LC,SK>
8
  <CL,DCS,NC,DGC,NF,DFA,PR,SK>
9
```

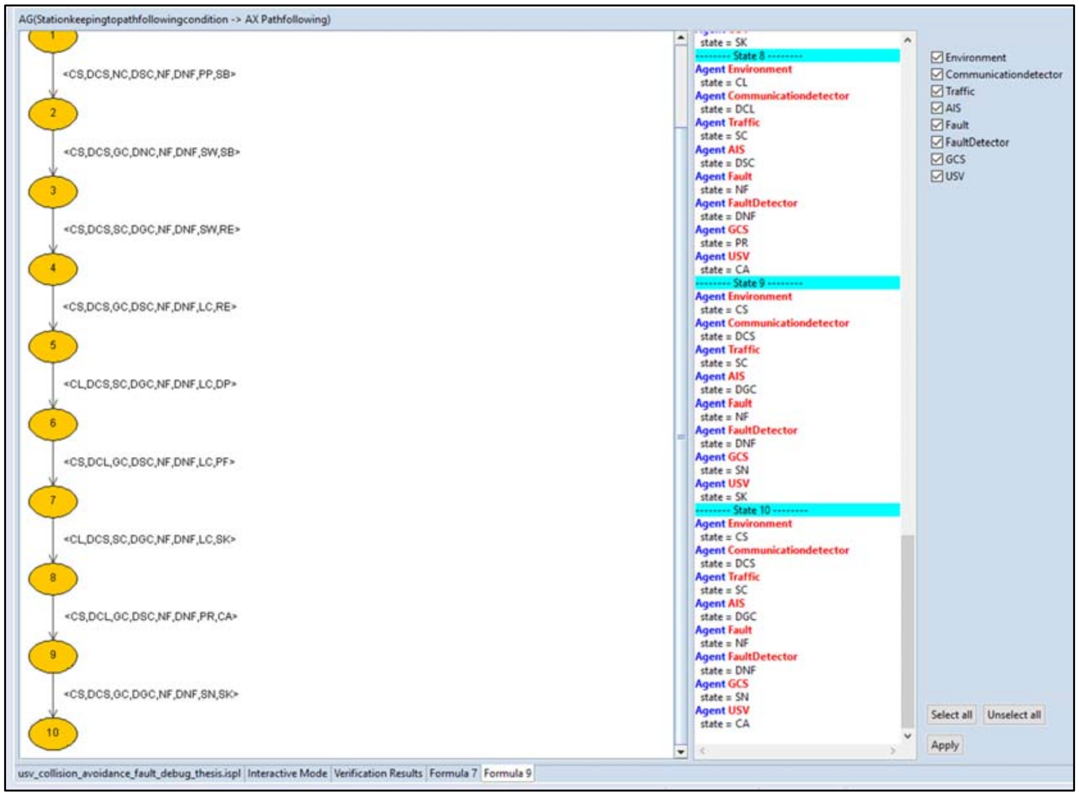
```
 state = LC
Agent USV
 state = PF
-------- State 7 --------
Agent Environment
 state = CS
Agent Communicationdetector
 state = DCS
Agent Traffic
 state = SC
Agent AIS
 state = DSC
Agent Fault
 state = NF
Agent FaultDetector
 state = DNF
Agent GCS
 state = LC
Agent USV
 state = SK
-------- State 8 --------
Agent Environment
 state = CS
Agent Communicationdetector
 state = DCS
Agent Traffic
 state = SC
Agent AIS
 state = DGC
Agent Fault
 state = FA
Agent FaultDetector
 state = DFA
Agent GCS
 state = PR
Agent USV
 state = SK
-------- State 9 --------
Agent Environment
 state = CL
Agent Communicationdetector
 state = DCL
Agent Traffic
 state = NC
Agent AIS
 state = DNC
Agent Fault
 state = NF
Agent FaultDetector
 state = DNF
Agent GCS
 state = PP
Agent USV
 state = FA
```

☑ Environment
☑ Communicationdetector
☑ Traffic
☑ AIS
☑ Fault
☑ FaultDetector
☑ GCS
☑ USV

Fig. 7. 25. The counterexample of Formula 7

From State 8 to State 9, we found the state of GCS changed from *PR* (*Path Re-planning*) to PP (*Path Planning*) as the fault event of the USV was detected and the USV state changed from *SK* (*Station Keeping*) to *FA* (*Fault*). This counterexample obeys the safety of the decision-making system. When the USV detects the fault, it will be in the fault mode and standby, and the GCS should change to the initial path planning mode instead of sending the new waypoints to the USV.

The counterexample of Formula 9 is shown in Fig. 7. 26. The property of Formula 9 reads, ‘*When the USV is in station keeping state, if the communication state is good and no fault is detected and the GCS sends the new waypoints, the USV will change to path following state*’.



```
AG(Stationkeepingtopathfollowingcondition -> AX Pathfollowing)

1
  <CS,DCS,NC,DSC,NF,DNF,PP,SB>
2
  <CS,DCS,GC,DNC,NF,DNF,SW,SB>
3
  <CS,DCS,SC,DGC,NF,DNF,SW,RE>
4
  <CS,DCS,GC,DSC,NF,DNF,LC,RE>
5
  <CL,DCS,SC,DGC,NF,DNF,LC,DP>
6
  <CS,DCL,GC,DSC,NF,DNF,LC,PF>
7
  <CL,DCS,SC,DGC,NF,DNF,LC,SK>
8
  <CS,DCL,GC,DSC,NF,DNF,PR,CA>
9
  <CS,DCS,GC,DGC,NF,DNF,SN,SK>
10
```

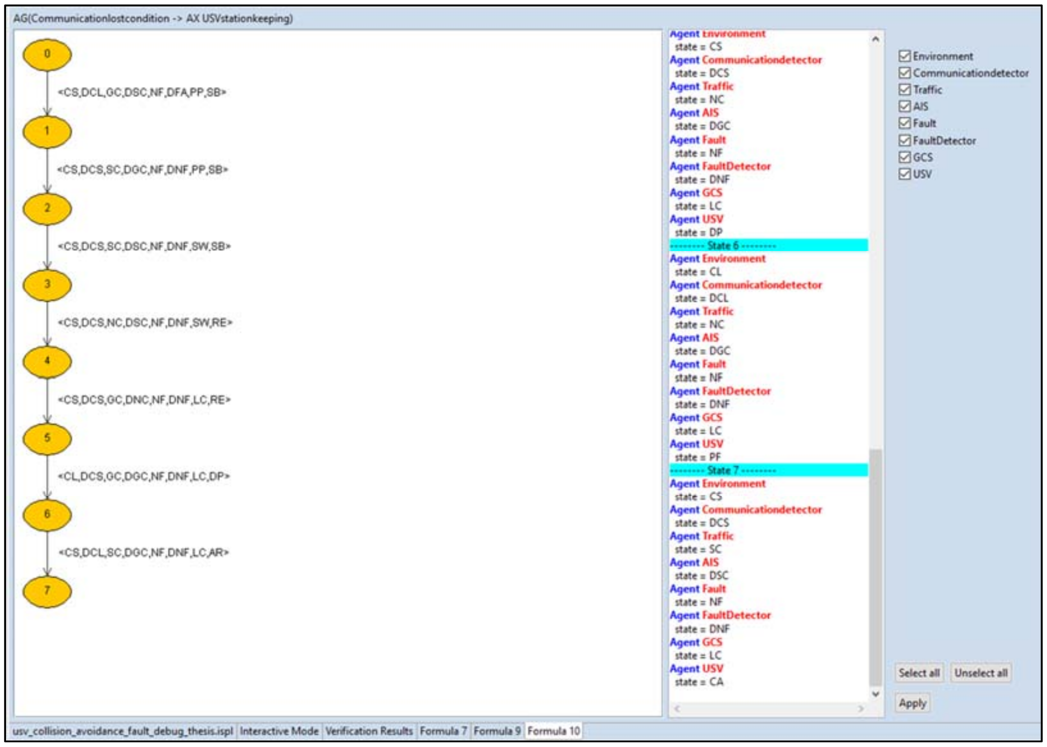
```
 state = SK
--------- State 8 ---------
Agent Environment
 state = CL
Agent Communicationdetector
 state = DCL
Agent Traffic
 state = SC
Agent AIS
 state = DSC
Agent Fault
 state = NF
Agent FaultDetector
 state = DNF
Agent GCS
 state = PR
Agent USV
 state = CA
--------- State 9 ---------
Agent Environment
 state = CS
Agent Communicationdetector
 state = DCS
Agent Traffic
 state = SC
Agent AIS
 state = DGC
Agent Fault
 state = NF
Agent FaultDetector
 state = DNF
Agent GCS
 state = SN
Agent USV
 state = SK
--------- State 10 ---------
Agent Environment
 state = CS
Agent Communicationdetector
 state = DCS
Agent Traffic
 state = SC
Agent AIS
 state = DGC
Agent Fault
 state = NF
Agent FaultDetector
 state = DNF
Agent GCS
 state = SN
Agent USV
 state = CA
```

☑ Environment
☑ Communicationdetector
☑ Traffic
☑ AIS
☑ Fault
☑ FaultDetector
☑ GCS
☑ USV

Fig. 7. 26. The counterexample of Formula 9

The counterexample shows that the transition from State 9 to State 10 indicates the state of the USV changes from *SK* to *CA* instead of *PF.* The state change occurred when the communication channel recovered and a give-way collision risk appeared at the same time, allowing the USV to first avoided the collision instead of continuing to follow the path. This counterexample is reasonable from a safety perspective, as when a give-way collision risk is detected, avoiding the collision has the priority over following the path.

The counterexample of Formula 10 is shown in Fig. 7. 27. The property of Formula 10 is '*When the USV is in the path following state and no fault is detected, if the communication loss is detected, the USV will change to the station keeping state*'.


```
AG(Communicationlostcondition -> AX USVstationkeeping)

0
<CS,DCL,GC,DSC,NF,DFA,PP,SB>
1
<CS,DCS,SC,DGC,NF,DNF,PP,SB>
2
<CS,DCS,SC,DSC,NF,DNF,SW,SB>
3
<CS,DCS,NC,DSC,NF,DNF,SW,RE>
4
<CS,DCS,GC,DNC,NF,DNF,LC,RE>
5
<CL,DCS,GC,DGC,NF,DNF,LC,DP>
6
<CS,DCL,SC,DGC,NF,DNF,LC,AR>
7
```

```
Agent Environment
  state = CS
Agent Communicationdetector
  state = DCS
Agent Traffic
  state = NC
Agent AIS
  state = DGC
Agent Fault
  state = NF
Agent FaultDetector
  state = DNF
Agent GCS
  state = LC
Agent USV
  state = DP
-------- State 6 --------
Agent Environment
  state = CL
Agent Communicationdetector
  state = DCL
Agent Traffic
  state = NC
Agent AIS
  state = DGC
Agent Fault
  state = NF
Agent FaultDetector
  state = DNF
Agent GCS
  state = LC
Agent USV
  state = PF
-------- State 7 --------
Agent Environment
  state = CS
Agent Communicationdetector
  state = DCS
Agent Traffic
  state = SC
Agent AIS
  state = DSC
Agent Fault
  state = NF
Agent FaultDetector
  state = DNF
Agent GCS
  state = LC
Agent USV
  state = CA
```

Fig. 7. 27. The counterexample of Formula 10

From State 6 to State 7, we may find the state of USV change from *PF* (*Path Following*) to *CA* (*Collision Avoidance*) instead of *SK* (*Station Keeping*). From State 9, we find that when the communication loss is detected and a give-way collision risk is also detected by AIS, the USV will avoid the collision risk first instead of station keeping. This counterexample obeys the safety design of the decision-making system and shows that the proposed modelling method may help us verify the decision-making system under multiple concurrent uncertainties.

The counterexample of Formula 2 helps us to improve the safety property of the decision-making system when the communication loss and collision risk events happen at the same time. The counterexamples of Formulas 7, 9 and 10 are not expected when we design the decision-making system. However, these counterexamples are reasonable for the safety of the decision-making system. We conclude that the model checking method helps us to find unexpected situations to ensure the safety of the decision-making system.

7.4 Long Range USV Mission with Multiple Energy Resources

In this section, the decision-making system of the USV with multiple energy resources is modelled and verified. The mission scenario in this section extends the previous mission scenario by taking into account energy generation and energy consumption. This section is organized as follows: Section 7.4.1 describes the case analysis and mission scenario. The Kripke models of the energy, fault event, USV decision-making system and GCS decision-making system are explained in Section 7.4.2. The properties to be verified and the setup of the model checker are introduced in Section 7.4.3. Section 7.4.3 also includes the verification result, the analysis and the discussion.

7.4.1 Case Analysis and Mission Scenario

The long endurance USV usually has multiple energy resources, including wind energy, solar energy or diesel energy. For example, the C-Enduro USV may utilize wind energy, solar energy and diesel energy. By measuring the battery level, the energy consumption states and the energy generation states, the USV may make the decision whether it should accelerate, keep the normal speed or station keeping. This decision-making system taking account of the energy generation and energy consumption will improve the mission execution capability and the endurance capability. In Section 7.4.1.1 and Section 7.4.1.2, the energy consumption and energy generation of the C-Enduro USV will be analysed, and the analysis results will be used for modelling the new behaviour of the USV decision-making system, which is introduced in Section 7.4.1.3.

7.4.1.1 C-Enduro USV Energy Consumption Analysis

The energy consumption of the C-Enduro USV under different speeds has been tested by ASV Global Ltd, and the relation between the energy consumption and the travelling speed is listed in Table. 7. 2. Note that the experimental data has been scaled-down as the data is confidential. The C-Enduro USV was tested in Portsmouth Harbour, and we assumed a sea current speed of 0 m/s when the vehicle was tested.

Table. 7. 2. C-Enduro USV energy consumption under different speeds

Speed (m/s)	0.5	0.8	1.2	1.6	2	2.4	2.6	2.8	3.0
Power Consumption (W)	1.9	4.8	8.7	15.4	23.0	36.8	51.5	75.0	100.0

The normal speed of the C-Enduro USV is 2m/s. From the data presented in Table 7.2, we conclude that when the USV is travelling at a high speed or in a counter-current environment, the USV consumes power very quickly; When the USV is travelling at a low speed or in a favourable sea current environment, the energy consumption will be lower. To verify the USV decision-making system, the model of the energy consumption must consider both the energy consumption environmental condition (mainly the sea current state), and the USV states.

7.4.1.2 C-Enduro USV Energy Generation Analysis

The specification of the C-Enduro USV energy generation capability summarized by ASV Ltd are listed in Table. 7. 3. From Table. 7. 3, we conclude that the diesel generator may generate higher power than the peak output of the wind turbine and the solar panel system, and that the wind turbine system and solar panel system are affected by environmental uncertainties. The diesel generator may be controlled by the local battery level. When the battery level is very low, the diesel generator turns on automatically; when the battery level is high, the diesel generator turns off automatically. As a result, the model of the total energy generation system should be affected by both the environmental uncertainties and the battery level.

Table. 7. 3. Energy generation capability of C-Enduro USV

Power System	Power Generated
Wind turbine system	Peak output power of 500W
Solar panel system	12 high efficiency panels generating a peak electrical power of 1200W
Diesel generator system	Peak charging power of 2.5kW

7.4.1.3 Mission Scenario and Decision-Making System Introduction

The mission scenario that is modelled in Section 7.3 is extended by taking account of the energy consumption and energy generation, shown in Fig. 7. 28. In this scenario, the USV should maximize the utilization of natural energy when natural energy generation is favourable. When the energy generation and battery level are low, the USV should reduce the amount of the energy consumption by staying at station keeping state and triggering the diesel generator to generate power.

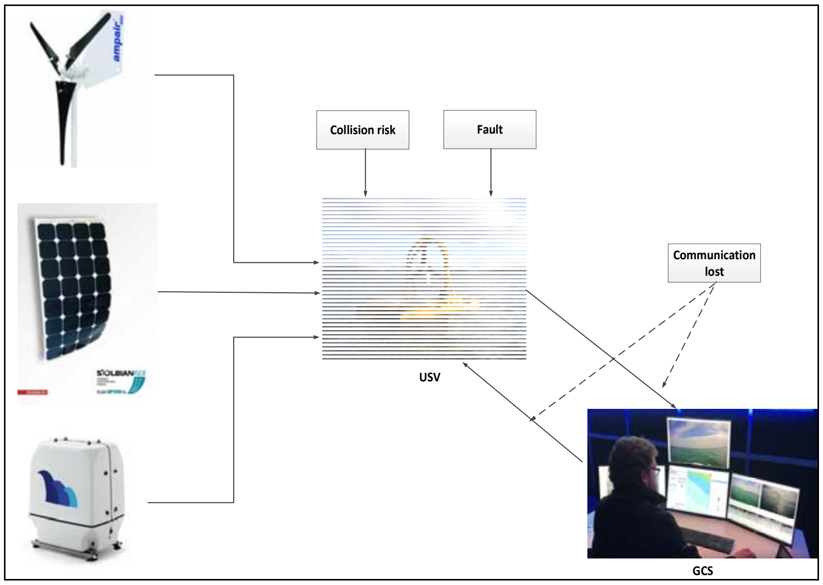


Fig. 7. 28. The USV mission with multiple energy resources

In this scenario, the environmental factors, including fault event, communication state, traffic information, energy consumption condition and energy generation condition are modelled as non-deterministic systems, resulting in the state at each moment having multiple potential transitions. The USV decision-making system is shown in Fig. 7. 29. The environmental factors determine the state transitions of the corresponding sensors, which eventually determine the state transitions of the USV.

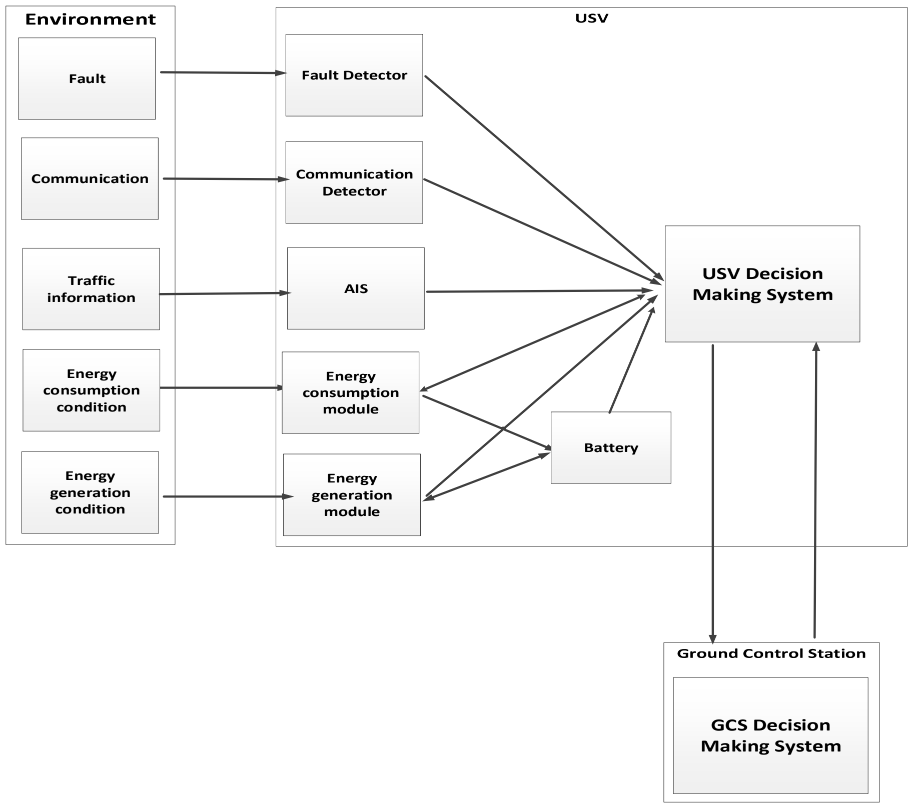


Fig. 7. 29. USV decision-making system taking account of energy

7.4.2 Kripke Modelling

The earlier USV scenario is extended by adding the energy generation condition, the energy generation module, the energy consumption condition, the energy consumption module and the battery. Environmental factors including the fault event, the communication state and the traffic information are taken into account. The previous fault event is extended by adding a severe fault event. The sensors, including the communication detector and the AIS, are the same as those used in the previous scenario. The Kripke models of the energy, the fault event, the USV and the GCS are described in this section.

7.4.2.1 Kripke Model for Energy

The USV energy model is comprised of five sub-models which include the energy generation condition model, the energy generation module model, the energy consumption condition model, the energy consumption module model and the battery model. Their relationships are shown in Fig. 7. 30.

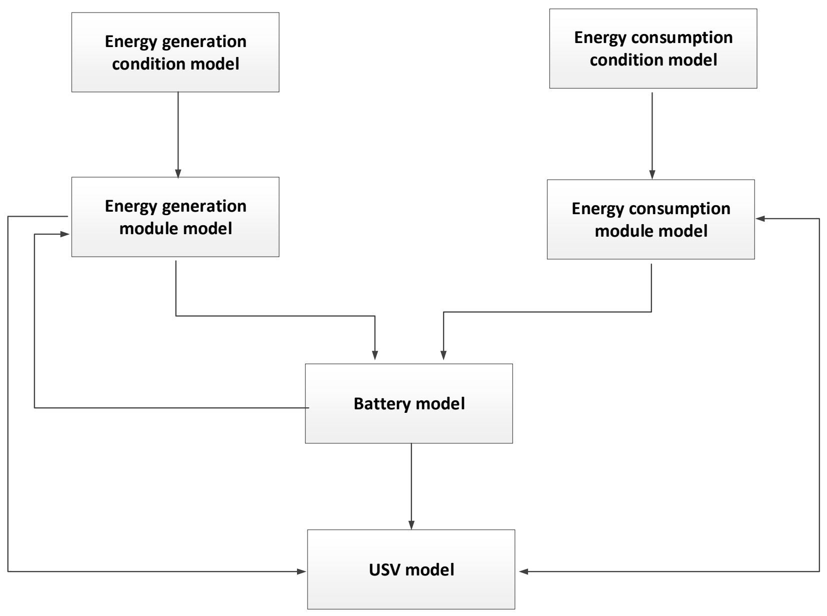

Energy generation condition model
Energy consumption condition model
Energy generation module model
Energy consumption module model
Battery model
USV model

Fig. 7. 30. USV energy model and sub-models

In Fig. 7. 30, the non-deterministic environmental factors may be modelled in the energy generation condition model and the energy consumption condition model. The energy generation condition model represents the uncertainties of solar irradiance and wind, which will affect the energy generation module model. The energy consumption condition model represents the uncertainty of the sea current, which will affect the energy consumption module model. The states of the solar panel, wind turbine and diesel generator are modelled in the energy generation module model, which will be affected by the energy generation condition and battery models. When the energy generation condition is favourable, the energy generation module will generate high power to the battery.

When the battery level is very low, it will trigger the diesel generator to start to generate power. Similarly, the energy consumption module is affected by the environmental energy consumption condition and the USV states. When the energy consumption condition is favourable, the energy will be consumed slowly, and when the USV is following a path at a high speed, the energy will be consumed quickly. The states of the energy generation module, the energy consumption module and the battery will also affect the USV states transition. For example, when energy generation and the energy consumption are favourable and the battery level is high, the USV will follow the path at high speed rather than at normal speed.

The energy model used in this scenario is proposed by abstracting the energy consumption and energy generation specifications of the C-Enduro USV, shown in Table. 7. 2 and Table. 7. 3. As accounting for all possible energy consumption and energy generation cases requires a lot of computational power, we simplified the energy consumption model, energy generation model and battery model by using integers to represent the amount of energy. The details are explained in Section (a), (b) and (c).

(a) Energy Generation

It is assumed that four energy generation conditions may appear in the USV mission period, which include the following: *Very Low Energy Generation Condition (VLEGC), Low Energy Generation Condition (LEGC), Medium Energy Generation Condition (MEGC)* and *High Energy Generation Condition (HEGC)*. The energy generation condition is non-deterministic, allowing it to transit from one state to its nearby state randomly, as shown in Fig. 7. 31.

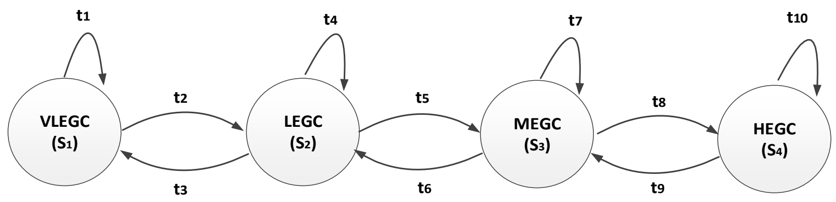


Fig. 7. 31. Kripke model for energy generation condition

Correspondingly, there are four states of the energy generation module, which include the following: *Very Low Energy Generation (VLEG), Low Energy Generation (LEG), Medium Energy Generation (MEG)* and *High Energy Generation (HEG)*. Four energy generation conditions will change four energy generation module states correspondingly. These four energy generation module states correspond to the amount of energy generation that will be added to the battery level. For convenience, the amount of energy generation is represented using integers, as shown in Table. 7. 4.

Table. 7. 4. The relationship between the energy generation condition, the energy generation state and the energy generation amount

Energy Generation Condition	*Very Low Energy Generation Condition (VLEGC)*	*Low Energy Generation Condition (LEGC)*	*Medium Energy Generation Condition (MEGC)*	*High Energy Generation Condition (HEGC)*
Energy Generation State	*Very Low Energy Generation (VLEG)*	*Low Energy Generation (LEG)*	*Medium Energy Generation (MEG)*	*High Energy Generation (HEG)*
Energy Generation Amount	0	+1	+2	+3

The descriptions in Table. 7. 4 are self-explanatory. For example, when the energy generation condition is in the *VLEGC* state, the energy generated will be

VLEG correspondingly and the energy added to the battery will be 0. Note that the diesel generator state is a special case: When the diesel generator is on, the energy generation state is always the *HEG* state, based on specification of the C-Enduro USV diesel generator.

(b) Energy Consumption

Referring to the specification of the C-Enduro USV energy consumption in Table. 7. 2, the energy consumption should be modelled considering the energy consumption condition (the sea current state) and the USV state.

The states of the energy consumption conditions, and the behaviours of the USV determine the states of the energy consumption module. It is assumed that there are three energy consumption conditions may appear in the USV mission, which include the following: *Low Energy Consumption Condition (LECC), Medium Energy Consumption Condition (MECC)* and *High Energy Consumption Condition (HECC)*. The energy consumption condition is modelled as a non-deterministic factor, as shown in Fig. 7. 32.

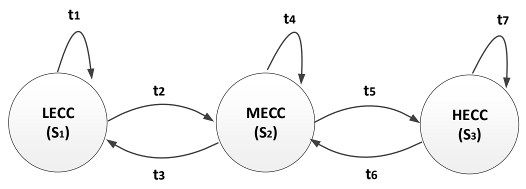


Fig. 7. 32. Kripke model for energy consumption condition

In terms of energy consumption, the behaviours of the USV may be classified into three groups, which include the following: *Low Energy Consumption Behaviour (LECB)*, including *Standby (SB), Ready (RE), Dispatched (DP), Station Keeping (SK), Arrive (AR)*; *Medium Energy Consumption Behaviour (MECB), including Path Following (PF) and Collision Avoidance (CA); and High Energy Consumption Behaviour (HECB), including Path Following in High Speed (PFH).*

Various combinations of the energy consumption conditions and the behaviours of the USV lead to different states of energy consumption, specially, the amount of battery level to be subtracted. The states of the energy consumption module are named as follows: *Very Low Energy Consumption (VLEC), Low Energy Consumption (LEC), Medium Energy Consumption (MEC), High Energy Consumption (HEC)* and *Very High Energy Consumption (VHEC).* The relationship among the energy consumption condition, the USV behaviour and the amount of energy consumption is listed in Table. 7. 5.

Table. 7. 5. The relation between the energy consumption condition, the USV behaviour and the energy consumption amount

Condition / Energy Consumption State/Amount / USV Behaviour	*Low Energy Consumption Condition (LECC)*	*Medium Energy Consumption Condition (MECC)*	*High Energy Consumption Condition (HECC)*
Low Energy Consumption Behaviour (LECB): Standby, Ready, Dispatched, Arrive, Station keeping.	*Very Low Energy Consumption (VLEC)* 0	*Low Energy Consumption (LEC)* -1	*Medium Energy Consumption (MEC)* -2
Medium Energy Consumption Behaviour (MECB): Path following, Collision Avoidance	*Low Energy Consumption (LEC)* -1	*Medium Energy Consumption (MEC)* -2	*High Energy Consumption (HEC)* -3
High Energy Consumption Behaviour (HECB): Path following in high speed	*Medium Energy Consumption (MEC)* -2	*High Energy Consumption (HEC)* -3	*Very High Energy Consumption (VHEC)* -4

(c) Battery

The battery level is represented using integers from 0 to 10. The sum of the energy consumption amount and the energy generation amount yields the changing battery level amount. For example, if the current state of the battery level is 5, the state of the energy generation module is *Low Energy Generation (LEG, +1)* and the state of the energy consumption module is *Medium Energy Consumption (MEC, -2),* then the next state of the battery level will be updated to 4 (5+1-2 = 4). When the battery level is 1, the diesel generator is triggered and when the battery level is above 2, the diesel generator is turned off. Note that the battery level will not be greater than 10, as we assume that 10 is the full state of the battery.

The states of the energy consumption module, the energy generation module and the battery level module will be used to improve the design of the decision-making system mentioned in the previous USV scenario.

7.4.2.2 Kripke Model for Fault Event

In this scenario, the fault event is extended by adding a severe fault event. In a long-range mission, the USV may malfunction but may still have the collision avoidance capability, which differs from the severe fault event where the USV cannot operate. The fault event includes non-fault, fault and severe fault, as shown in Fig. 7. 33. The severe fault means the USV cannot operate, and will go to standby state immediately. The fault event means the USV cannot execute a path following command, but still maintains the collision avoidance capability. In this situation, the USV will stay at station keeping state. “Non-fault” means the USV is in a normal operation state.

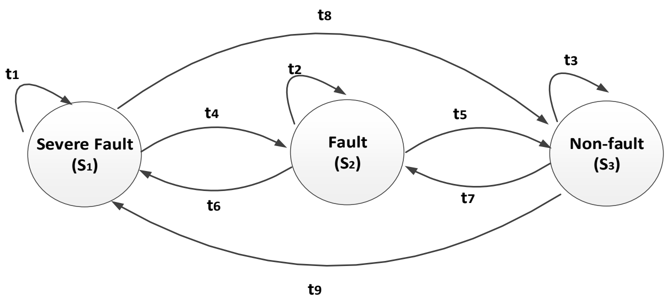


Fig. 7. 33. Kripke model for fault event

7.4.2.3 Kripke Model for USV

The behaviours of the USV are extended and the new behaviour is path following at high speed. The behaviours of the USV are defined as follows: *Standby (SB), Ready (RE), Dispatched (DP), Path Following (PF), Path Following in High Speed (PFH), Collision Avoidance (CA), Station Keeping (SK), Severe Fault (SFA)* and *Arrive (AR)*. In this mission scenario, the battery level is taken into account in the decision-making system. The relationships between the battery level and the USV behaviours are listed in Table. 7. 6. When the battery level is 0 or 1, the USV should be in *Standby* or *Severe Fault* state and the diesel generator will be triggered to generate power. When the battery level is 2, the USV may be in *Station Keeping* state, *Collision Avoidance* state, *Standby* state or *Severe Fault* state. When the battery level is above 3, the USV may be in *Path Following* state, *Station Keeping* state, *Collision Avoidance* state, *Ready state*, *Dispatched* state, *Standby* state and *Fault* state. When the battery level is above 9, the energy consumption condition is *Low Energy Consumption Condition,* and the energy generation condition is *High Energy Generation Condition*, the USV should be in *Path Following in High Speed* state if the communication state is normal and no give-way collision risk occurs. The *Path Following in High Speed* occurs when the battery level is high, and energy generation is higher than the energy consumption, allowing the high-speed path following state to maximize the utilization of the natural energy. The state transitions of USV are shown in Fig. 7. 34.

Table. 7. 6. The relationship between battery level and the USV behaviours

0	1	2	3	4	5	6	7	8	9	10
SB/SFA		*SK/CA/SB/SFA*	*PF/SK/CA/RE/DP/SB/SFA*						*PFH/PF/SK/CA/RE/DP/SB/SFA*	

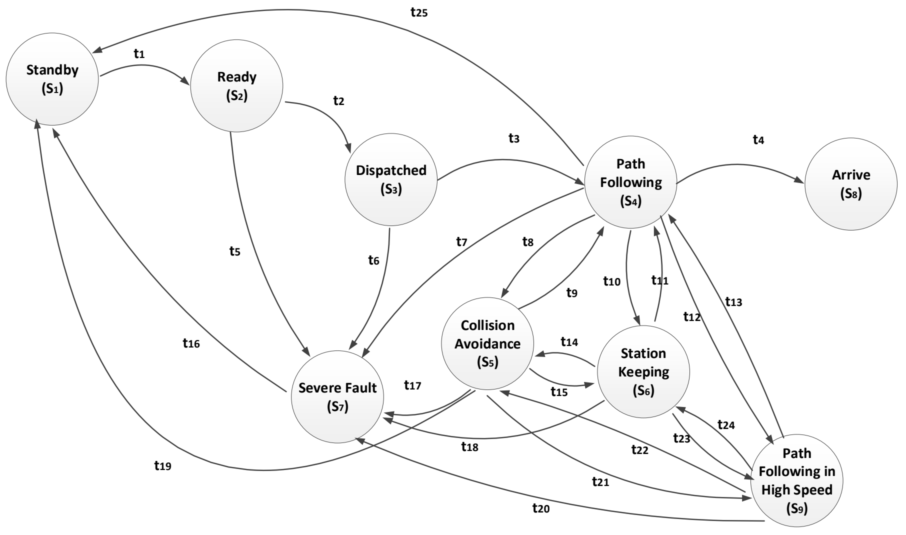


Fig. 7. 34. Kripke model for the USV

The accessibility relationships are described as follows:

- $(S_1 \rightarrow S_2)$: If the USV received the mission from the GCS, no fault is detected and battery level is above 2, this transition happens.

$$t_1 = \begin{cases} True, & If\ the\ USV\ received\ the\ mission\ \ and\ \ no\ fault\ detected \\ False, & Otherwise \end{cases}$$

- $(S_2 \rightarrow S_3)$: If the USV received the launching command from GCS and no fault is detected, this transition happens.

$$t_2 = \begin{cases} True, & If\ the\ USV\ received\ the\ mission \\ False, & Otherwise \end{cases}$$

- $(S_3 \rightarrow S_4)$: If the USV has been dispatched and no fault is detected, this transition happens.

$$t_3 = \begin{cases} True, & If\ the\ USV\ has\ been\ dispatched\ and\ no\ fault\ detected \\ False, & Otherwise \end{cases}$$

- $(S_4 \rightarrow S_8)$: If the USV arrived the destination, this transition happens.

$$t_4 = \begin{cases} True, & If\ the\ USV\ has\ arrived \\ False, & Otherwise \end{cases}$$

- $(S_5 \rightarrow S_4), (S_6 \rightarrow S_4), (S_9 \rightarrow S_4)$: If the USV is in path following (high speed) or collision avoidance or station keeping state, and no giving-way collision risk is detected and communication channel is in good status and no fault is detected and the battery level is above 2, these transitions happen.

$$t_9, t_{11}, t_{13} = \begin{cases} True,\ If\ the\ USV\ is\ in\ path\ following\ (high\ speed)\ or \\ collision\ avoidance\ or\ station\ keeping\ state, and\ no \\ giving-way\ collision\ risk\ found\ and\ communication \\ channel\ is\ in\ good\ status\ and\ no\ fault\ is\ detected\ and \\ the\ battery\ level\ is\ above\ 2 \\ \\ False, \quad Otherwise \end{cases}$$

- $(S_4 \rightarrow S_5), (S_6 \rightarrow S_5), (S_9 \rightarrow S_5)$: If the USV is in path following (normal speed or high speed) or station keeping state, and a giving-way collision risk is detected and communication channel is in good status and no fault is detected and the battery level is above 1, these transitions happen.

$$t_8, t_{14}, t_{22} = \begin{cases} True, \quad If\ the\ USV\ in\ path\ following\ (normal\ or\ high\ speed) \\ or\ station\ keeping\ state, and\ a\ giving-way \\ collision\ risk\ appears\ and\ communication \\ channel\ is\ in\ good\ status\ and\ no\ fault\ is\ detected\ and \\ the\ battery\ level\ is\ above\ 1. \\ \\ False, \quad Otherwise \end{cases}$$

- $(S_4 \rightarrow S_6), (S_5 \rightarrow S_6), (S_9 \rightarrow S_6)$: If the USV is in path following (normal speed or high speed) or collision avoidance, and no giving-way collision risk detected and communication channel is lost and no severe fault is detected and the battery level is above 1; Or fault event is detected and no collision risk is detected; Or no fault is detected and the battery level is 2 and no collision risk is detected; These transitions happen.

$$t_{10}, t_{15}, t_{24} = \begin{cases} True, & If\ the\ USV\ in\ path\ following\ or\ collision \\ & avoidance, and\ communication\ is\ lost \\ & and\ no\ giving-way\ collision\ risk\ appears \\ & and\ no\ fault\ detected\ and\ the\ battery\ level\ is \\ & above\ 1;\ \ Or\ fault\ event\ is\ detected \\ & and\ no\ collision\ risk\ is\ detected. \\ False, & Otherwise \end{cases}$$

- $(S_2 \to S_7), (S_3 \to S_7), (S_4 \to S_7), (S_5 \to S_7), (S_6 \to S_7), (S_9 \to S_7)$: If the USV has detected severe fault, these transitions happen.

$$t_5, t_6, t_7, t_{17}, t_{18}, t_{20} = \begin{cases} True, & If\ the\ USV\ has\ detected\ severe\ fault. \\ False, & Otherwise \end{cases}$$

- $(S_7 \to S_1)$: If the USV has been in fault state, this transition happens.

$$t_{16} = \begin{cases} True, & If\ the\ USV\ has\ been\ in\ fault\ state \\ False, & Otherwise \end{cases}$$

- $(S_5 \to S_1, S_4 \to S_1)$: If the USV battery level is 0 or 1, this transition happens.

$$t_{19}, t_{25} = \begin{cases} True, & If\ the\ USV\ battery\ level\ is\ 0\ or\ 1 \\ False, & Otherwise \end{cases}$$

- $(S_4 \to S_9), (S_5 \to S_9), (S_6 \to S_9)$: If there is no give-way collision risk, no fault is detected and battery is above 8 and the energy generation is higher than energy consumption, these transitions happen.

$$t_{12}, t_{21}, t_{23} = \begin{cases} True, \quad If\ there\ is\ no\ give-way\ collision\ risk\ detected, \\ no\ fault\ is\ detected, battery\ is\ above\ 8\ and\ the\ energy \\ \quad generation\ is\ higher\ than\ energy\ consumption \\ \\ False, \quad Otherwise \end{cases}$$

7.4.2.4 Kripke Model for GCS

The Kripke model for the GCS, including the transitions, is the same as the model described in the previous scenario. The Kripke model for the GCS in this scenario is shown in Fig. 7. 35.

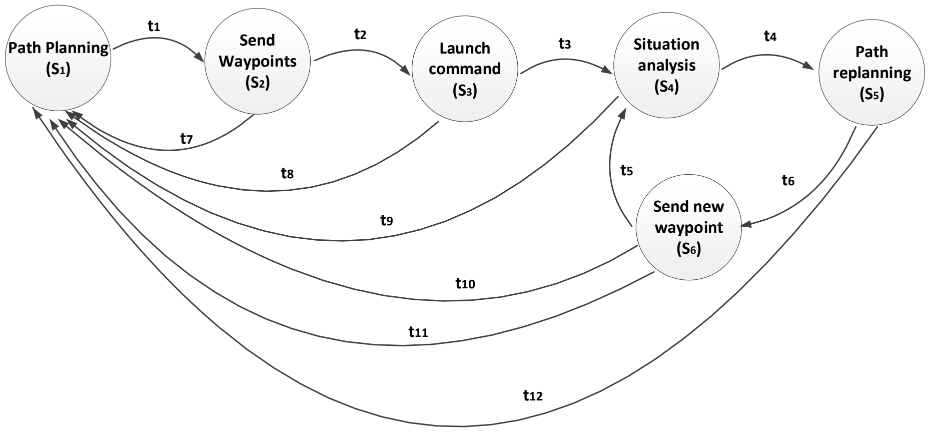


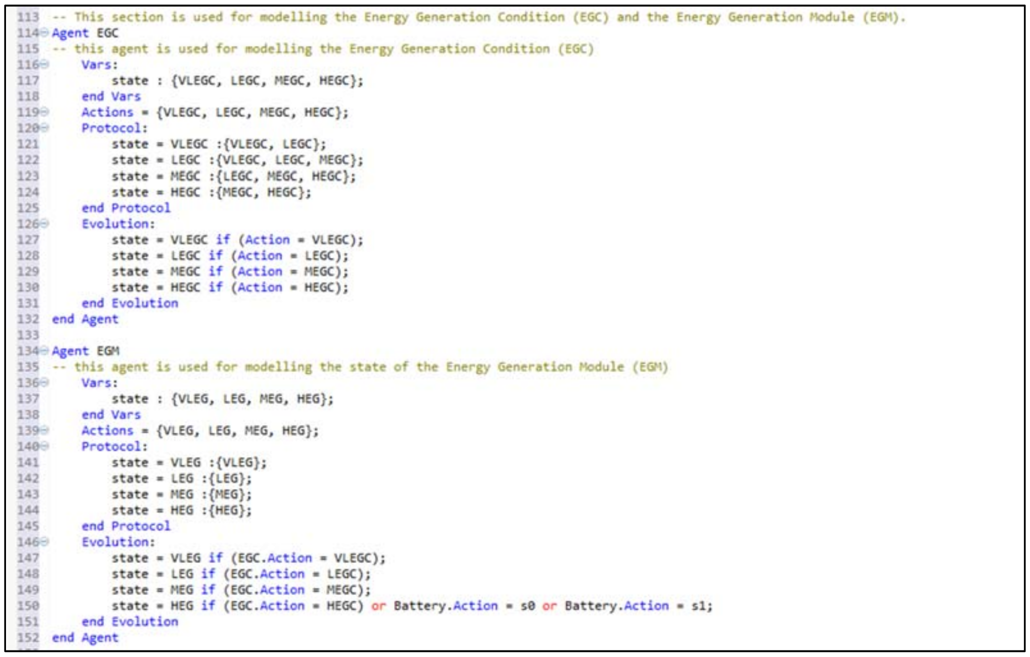
Fig. 7. 35. Kripke model for the GCS

7.4.3 Model Checking with MCMAS

In this section, the Kripke models were translated into ISPL code. The properties to be verified were expressed using CTL.

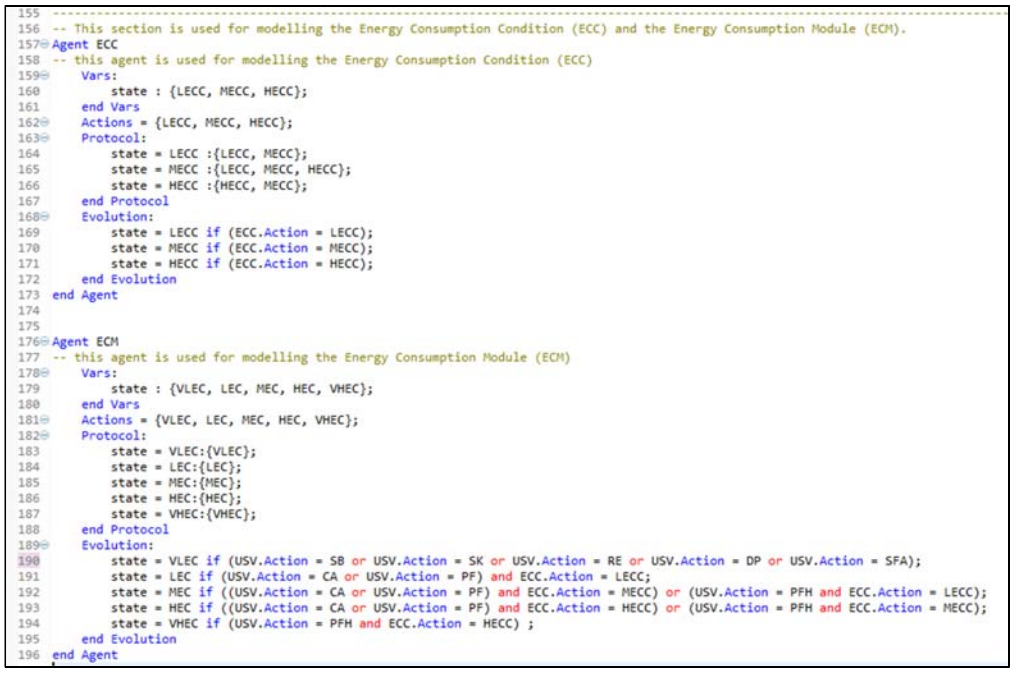
7.4.3.1 MCMAS Model

The ISPL code of energy generation condition and module is shown in Fig. 7. 36 The ISPL code of energy consumption condition and energy consumption module is shown in Fig. 7. 37. In the *Protocols* sections of Fig. 7. 36 and Fig. 7. 37, we see that the energy generation condition and energy consumption condition are non-deterministic models, because in each *State*, they may have random *Actions* and these *Actions* will determine the next *States* of the corresponding conditions. In Fig. 7. 36, the *States* of the energy generation module are determined not only by the *Action* of the energy generation condition but also by the *Action* of the battery, as the diesel generator turns on automatically when the battery is low. In Fig. 7. 37, we may see that the *States* of the energy consumption module are affected by the *Actions* of the energy consumption module and the USV.



```
-- This section is used for modelling the Energy Generation Condition (EGC) and the Energy Generation Module (EGM).
Agent EGC
-- this agent is used for modelling the Energy Generation Condition (EGC)
    Vars:
        state : {VLEGC, LEGC, MEGC, HEGC};
    end Vars
    Actions = {VLEGC, LEGC, MEGC, HEGC};
    Protocol:
        state = VLEGC :{VLEGC, LEGC};
        state = LEGC :{VLEGC, LEGC, MEGC};
        state = MEGC :{LEGC, MEGC, HEGC};
        state = HEGC :{MEGC, HEGC};
    end Protocol
    Evolution:
        state = VLEGC if (Action = VLEGC);
        state = LEGC if (Action = LEGC);
        state = MEGC if (Action = MEGC);
        state = HEGC if (Action = HEGC);
    end Evolution
end Agent

Agent EGM
-- this agent is used for modelling the state of the Energy Generation Module (EGM)
    Vars:
        state : {VLEG, LEG, MEG, HEG};
    end Vars
    Actions = {VLEG, LEG, MEG, HEG};
    Protocol:
        state = VLEG :{VLEG};
        state = LEG :{LEG};
        state = MEG :{MEG};
        state = HEG :{HEG};
    end Protocol
    Evolution:
        state = VLEG if (EGC.Action = VLEGC);
        state = LEG if (EGC.Action = LEGC);
        state = MEG if (EGC.Action = MEGC);
        state = HEG if (EGC.Action = HEGC) or Battery.Action = s0 or Battery.Action = s1;
    end Evolution
end Agent
```

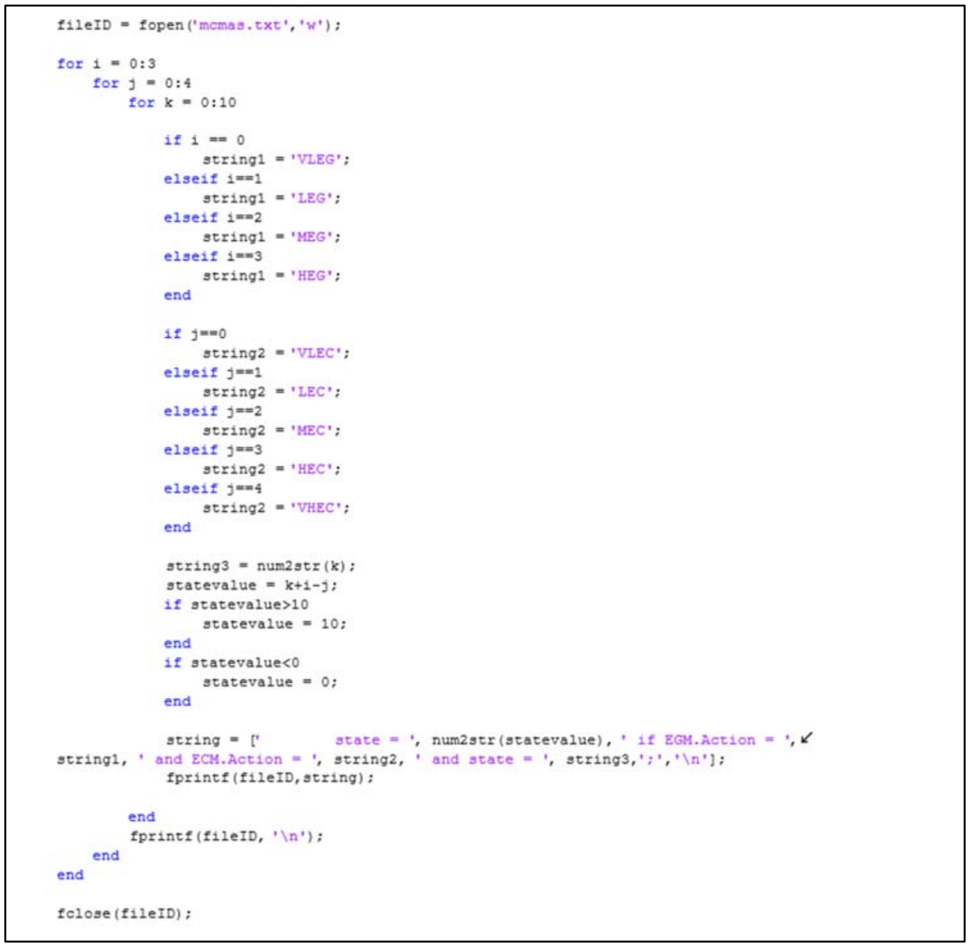
Fig. 7. 36. ISPL code of energy generation condition model and energy generation module model



```
-----------------------------------------------------------------------------------------------------------------
-- This section is used for modelling the Energy Consumption Condition (ECC) and the Energy Consumption Module (ECM).
Agent ECC
-- this agent is used for modelling the Energy Consumption Condition (ECC)
    Vars:
        state : {LECC, MECC, HECC};
    end Vars
    Actions = {LECC, MECC, HECC};
    Protocol:
        state = LECC :{LECC, MECC};
        state = MECC :{LECC, MECC, HECC};
        state = HECC :{HECC, MECC};
    end Protocol
    Evolution:
        state = LECC if (ECC.Action = LECC);
        state = MECC if (ECC.Action = MECC);
        state = HECC if (ECC.Action = HECC);
    end Evolution
end Agent


Agent ECM
-- this agent is used for modelling the Energy Consumption Module (ECM)
    Vars:
        state : {VLEC, LEC, MEC, HEC, VHEC};
    end Vars
    Actions = {VLEC, LEC, MEC, HEC, VHEC};
    Protocol:
        state = VLEC:{VLEC};
        state = LEC:{LEC};
        state = MEC:{MEC};
        state = HEC:{HEC};
        state = VHEC:{VHEC};
    end Protocol
    Evolution:
        state = VLEC if (USV.Action = SB or USV.Action = SK or USV.Action = RE or USV.Action = DP or USV.Action = SFA);
        state = LEC if (USV.Action = CA or USV.Action = PF) and ECC.Action = LECC;
        state = MEC if ((USV.Action = CA or USV.Action = PF) and ECC.Action = MECC) or (USV.Action = PFH and ECC.Action = LECC);
        state = HEC if ((USV.Action = CA or USV.Action = PF) and ECC.Action = HECC) or (USV.Action = PFH and ECC.Action = MECC);
        state = VHEC if (USV.Action = PFH and ECC.Action = HECC) ;
    end Evolution
end Agent
```

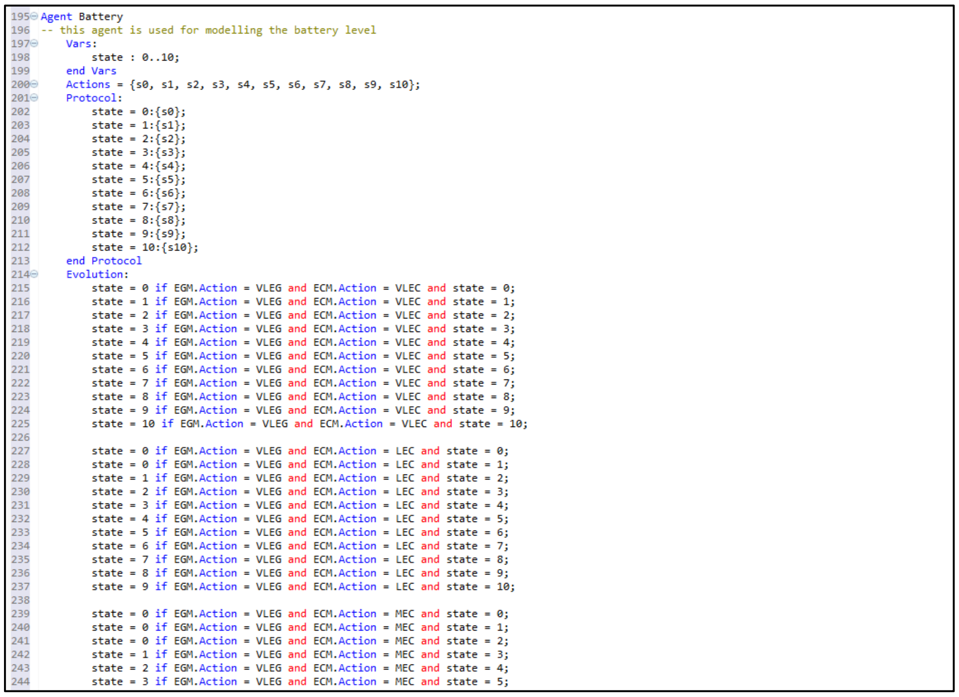
Fig. 7. 37. ISPL code of energy consumption condition model and energy consumption module model

The next state of the battery may be affected by the current state of the battery, the Action of the energy generation module and the Action of the energy consumption module. We used integers from 0 to 10 to represent the state of the battery, *VLEG*, *LEG*, *MEG* and *HEG* to represent the states of the energy generation module and *VLEC*, *LEC*, *MEC*, *HEC* and *VHEC* to represent the states of the energy consumption module. Therefore, we have 220 combinations to represent the Evolution of the battery state. Instead of writing them manually into the MCMAS model, a model code generator was developed using Matlab 2016b. The ISPL code generator is shown in Fig. 7. 38. Part of the generated ISPL code for the battery is shown in Fig. 7. 39. The ISPL code for the USV is shown in Fig. 7. 40.



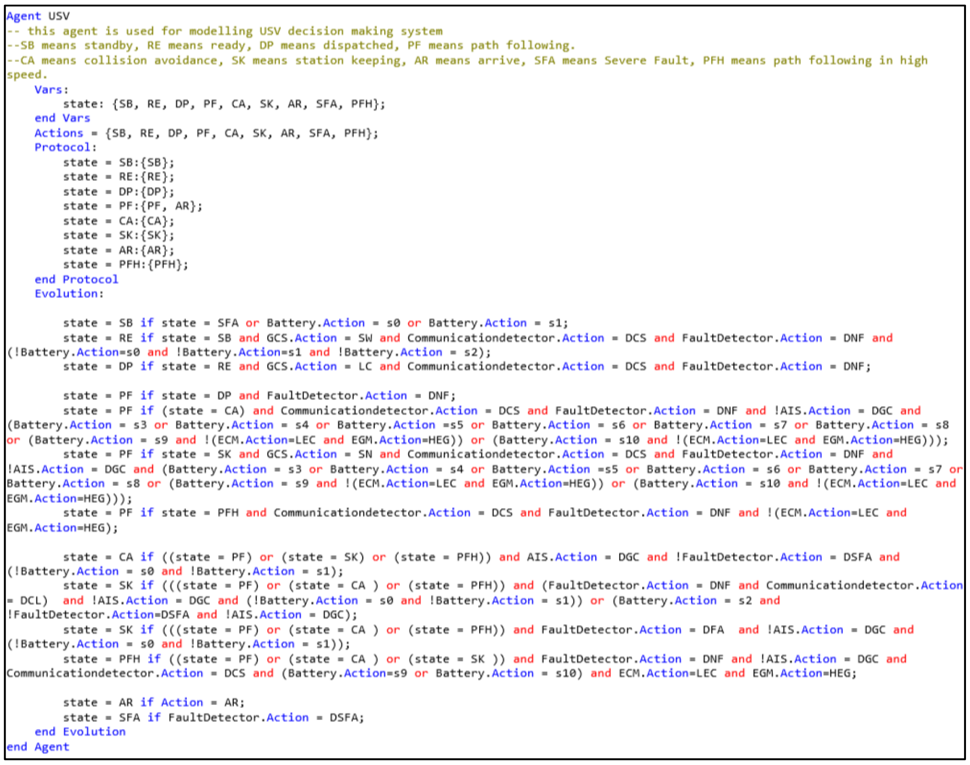
```
fileID = fopen('mcmas.txt','w');

for i = 0:3
    for j = 0:4
        for k = 0:10

            if i == 0
                string1 = 'VLEG';
            elseif i==1
                string1 = 'LEG';
            elseif i==2
                string1 = 'MEG';
            elseif i==3
                string1 = 'HEG';
            end

            if j==0
                string2 = 'VLEC';
            elseif j==1
                string2 = 'LEC';
            elseif j==2
                string2 = 'MEC';
            elseif j==3
                string2 = 'HEC';
            elseif j==4
                string2 = 'VHEC';
            end

            string3 = num2str(k);
            statevalue = k+i-j;
            if statevalue>10
                statevalue = 10;
            end
            if statevalue<0
                statevalue = 0;
            end

            string = ['        state = ', num2str(statevalue), ' if EGM.Action = ',↙
string1, ' and ECM.Action = ', string2, ' and state = ', string3,';','\n'];
            fprintf(fileID,string);

        end
        fprintf(fileID, '\n');
    end
end

fclose(fileID);
```

Fig. 7. 38. Battery ISPL code generation using Matlab code



```
Agent Battery
-- this agent is used for modelling the battery level
    Vars:
        state : 0..10;
    end Vars
    Actions = {s0, s1, s2, s3, s4, s5, s6, s7, s8, s9, s10};
    Protocol:
        state = 0:{s0};
        state = 1:{s1};
        state = 2:{s2};
        state = 3:{s3};
        state = 4:{s4};
        state = 5:{s5};
        state = 6:{s6};
        state = 7:{s7};
        state = 8:{s8};
        state = 9:{s9};
        state = 10:{s10};
    end Protocol
    Evolution:
        state = 0 if EGM.Action = VLEG and ECM.Action = VLEC and state = 0;
        state = 1 if EGM.Action = VLEG and ECM.Action = VLEC and state = 1;
        state = 2 if EGM.Action = VLEG and ECM.Action = VLEC and state = 2;
        state = 3 if EGM.Action = VLEG and ECM.Action = VLEC and state = 3;
        state = 4 if EGM.Action = VLEG and ECM.Action = VLEC and state = 4;
        state = 5 if EGM.Action = VLEG and ECM.Action = VLEC and state = 5;
        state = 6 if EGM.Action = VLEG and ECM.Action = VLEC and state = 6;
        state = 7 if EGM.Action = VLEG and ECM.Action = VLEC and state = 7;
        state = 8 if EGM.Action = VLEG and ECM.Action = VLEC and state = 8;
        state = 9 if EGM.Action = VLEG and ECM.Action = VLEC and state = 9;
        state = 10 if EGM.Action = VLEG and ECM.Action = VLEC and state = 10;

        state = 0 if EGM.Action = VLEG and ECM.Action = LEC and state = 0;
        state = 0 if EGM.Action = VLEG and ECM.Action = LEC and state = 1;
        state = 1 if EGM.Action = VLEG and ECM.Action = LEC and state = 2;
        state = 2 if EGM.Action = VLEG and ECM.Action = LEC and state = 3;
        state = 3 if EGM.Action = VLEG and ECM.Action = LEC and state = 4;
        state = 4 if EGM.Action = VLEG and ECM.Action = LEC and state = 5;
        state = 5 if EGM.Action = VLEG and ECM.Action = LEC and state = 6;
        state = 6 if EGM.Action = VLEG and ECM.Action = LEC and state = 7;
        state = 7 if EGM.Action = VLEG and ECM.Action = LEC and state = 8;
        state = 8 if EGM.Action = VLEG and ECM.Action = LEC and state = 9;
        state = 9 if EGM.Action = VLEG and ECM.Action = LEC and state = 10;

        state = 0 if EGM.Action = VLEG and ECM.Action = MEC and state = 0;
        state = 0 if EGM.Action = VLEG and ECM.Action = MEC and state = 1;
        state = 0 if EGM.Action = VLEG and ECM.Action = MEC and state = 2;
        state = 1 if EGM.Action = VLEG and ECM.Action = MEC and state = 3;
        state = 2 if EGM.Action = VLEG and ECM.Action = MEC and state = 4;
        state = 3 if EGM.Action = VLEG and ECM.Action = MEC and state = 5;
```

Fig. 7. 39. Part of the ISPL code of battery model



```
Agent USV
-- this agent is used for modelling USV decision making system
--SB means standby, RE means ready, DP means dispatched, PF means path following.
--CA means collision avoidance, SK means station keeping, AR means arrive, SFA means Severe Fault, PFH means path following in high
speed.
    Vars:
        state: {SB, RE, DP, PF, CA, SK, AR, SFA, PFH};
    end Vars
    Actions = {SB, RE, DP, PF, CA, SK, AR, SFA, PFH};
    Protocol:
        state = SB:{SB};
        state = RE:{RE};
        state = DP:{DP};
        state = PF:{PF, AR};
        state = CA:{CA};
        state = SK:{SK};
        state = AR:{AR};
        state = PFH:{PFH};
    end Protocol
    Evolution:

        state = SB if state = SFA or Battery.Action = s0 or Battery.Action = s1;
        state = RE if state = SB and GCS.Action = SW and Communicationdetector.Action = DCS and FaultDetector.Action = DNF and
(!Battery.Action=s0 and !Battery.Action=s1 and !Battery.Action = s2);
        state = DP if state = RE and GCS.Action = LC and Communicationdetector.Action = DCS and FaultDetector.Action = DNF;

        state = PF if state = DP and FaultDetector.Action = DNF;
        state = PF if (state = CA) and Communicationdetector.Action = DCS and FaultDetector.Action = DNF and !AIS.Action = DGC and
(Battery.Action = s3 or Battery.Action = s4 or Battery.Action =s5 or Battery.Action = s6 or Battery.Action = s7 or Battery.Action = s8
or (Battery.Action = s9 and !(ECM.Action=LEC and EGM.Action=HEG)) or (Battery.Action = s10 and !(ECM.Action=LEC and EGM.Action=HEG)));
        state = PF if state = SK and GCS.Action = SN and Communicationdetector.Action = DCS and FaultDetector.Action = DNF and
!AIS.Action = DGC and (Battery.Action = s3 or Battery.Action = s4 or Battery.Action =s5 or Battery.Action = s6 or Battery.Action = s7 or
Battery.Action = s8 or (Battery.Action = s9 and !(ECM.Action=LEC and EGM.Action=HEG)) or (Battery.Action = s10 and !(ECM.Action=LEC and
EGM.Action=HEG)));
        state = PF if state = PFH and Communicationdetector.Action = DCS and FaultDetector.Action = DNF and !(ECM.Action=LEC and
EGM.Action=HEG);

        state = CA if ((state = PF) or (state = SK) or (state = PFH)) and AIS.Action = DGC and !FaultDetector.Action = DSFA and
(!Battery.Action = s0 and !Battery.Action = s1);
        state = SK if (((state = PF) or (state = CA ) or (state = PFH)) and (FaultDetector.Action = DNF and Communicationdetector.Action
= DCL)  and !AIS.Action = DGC and (!Battery.Action = s0 and !Battery.Action = s1)) or (Battery.Action = s2 and
!FaultDetector.Action=DSFA and !AIS.Action = DGC);
        state = SK if (((state = PF) or (state = CA ) or (state = PFH)) and FaultDetector.Action = DFA  and !AIS.Action = DGC and
(!Battery.Action = s0 and !Battery.Action = s1));
        state = PFH if ((state = PF) or (state = CA ) or (state = SK )) and FaultDetector.Action = DNF and !AIS.Action = DGC and
Communicationdetector.Action = DCS and (Battery.Action=s9 or Battery.Action = s10) and ECM.Action=LEC and EGM.Action=HEG;

        state = AR if Action = AR;
        state = SFA if FaultDetector.Action = DSFA;
    end Evolution
end Agent
```

Fig. 7. 40. ISPL code of USV model

7.4.3.2 Modelling of Properties to be Verified

There are 14 properties to be verified. These properties and corresponding CTL formulas are shown as follows:

1, 'After the USV received the mission (ready state), if the communication state is good and no fault is detected by the USV, the USV battery level is above 2, then the GCS sends the launching command, and the USV will be dispatched'

$$AG((USV.state = RE \wedge Communicationdetector.state = DCS \wedge Faultdetector.state = DNF \wedge GCS.state = LC \wedge Battery.state > 2) \rightarrow AX(USV.state = DP))$$

2, 'When the USV is in the path following state and no fault is detected, the USV battery level is above 2, then a give-way collision risk is detected, the USV will either change way to avoid the collision or arrive at the destination'.

$$AG((USV.state = PF \wedge Faultdetector.state = DNF \wedge Battery.state > 2 \wedge AIS.state = DGC) \rightarrow AX(USV.state = CA \vee USV.state = AR))$$

3, '*If the give-way collision does not appear, the USV will never alter its way to avoid the collision'.*

$$AG(!AIS.state = DGC \rightarrow AX(!USV.state = CA))$$

4, 'After the USV avoided the collision risk, if the communication state is good, no give-way collision risk is detected and no fault is detected, and the USV battery level is above 2, then the USV will always continue to follow the path in normal speed'.

$$AG((USV.state = CA \wedge Communicationdetector.state = DCS \wedge !AIS.state = DGS \wedge Battery.state > 2 \wedge Faultdetector.state = DNF) \rightarrow AX(USV.state = PF))$$

5, *'When the USV is in the station keeping state and no fault is detected and the GCS is in the situation analysis state, if good communication state is detected, the GCS will always re-plan the path'.*

$$AG((USV.state = SK \wedge Faultdetector.state = DNF \wedge GCS.state = SiA \wedge Communicationdetector.state = DCS) \rightarrow AX(GCS.state = PR))$$

6, *'When the USV is in path following state, and there is no give-way collision risk detected and no fault event is detected, the USV battery level is above 2, if the communication is lost, the USV will either change to station keeping state or arrive at the destination'.*

$$AG((USV.state = PF \wedge Faultdetector.state = DNF \wedge !AIS.state = DGC \wedge Battery.sgtate > 2 \wedge Communicationdetector.state = DCL) \rightarrow AX(USV.state = SK \vee USV.state = AR))$$

7, *'When the USV is in station keeping state, if the communication state is good and the USV battery level is above 2, after the GCS sends the new waypoints, the USV will change to path following state'.*

$$AG((USV.state = SK \wedge Communicationdetector.state = DCS \wedge !AIS.state = DGC \wedge Battery.state > 2 \wedge GCS.state = SN) \rightarrow AX(USV.state = PF))$$

8, 'When the USV is in the path following state and no fault is detected, if the communication loss is detected, the USV will change to station keeping state'.

$$AG((USV.state = PF \wedge Faultdetector.state = DNF \wedge Communicationdetector.state = DCL) \rightarrow AX(USV.state = SK))$$

9, 'If the communication is not lost or the USV battery level is not below 3 or there is not fault detected, the USV will not change to station keeping state'

$$AG(!(Communicaitondetector.state = DCL \vee Battery.state < 3 \vee Faultdetector.state = DFA)) \rightarrow AF(!USV.state = SK)$$

10, '*If the communication is lost, the USV will change to standby or station keeping state'.*

$$AG(Communicaitondetector.state = DCL) \rightarrow AF(USV.state = SB\ or\ USV.state = SK)$$

11, 'If severe fault event is detected, the USV will transit to standby state directly'

$$AG(Faultdetector.state = DSFA \rightarrow AX(USV.state = SB))$$

12, 'If the battery level is less than 2, the USV will not following the path'

$$AG(Battery.state < 2 \rightarrow AX(!\,USV.state = PF))$$

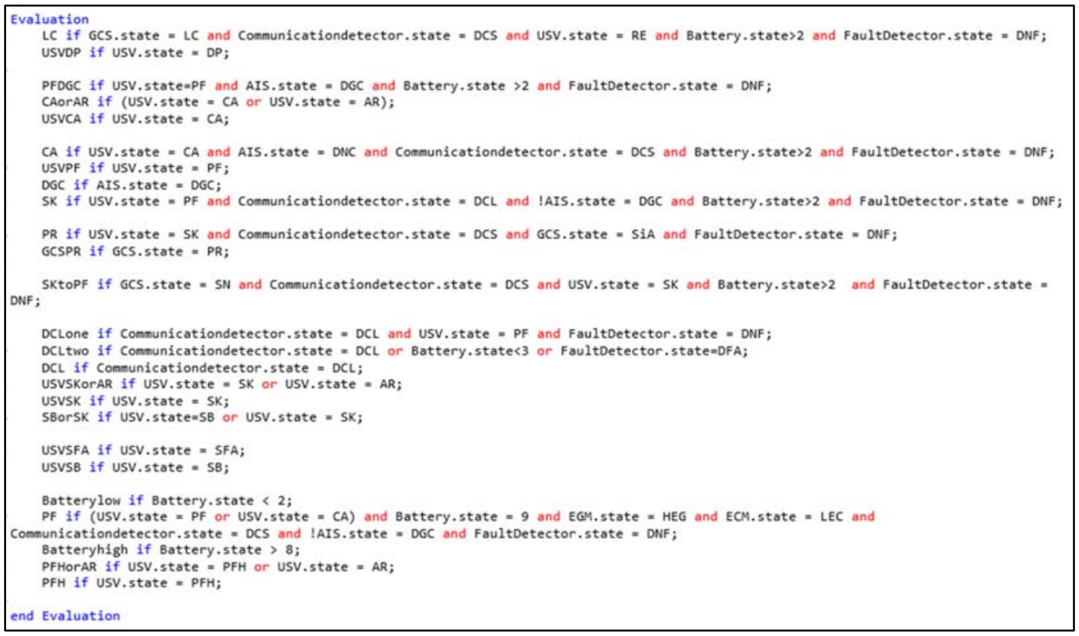
13, 'When the USV is in path following or collision avoidance state, battery level is 9, energy generation is high, energy consumption is low, no give-way collision risk is detected, no fault is detected, the USV will either follow the path at high speed or arrive at the destination'.

$$AG((USV.state = PF \vee USV.state = CA\,) \wedge Battery.state = 9\ \wedge EGM.state = HEG\ \wedge ECM.state = LEC\ \wedge Communicationdetector.state = DCS \wedge !\,AIS.state = DGC \wedge Faultdetector.state = DNF \rightarrow AX(USV.state = PFH \vee USV.state = AR))$$

14, 'If the battery level is not above 8, the USV will never follow the path at high speed.'

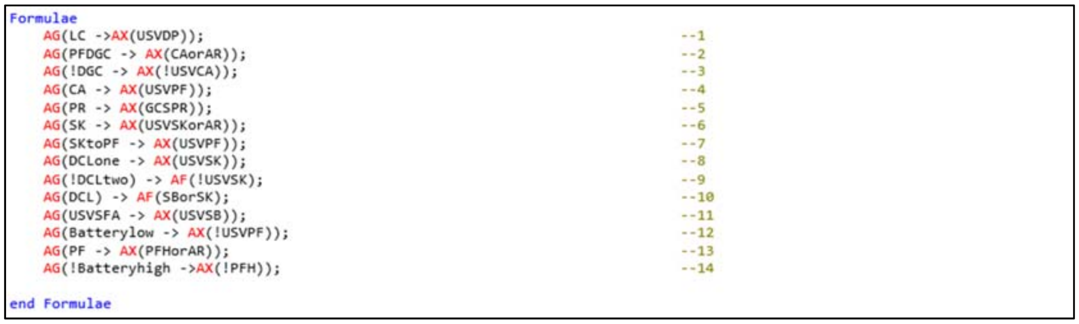
$$AG(!\,Battery.state > 8 \rightarrow AX(!\,USV.state = PFH))$$

The properties are expressed in the *Evaluation* and *Formulae* sections, as shown in Fig. 7. 41 and Fig. 7. 42.



```
Evaluation
    LC if GCS.state = LC and Communicationdetector.state = DCS and USV.state = RE and Battery.state>2 and FaultDetector.state = DNF;
    USVDP if USV.state = DP;

    PFDGC if USV.state=PF and AIS.state = DGC and Battery.state >2 and FaultDetector.state = DNF;
    CAorAR if (USV.state = CA or USV.state = AR);
    USVCA if USV.state = CA;

    CA if USV.state = CA and AIS.state = DNC and Communicationdetector.state = DCS and Battery.state>2 and FaultDetector.state = DNF;
    USVPF if USV.state = PF;
    DGC if AIS.state = DGC;
    SK if USV.state = PF and Communicationdetector.state = DCL and !AIS.state = DGC and Battery.state>2 and FaultDetector.state = DNF;

    PR if USV.state = SK and Communicationdetector.state = DCS and GCS.state = SiA and FaultDetector.state = DNF;
    GCSPR if GCS.state = PR;

    SKtoPF if GCS.state = SN and Communicationdetector.state = DCS and USV.state = SK and Battery.state>2  and FaultDetector.state =
DNF;

    DCLone if Communicationdetector.state = DCL and USV.state = PF and FaultDetector.state = DNF;
    DCLtwo if Communicationdetector.state = DCL or Battery.state<3 or FaultDetector.state=DFA;
    DCL if Communicationdetector.state = DCL;
    USVSKorAR if USV.state = SK or USV.state = AR;
    USVSK if USV.state = SK;
    SBorSK if USV.state=SB or USV.state = SK;

    USVSFA if USV.state = SFA;
    USVSB if USV.state = SB;

    Batterylow if Battery.state < 2;
    PF if (USV.state = PF or USV.state = CA) and Battery.state = 9 and EGM.state = HEG and ECM.state = LEC and
Communicationdetector.state = DCS and !AIS.state = DGC and FaultDetector.state = DNF;
    Batteryhigh if Battery.state > 8;
    PFHorAR if USV.state = PFH or USV.state = AR;
    PFH if USV.state = PFH;

end Evaluation
```

Fig. 7. 41. ISPL code for Evaluation

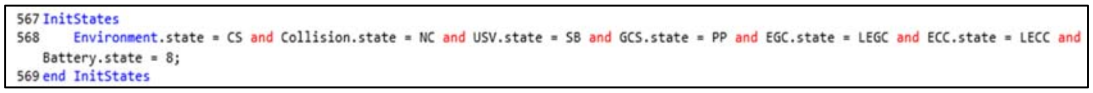


```
Formulae
    AG(LC ->AX(USVDP));                                                        --1
    AG(PFDGC -> AX(CAorAR));                                                   --2
    AG(!DGC -> AX(!USVCA));                                                    --3
    AG(CA -> AX(USVPF));                                                       --4
    AG(PR -> AX(GCSPR));                                                       --5
    AG(SK -> AX(USVSKorAR));                                                   --6
    AG(SKtoPF -> AX(USVPF));                                                   --7
    AG(DCLone -> AX(USVSK));                                                   --8
    AG(!DCLtwo) -> AF(!USVSK);                                                 --9
    AG(DCL) -> AF(SBorSK);                                                     --10
    AG(USVSFA -> AX(USVSB));                                                   --11
    AG(Batterylow -> AX(!USVPF));                                              --12
    AG(PF -> AX(PFHorAR));                                                     --13
    AG(!Batteryhigh ->AX(!PFH));                                               --14

end Formulae
```

Fig. 7. 42. ISPL code for Formulae

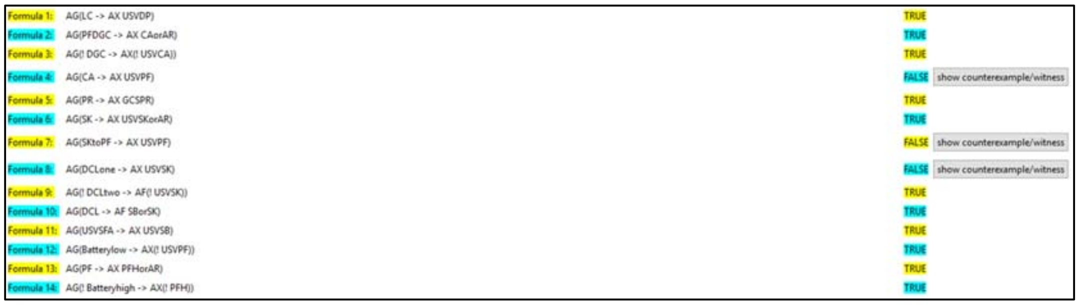
7.4.3.3 Verification result

The *InitStates* section of the code is shown in Fig. 7. 43. The verification results are shown in Fig. 7. 44.



```
InitStates
    Environment.state = CS and Collision.state = NC and USV.state = SB and GCS.state = PP and EGC.state = LEGC and ECC.state = LECC and
   Battery.state = 8;
end InitStates
```

Fig. 7. 43. ISPL code for InitStates



```
Formula 1:  AG(LC -> AX USVDP)                    TRUE
Formula 2:  AG(PFDGC -> AX CAorAR)                TRUE
Formula 3:  AG(! DGC -> AX(! USVCA))              TRUE
Formula 4:  AG(CA -> AX USVPF)                    FALSE  show counterexample/witness
Formula 5:  AG(PR -> AX GCSPR)                    TRUE
Formula 6:  AG(SK -> AX USVSKorAR)                TRUE
Formula 7:  AG(SKtoPF -> AX USVPF)                FALSE  show counterexample/witness
Formula 8:  AG(DCLone -> AX USVSK)                FALSE  show counterexample/witness
Formula 9:  AG(! DCLtwo -> AF(! USVSK))           TRUE
Formula 10: AG(DCL -> AF SBorSK)                  TRUE
Formula 11: AG(USVSFA -> AX USVSB)                TRUE
Formula 12: AG(Batterylow -> AX(! USVPF))         TRUE
Formula 13: AG(PF -> AX PFHorAR)                  TRUE
Formula 14: AG(! Batteryhigh -> AX(! PFH))        TRUE
```

Fig. 7. 44. Verification results

7.4.3.4 Analysis and Discussion

The property of Formula 4 reads, '*After the USV avoided the collision risk, if the communication state is good, no give-way collision risk is detected and no fault is detected, the USV battery level is above 2, then the USV will always continue to follow the path in normal speed*'. By checking the counterexample, we found the USV transits to path following at high speed instead of path following at normal speed, because the record shows that the battery level was 10 and the energy generation was higher than energy consumption. Therefore, in this situation, it is reasonable to accelerate rather than travelling at normal speed which will maximize the utilization of the natural energy.

The property of Formula 7 is '*When the USV is in station keeping state, if the communication state is good and the USV battery level is above 2, after the GCS send the new waypoints, the USV will change to path following state*'. The record shows that after the GC sent the new waypoint, the USV detected a collision risk, so it transited to collision avoidance state instead of following the path and this may protect the safety of the USV. This counterexample is reasonable.

The property of Formula 8 is '*When the USV is in the path following state and no fault is detected, if the communication loss is detected, the USV will change to station keeping state*'. The records show that when the USV is following the path, and the communication is lost and it also detected the give-way collision risk at the same time, it will transit to collision avoidance state instead of the station keeping state. This counterexample obeys the safety design of the decision-making system.

The result of Formula 1 shows the USV may be dispatched successfully. The results of Formulas 2 and 3 show that the decision-making system may perform the collision avoidance behaviour successfully, which verified the corresponding safety properties. The results of Formulas 4, 12, 13 and 14 proved the long endurance properties of the decision-making system. The Formula 12 shows that when the battery is low, the USV will not path the path and it will stay in low energy consumption state and wait for energy generation. Formulas 13 and 14 show that only when the battery level is high and the energy generation is higher than the energy consumption, the USV will accelerate. The result of Formula 5 shows the GCS may perform the path re-planning properly. The results of Formulas 6, 9, and 10 demonstrate that the decision-making system can perform the station keeping behaviour successfully. The result of Formula 11 shows that when the severe fault event is detected, the USV will stay at standby mode, which is a safety property.

7.5 Multiple Autonomous Vehicles Mission with Fault Tolerance Capability and Mesh Network

7.5.1 Mission Scenario

Multiple autonomous agents are used to construct a redundant system in this scenario. Fault tolerance is an important property for improving the mission completion, achieved by using another USV to replace the faulty USV. Another improved feature is the implementation of the mesh network. Reliable communication is a key factor in maintaining the safety of the USV, especially in a long-range mission. Wireless mesh network (WMNs) are dynamically self-organized and self-configured, with nodes in the network automatically establishing an ad hoc network and maintaining the mesh connectivity. Mesh network topology may increase the signal coverage area and offers communication redundancy. The mesh network has the following advantages:

First, a broken node will not distract the transmission of data in a mesh network. A broken device will be ignored by the signals, and find a new device connected with the node autonomously. Second, additional devices in a mesh topology will not affect network connections. Finally, a mesh network may handle high volumes of network traffic as each additional device in the network is considered a node.

The long endurance USV is commanded to collect data in a long-range area. Another regular USV and UAV are sent to route the signal between the long endurance USV and the GCS to increase the communication range and improve the reliability of the communication. The long endurance USV (USV1), the regular USV (USV2), the UAV and the GCS are configured as communication nodes for routing signals. The regular USV is responsible for signal routing when the long endurance USV is operating normally, and replaces the long endurance USV to continue the mission when the long endurance USV loses signal for a long time or has a fault event. If the long endurance USV or the regular USV arrive the destination, the UAV should return to launch. However, if the UAV also has fault or communication loss with the GCS, it is assumed that it will land on the sea. The scenario is shown in Fig. 7. 45. The details of modelling the USVs, the UAV, the GCS and the mesh network are described in Section 7.5.2.

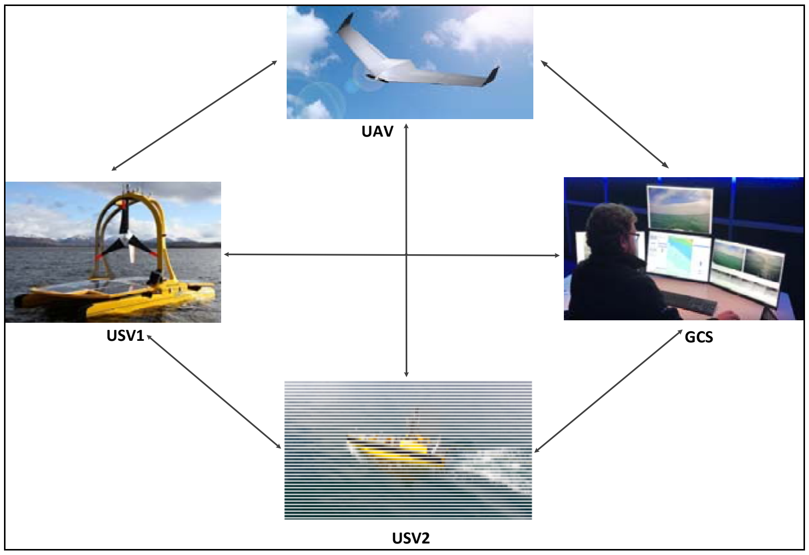


Fig. 7. 45. Mission scenario with multiple autonomous vehicles

7.5.2 Kripke Modelling

7.5.2.1 Kripke Model of Long Endurance USV

During the long endurance USV missions, a distinction logic between the intermittent and permanent communication losses is required to maximise the success of mission completion (Choi, et al., 2015). In this mission scenario, when lost communication between the USV and the GCS has been detected continuously three times, it is assumed to be lost permanently, and severthe regular USV commanded to take place of the long endurance USV.

It is also assumed that the long endurance USV does not have the capability to recover from the fault event, modifying the fault event model. The fault state cannot transit to the non-fault state, and may transit to the fault state and the severe fault state. When the long endurance USV is confirmed to be lost permanently or have a fault or severe fault event, it will stay at the station keeping state (fault event) or collision avoidance state when the collision is detected (fault event) or standby (severe event). In this situation, the regular USV will take the place of the long endurance USV to continue the mission. The accessibility relationships of the long endurance USV model are the same as those described for the model in Section 7.4, as shown in Fig. 7. 46.

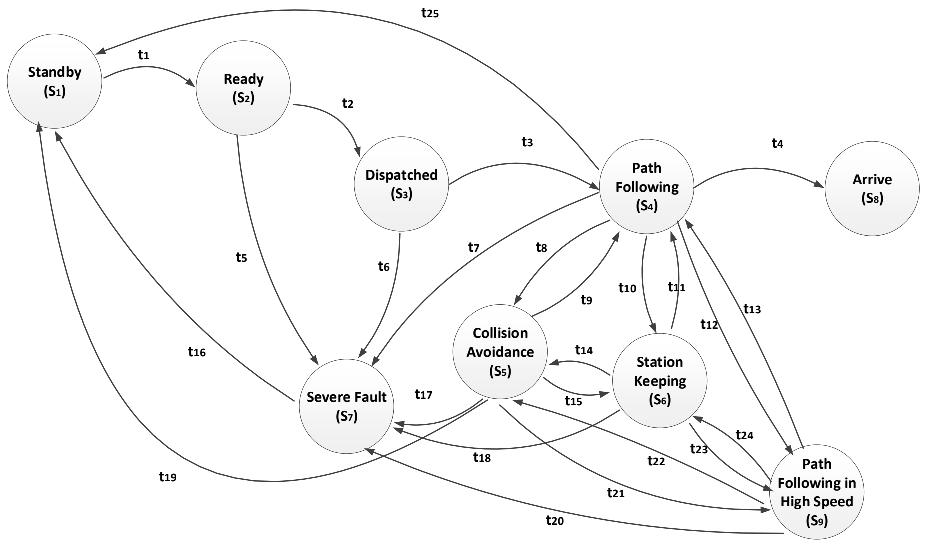


Fig. 7. 46. Kripke model for long endurance USV (USV1)

7.5.2.2 Kripke model of regular USV

The regular USV, which has no additional energy resources, is expected to travel to the designated position to route the signal and take place of the long endurance USV in case the long endurance USV is lost permanently or has fault. If the long endurance USV completes the mission, the regular USV should return to launch. Therefore, the Kripke model of the regular USV is built by modifying the Kripke model of the USV we built in Section 7.3, by adding the routing signal state and the return to launch state, as shown in Fig. 7. 47.

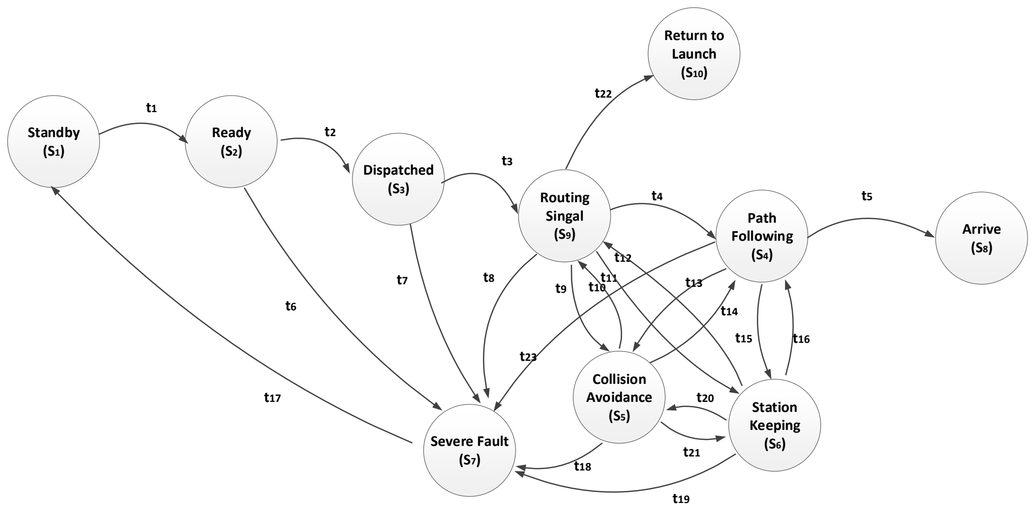


Fig. 7. 47. Kripke model for the regular USV (USV2)

The accessibility relationships are described as follows:

- $(S_1 \rightarrow S_2)$: If USV1 is in ready state and the communication between USV1 and USV2 is normal, this transition happens.

$$t_1 = \begin{cases} True, & If\ USV1\ is\ in\ ready\ state\ and\ the\ communication \\ & between\ USV1\ and\ USV2\ is\ normal \\ False, & Otherwise \end{cases}$$

- $(S_2 \rightarrow S_3)$: If USV1 is in dispatched state and the communication between USV1 and USV2 is normal, this transition happens.

$$t_2 = \begin{cases} True, & If\ USV1\ is\ in\ dispatched\ state\ and\ the\ communication \\ & \quad between\ the\ USV1\ and\ USV2\ is\ normal \\ False, & Otherwise \end{cases}$$

- $(S_3 \rightarrow S_9)$: If USV1 is in path following state and the communication between USV1 and USV2 is normal, USV2 will analyse the mission of USV1 and travel to the designated position and routing signal, then this transition happens.

$$t_3 = \begin{cases} True, & If\ USV1\ is\ in\ path\ following\ state\ and\ the \\ & communication\ between\ the\ USV1\ and\ USV2\ is\ normal \\ False, & Otherwise \end{cases}$$

- $(S_9 \rightarrow S_{10})$: If USV1 arrives at the destination and the communication between USV1 and USV2 is normal, this transition happens.

$$t_{22} = \begin{cases} True, & If\ USV1\ arrives\ the\ destination\ and\ the\ communication \\ & \quad between\ the\ USV1\ and\ USV2\ is\ normal \\ False, & Otherwise \end{cases}$$

- $(S_5 \rightarrow S_4), (S_6 \rightarrow S_4), (S_9 \rightarrow S_4)$: If USV1 has a fault event or is lost permanently, no give-way collision risk is detected and the communication channel between USV1 and USV2 is in good status and no fault is detected, these transitions happen.

$$t_{14}, t_{16}, t_4 = \begin{cases} True, & If\ USV1\ has\ fault\ event\ or\ is\ lost\ permanently \\ no\ give-way\ collision\ risk\ found\ and\ communication \\ \quad channel\ is\ in\ good\ status\ and\ no\ fault\ is\ detected \\ False, & Otherwise \end{cases}$$

- $(S_4 \rightarrow S_5), (S_6 \rightarrow S_5), (S_9 \rightarrow S_5)$: If a give-way collision risk is detected and no fault is detected, these transitions happen.

$$t_{13}, t_{20}, t_9 = \begin{cases} True, & If\ a\ give-way\ collision\ risk\ is\ detected\ and \\ & \quad and\ no\ fault\ is\ detected \\ False, & Otherwise \end{cases}$$

- $(S_4 \rightarrow S_6), (S_5 \rightarrow S_6), (S_9 \rightarrow S_6)$: If no give-way collision risk appears and the communication channel between USV2 and the GCS is lost or fault is detected, these transitions happen.

$$t_{15}, t_{21}, t_{11} = \begin{cases} True, & If\ no\ give-way\ collision\ risk\ appears\ and \\ & communication\ channel\ between\ the\ USV2\ and\ the \\ & GCS\ is\ lost\ or\ fault\ is\ detected \\ False, & Otherwise \end{cases}$$

- $(S_5 \rightarrow S_9), (S_6 \rightarrow S_9)$: If there is no fault of USV1 and it is not lost permanently and there is no give-way collision and no fault of USV2 and the communication between USV1 and USV2 is normal and the communication between USV2 and the GCS is also normal, this transition happens.

$$t_{10}, t_{12} = \begin{cases} True, & If\ there\ is\ no\ fault\ of\ USV1\ and\ it\ is\ not\ lost\ permanently \\ & and\ there\ is\ no\ give-way\ collision\ and\ no\ fault\ of\ USV2 \\ & and\ the\ communication\ between\ the\ USV1\ and\ USV2\ is\ normal \\ & and\ the\ communication\ between\ the\ USV2\ and\ the\ GCS\ is \\ & also\ normal, this\ transition\ happens \\ False, & Otherwise \end{cases}$$

- $(S_2 \rightarrow S_7), (S_3 \rightarrow S_7), (S_4 \rightarrow S_7), (S_5 \rightarrow S_7), (S_6 \rightarrow S_7), (S_9 \rightarrow S_7)$: If USV2 has detected severe fault, these transitions happen.

$$t_6, t_7, t_{23}, t_{18}, t_{19}, t_8 = \begin{cases} True, & If\ the\ USV\ has\ detected\ fault. \\ False, & Otherwise \end{cases}$$

- $(S_7 \rightarrow S_1)$: If the USV has been in a severe fault state, this transition happens.

$$t_{17} = \begin{cases} True, & If\ the\ USV\ has\ been\ in\ severe\ fault\ state \\ False, & Otherwise \end{cases}$$

- $(S_4 \rightarrow S_8)$: If the USV arrives at the destination, this transition happens.

$$t_5 = \begin{cases} True, & the\ USV\ has\ arrived\ the\ destination \\ False, & Otherwise \end{cases}$$

7.5.2.3 Kripke model of UAV

When the long endurance USV is following the path, the UAV will take off, fly to the designated destination and route the signal. The UAV is assumed to be able to land on the sea when the fault event is detected. When USV1 or USV2 arrives at the destination, the UAV will return to the launch point (return to launch). When the UAV loses signal, it should also return to launch. The Kripke model is shown in Fig. 7. 48.

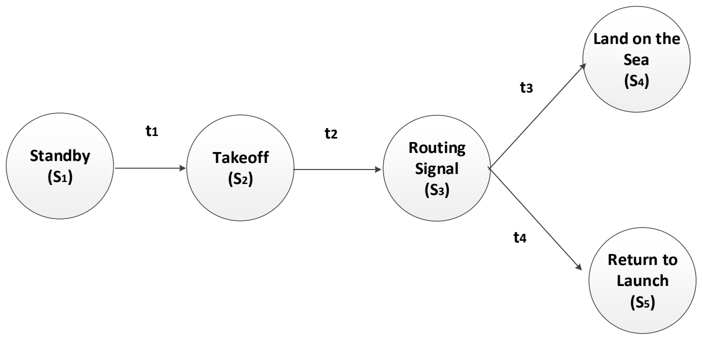


Fig. 7. 48. Kripke model for the UAV

The accessibility relationships are described as follows:

- $(S_1 \rightarrow S_2)$: If the path following state of USV1 is detected, this transition happens.

$$t_1 = \begin{cases} True, & If\ the\ USV1\ is\ path\ following\ and\ the\ direct\ or\ indirect \\ & communication\ between\ the\ USV1\ and\ UAV\ is\ normal \\ False, & Otherwise \end{cases}$$

- $(S_2 \rightarrow S_3)$: If the UAV takes off successfully and no fault event is detected, this transition happens.

$$t_2 = \begin{cases} True, & If\ the\ UAV\ takes\ off\ successfully\ and\ no\ fault \\ & event\ is\ detected \\ False, & Otherwise \end{cases}$$

- $(S_3 \rightarrow S_4)$: If the UAV is routing the signal and the fault event of the UAV is detected, this transition happens.

$$t_3 = \begin{cases} True, & If\ the\ UAV\ is\ routing\ the\ signal\ and\ the\ fault\ event \\ & of\ the\ UAV\ is\ detected \\ False, & Otherwise \end{cases}$$

- $(S_3 \rightarrow S_5)$: If the UAV is routing the signal and the arrival of USV1 or USV2 is detected or the communication between the UAV and the GCS is lost, this transition happens.

$$t_4 = \begin{cases} True, & If\ the\ UAV\ is\ routing\ the\ signal, the\ arrival\ of \\ & USV1\ or\ USV2\ is\ detected\ or\ the\ communication\ between\ the\ USV \\ & and\ the\ GCS\ is\ lost \\ False, & Otherwise \end{cases}$$

7.5.2.4 Kripke model of GCS

When USV1 is performing normally, the accessibility relationships between the states of the GCS model are the same as the ones we used in Section 7.3. However, when a permanent loss or fault/severe fault event of USV1 is confirmed, the GCS will switch to control the corresponding state of USV2 and the accessibility relationships are the same as those with USV1, as shown in Fig. 7. 49.

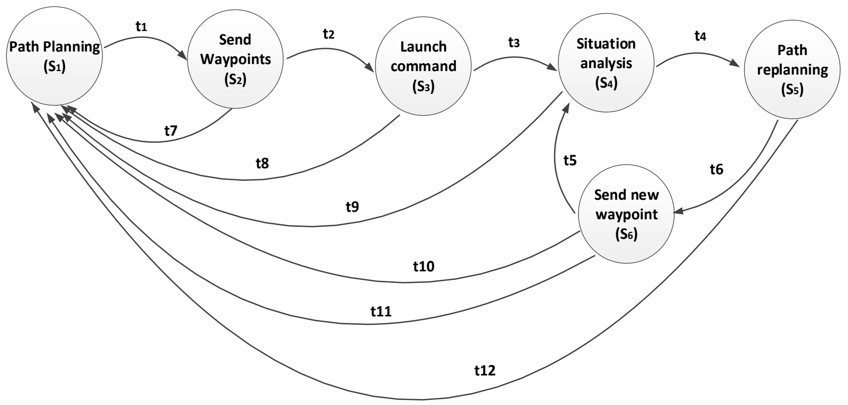


Fig. 7. 49. Kripke model for GCS

7.5.2.5 Kripke model of mesh network

The GCS, USV1, USV2 and UAV are configured as communication nodes, as shown in Fig. 7. 50 . A mesh network where all the nodes are connected is a full mesh network. A full mesh network yields the greatest amount of redundancy, so when one node fails, network traffic may be directed to any of the other nodes.

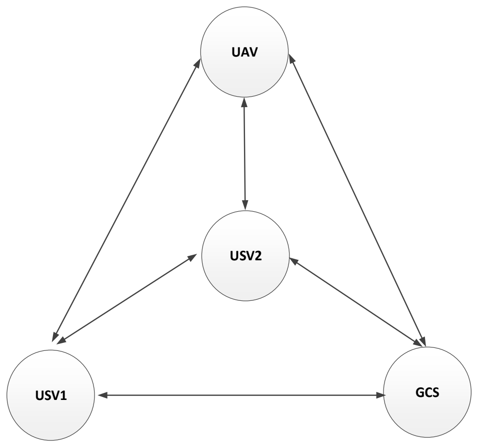


Fig. 7. 50. Communication channels between the agents

The mesh network between multiple autonomous agents is made up of the communication channels between arbitrary autonomous agents. The communication channel of two agents may be direct or indirect. For example, USV1 may send data to the GCS directly, via UAV, via USV2 or via both UAV and USV2. Instead of modelling all the possible communication paths, we modelled the direct and indirect communication channels between two vehicles by using one communication model, which has a normal communication state and a communication loss state. As it is possible to test the connection between two routers directly using the mesh network, this modelling method is reasonable and will also reduce the computing load. If one autonomous agent requires another agent’s information, the state of the communication channel between them will be queried. For example, the UAV will take off only when USV1 is detected to be path following, based on the information from USV1 transmitted to UAV through the mesh network. The direct and indirect communication channel and the corresponding communication detector between USV1 and the UAV may be modelled as shown in Fig. 7. 51 and Fig. 7. 52.

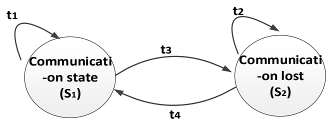


Fig. 7. 51. The Kripke model for the communication between USV1 and UAV

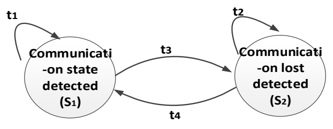

t1
t2
t3
t4
Communicati -on state detected (S1)
Communicati -on lost detected (S2)

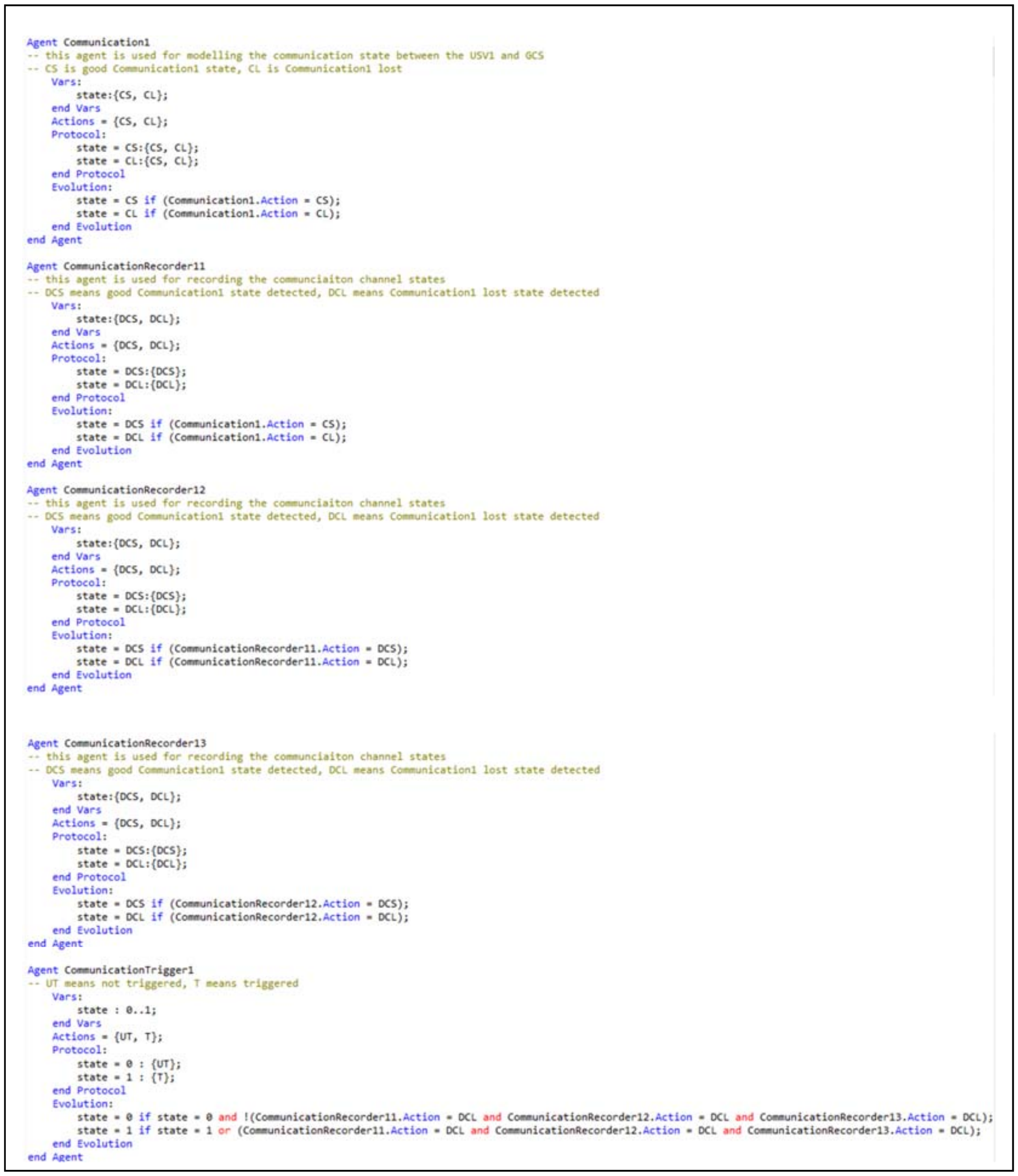
Fig. 7. 52. The Kripke model for the communication detector between USV1 and UAV

The accessibility relations of the communication model and the communication detector model are the same as those used in Section 7.3 and Section 7.4. Using the same method as in the previous sections, six communication channels and six communication detectors may be modelled.

7.5.3 Model Checking with MCMAS

7.5.3.1 MCMAS model

The Kripke models of the autonomous vehicles are translated into the MCMAS models using the same methods employed in Scenario 1 and Scenario 2. The method of modelling the permanent signal loss will be explained in this section. Three sequences of the communication states are modelled as three communication recorders, shown in Fig. 7. 53. *CommunicationRecorder11* records the latest state of the communication states between USV1 and GCS. *CommunicationRecorder12* records the state of *Commmunication11* and *CommunicationRecorder13* records the state of the *CommunicationRecorder12*. When the states of *CommunicationRecorder11*, *CommunicationRecorder12* and *CommunicationRecorder13* are all *DCL*, the state of *CommunicationTrigger1* will change to be 1, which means a permanent loss status is confirmed. As a result, the action *T*, will be sent to USV2. Once the permanent loss status is confirmed, USV2 will transit from the routing signal to the path following state.



```
Agent Communication1
-- this agent is used for modelling the communication state between the USV1 and GCS
-- CS is good Communication1 state, CL is Communication1 lost
    Vars:
        state:{CS, CL};
    end Vars
    Actions = {CS, CL};
    Protocol:
        state = CS:{CS, CL};
        state = CL:{CS, CL};
    end Protocol
    Evolution:
        state = CS if (Communication1.Action = CS);
        state = CL if (Communication1.Action = CL);
    end Evolution
end Agent

Agent CommunicationRecorder11
-- this agent is used for recording the communciaiton channel states
-- DCS means good Communication1 state detected, DCL means Communication1 lost state detected
    Vars:
        state:{DCS, DCL};
    end Vars
    Actions = {DCS, DCL};
    Protocol:
        state = DCS:{DCS};
        state = DCL:{DCL};
    end Protocol
    Evolution:
        state = DCS if (Communication1.Action = CS);
        state = DCL if (Communication1.Action = CL);
    end Evolution
end Agent

Agent CommunicationRecorder12
-- this agent is used for recording the communciaiton channel states
-- DCS means good Communication1 state detected, DCL means Communication1 lost state detected
    Vars:
        state:{DCS, DCL};
    end Vars
    Actions = {DCS, DCL};
    Protocol:
        state = DCS:{DCS};
        state = DCL:{DCL};
    end Protocol
    Evolution:
        state = DCS if (CommunicationRecorder11.Action = DCS);
        state = DCL if (CommunicationRecorder11.Action = DCL);
    end Evolution
end Agent


Agent CommunicationRecorder13
-- this agent is used for recording the communciaiton channel states
-- DCS means good Communication1 state detected, DCL means Communication1 lost state detected
    Vars:
        state:{DCS, DCL};
    end Vars
    Actions = {DCS, DCL};
    Protocol:
        state = DCS:{DCS};
        state = DCL:{DCL};
    end Protocol
    Evolution:
        state = DCS if (CommunicationRecorder12.Action = DCS);
        state = DCL if (CommunicationRecorder12.Action = DCL);
    end Evolution
end Agent

Agent CommunicationTrigger1
-- UT means not triggered, T means triggered
    Vars:
        state : 0..1;
    end Vars
    Actions = {UT, T};
    Protocol:
        state = 0 : {UT};
        state = 1 : {T};
    end Protocol
    Evolution:
        state = 0 if state = 0 and !(CommunicationRecorder11.Action = DCL and CommunicationRecorder12.Action = DCL and CommunicationRecorder13.Action = DCL);
        state = 1 if state = 1 or (CommunicationRecorder11.Action = DCL and CommunicationRecorder12.Action = DCL and CommunicationRecorder13.Action = DCL);
    end Evolution
end Agent
```

Fig. 7. 53. ISPL code of communication recorders and trigger

7.5.3.2 Modelling of properties to be verified

Aside from the 14 properties verified in Section 7.4, there are six new properties to be verified in this section. These properties and their corresponding CTL formulas are described as follow:

15, 'When USV1 is following the path, USV2 is routing the signal, and there is no collision risk for USV2, the communications between USV2 and the GCS and USV1 are normal and there is no fault in USV2, and USV1 detects a fault event, USV2 should transit from routing signal state to path following state or USV1 arrives at the destination and USV2 returns to launch.'

$$\begin{aligned} AG((&USV1.state = PF \wedge USV2.state = RS \wedge !AIS2.state \\ &= DGC \wedge Communicationdetector2.state \\ &= DCS \wedge USV1USV2Communicationdetector.state \\ &= DCS \wedge FaultDetector2.state = DNF \wedge FaultTrigger1.state \\ &= 1 \rightarrow AX(USV2.state = PF \vee (USV1.state = AR \wedge USV2.state \\ &= RTL))) \end{aligned}$$

16, 'If there is no fault in USV1 and USV1 has not been confirmed to be lost permanently, USV2 will never take the place of USV1 to execute the path following behaviour'.

$$\begin{aligned} AG(&FaultTrigger1.state = 0\ and\ CommunicaitonTrigger.state = 0 \\ &\rightarrow AX(!USV2.state = PF)) \end{aligned}$$

17, 'If USV1 or USV2 arrives at the destination, there is no fault in UAV and the communication channels among the UAV, USV1 and USV2 are normal, the UAV will transit from routing signal state to return to launch state'.

$$\begin{aligned} AG(((&USV1.state = AR \vee USV2.state = AR) \wedge UAV.state \\ &= RS \wedge UAVFaultdetector.state \\ &= DNF \wedge UAVUSV1Communicationdetector.state \\ &= DCS \wedge UAVUSV2Communicationdetector.state = DCS) \\ &\rightarrow AX(UAV.state = RTL)) \end{aligned}$$

18, 'If USV1 has no fault and is not lost permanently, when USV2 just avoided a collision risk, and communication with GCS is normal and no give-way collision risk appears and there is no fault event of USV2, USV2 will transit to the routing signal state'.

$$\begin{aligned} AG(\ USV2.state = CA\ \wedge Communicationdetector2.state \\ = DCS \wedge FaultDetector2.state = DNF\ \wedge !\,AIS2.state \\ = DGC\ \wedge FaultTrigger1.state = 0\ and\ CommunicationTrigger \\ = 0 \rightarrow AX(USV2.state = RS)) \end{aligned}$$

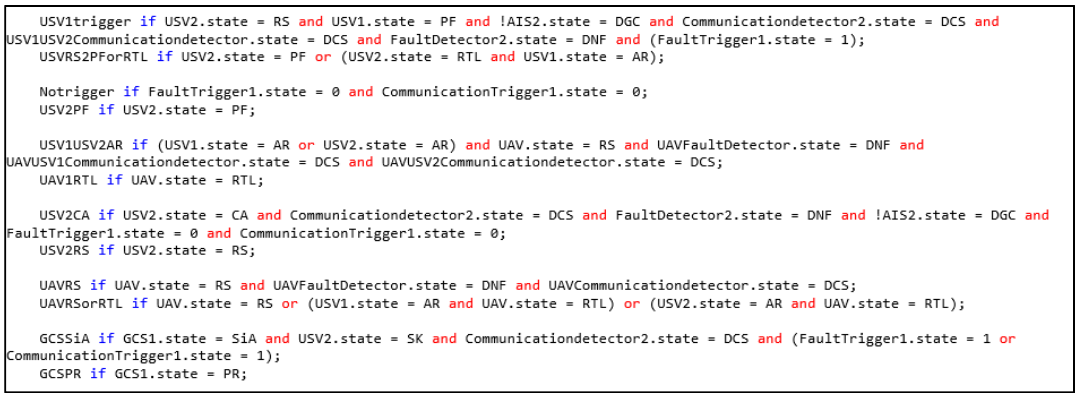
19, *'If the UAV is routing the signal, there is no fault of UAV, the communication between the UAV and the GCS is normal, then UAV will continue to route the signal, or USV1 arrives at the destination and UAV returns to launch, or USV2 arrives the destination and UAV returns to launch'.*

$AG(UAV.state = RS\ \wedge UAVFaultDetector.state = DNF\ \wedge$
$UAVCommunicationdetector.state = DCS \rightarrow AX(UAV.state = RS \vee$
$(USV1.state = AR \wedge UAV.state = RTL) \vee (USV2.state = AR \wedge UAV.state =$
$RTL))$

20, *'The GCS is in the situation analysis state and USV1 has abandoned the mission and USV2 is in station keeping state then the communication between USV2 and the GCS is recovered, then the GCS will be in path re-planning state to generate a new path for USV2 to continue the mission'.*

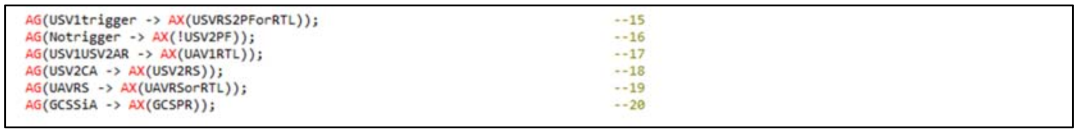
$$\begin{aligned} AG(GCS.state = SiA \wedge USV2.state = SK \wedge Communicationdetector2.state \\ = DCS\ \wedge (FaultTrigger1.state \\ = 1\ \vee CommunicationTrigger.state = 1) \rightarrow AX(GCS1.state \\ = PR)) \end{aligned}$$

The properties are expressed in the *Evaluation* and *Formulae* code sections, shown in Fig. 7. 54 and Fig. 7. 55.



```
    USV1trigger if USV2.state = RS and USV1.state = PF and !AIS2.state = DGC and Communicationdetector2.state = DCS and
USV1USV2Communicationdetector.state = DCS and FaultDetector2.state = DNF and (FaultTrigger1.state = 1);
    USVRS2PForRTL if USV2.state = PF or (USV2.state = RTL and USV1.state = AR);

    Notrigger if FaultTrigger1.state = 0 and CommunicationTrigger1.state = 0;
    USV2PF if USV2.state = PF;

    USV1USV2AR if (USV1.state = AR or USV2.state = AR) and UAV.state = RS and UAVFaultDetector.state = DNF and
UAVUSV1Communicationdetector.state = DCS and UAVUSV2Communicationdetector.state = DCS;
    UAV1RTL if UAV.state = RTL;

    USV2CA if USV2.state = CA and Communicationdetector2.state = DCS and FaultDetector2.state = DNF and !AIS2.state = DGC and
FaultTrigger1.state = 0 and CommunicationTrigger1.state = 0;
    USV2RS if USV2.state = RS;

    UAVRS if UAV.state = RS and UAVFaultDetector.state = DNF and UAVCommunicationdetector.state = DCS;
    UAVRSorRTL if UAV.state = RS or (USV1.state = AR and UAV.state = RTL) or (USV2.state = AR and UAV.state = RTL);

    GCSSiA if GCS1.state = SiA and USV2.state = SK and Communicationdetector2.state = DCS and (FaultTrigger1.state = 1 or
CommunicationTrigger1.state = 1);
    GCSPR if GCS1.state = PR;
```

Fig. 7. 54. ISPL code for Evaluation

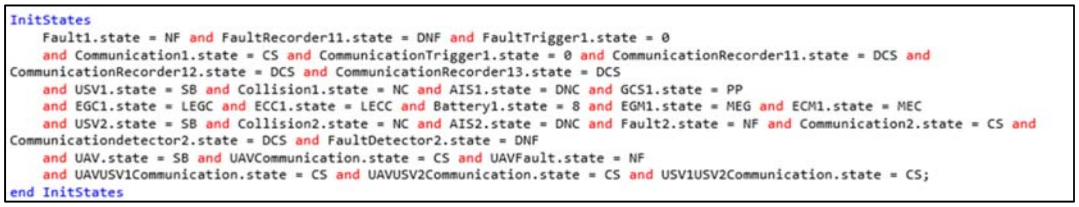


```
  AG(USV1trigger -> AX(USVRS2PForRTL));                          --15
  AG(Notrigger -> AX(!USV2PF));                                  --16
  AG(USV1USV2AR -> AX(UAV1RTL));                                 --17
  AG(USV2CA -> AX(USV2RS));                                      --18
  AG(UAVRS -> AX(UAVRSorRTL));                                   --19
  AG(GCSSiA -> AX(GCSPR));                                       --20
```

Fig. 7. 55. ISPL code for Formulae

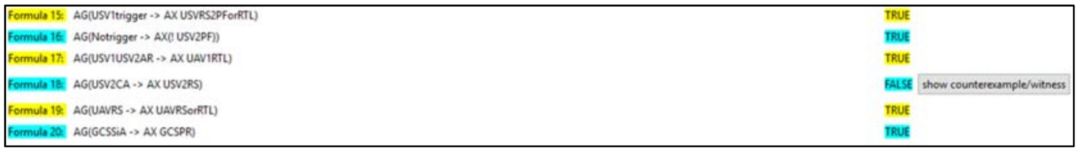
7.5.3.3 Verification Results

The configuration of the *InitStates* code is shown in Fig. 7. 56. Note that the results of Formula 1 - Formula 14 have the same results as those shown in Section 7.4. The verification results of the new Formulas are shown in Fig. 7. 57. In the verification results, we see that Formula 18 shows the '*FALSE*' result. The error trace may be acquired using the 'show counterexample/witness' option. The analysis of these three error traces are discussed in Section 7.5.3.4.



```
InitStates
    Fault1.state = NF and FaultRecorder11.state = DNF and FaultTrigger1.state = 0
    and Communication1.state = CS and CommunicationTrigger1.state = 0 and CommunicationRecorder11.state = DCS and
CommunicationRecorder12.state = DCS and CommunicationRecorder13.state = DCS
    and USV1.state = SB and Collision1.state = NC and AIS1.state = DNC and GCS1.state = PP
    and EGC1.state = LEGC and ECC1.state = LECC and Battery1.state = 8 and EGM1.state = MEG and ECM1.state = MEC
    and USV2.state = SB and Collision2.state = NC and AIS2.state = DNC and Fault2.state = NF and Communication2.state = CS and
Communicationdetector2.state = DCS and FaultDetector2.state = DNF
    and UAV.state = SB and UAVCommunication.state = CS and UAVFault.state = NF
    and UAVUSV1Communication.state = CS and UAVUSV2Communication.state = CS and USV1USV2Communication.state = CS;
end InitStates
```

Fig. 7. 56. ISPL code for InitStates



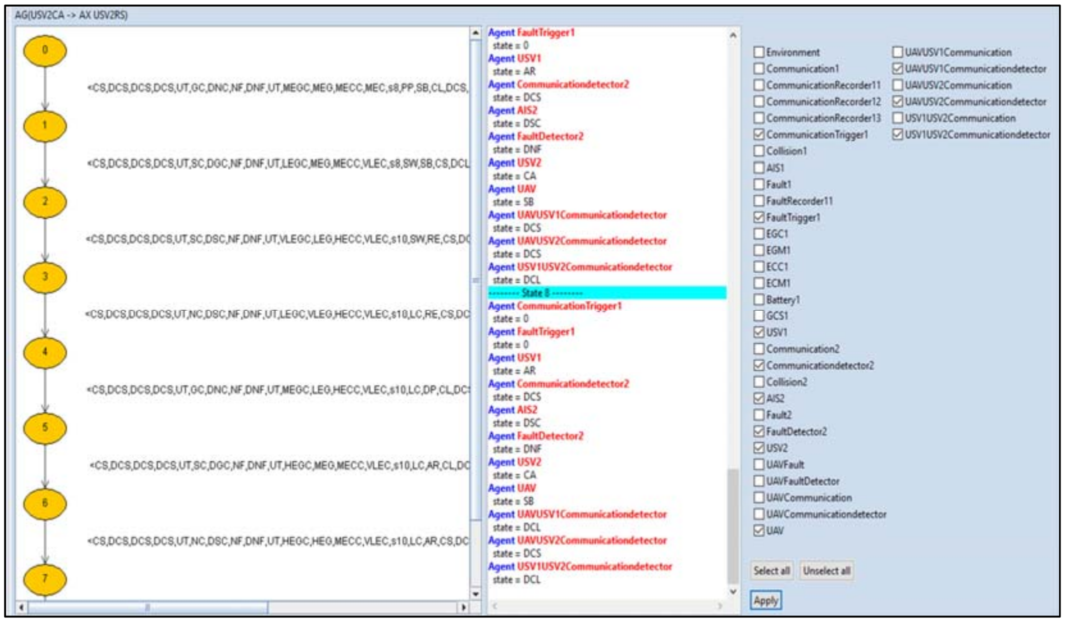
```
Formula 15: AG(USV1trigger -> AX USVRS2PForRTL)          TRUE
Formula 16: AG(Notrigger -> AX(! USV2PF))                TRUE
Formula 17: AG(USV1USV2AR -> AX UAV1RTL)                 TRUE
Formula 18: AG(USV2CA -> AX USV2RS)                      FALSE  show counterexample/witness
Formula 19: AG(UAVRS -> AX UAVRSorRTL)                   TRUE
Formula 20: AG(GCSSiA -> AX GCSPR)                       TRUE
```

Fig. 7. 57. Verification results

7.5.3.4 Analysis and discussion

Formulas 15, 16, 17, 19 and 20 acquire '*TRUE*' results. The result of Formula 15 shows that when USV1 has a fault event, USV2 will take the place of USV1 to continue the mission, proving that the fault tolerance property of the whole system may improve the completion of the mission. The result of Formula 16 shows that if USV1 performs well, USV2 will not take the place of USV1, which makes full use of the energy generation capability of the long endurance USV1, and keeps reliable communication using USV2 as the communication node of the wireless mesh network. The result of Formula 17 shows that once USV1 or USV2 completes the mission, the UAV may return to launch successfully. The result of Formula 19 shows if there is no fault or communication loss of UAV, it will route the signal or return to launch in the case that USV1 or USV2 arrive the destination. The result of Formula 20 shows that after USV1 has malfunctioned or is lost permanently, when USV2 is in station keeping state and the communication between USV2 and GCS is recovered, the GCS will transit from the situation analysis state to the path re-planning state to help USV2 plan a new path. This property means the GCS may take over USV2 when USV1 malfunctions or is lost permanently.

Formula 18 generates a '*FALSE*' result and the counterexample of Formulae 18 is shown in Fig. 7. 58. The property of Formula 18 is '*If USV1 has no fault and is not lost permanently, when USV2 just avoided a collision risk then no give-way collision risk appears, USV2 will transit to the routing signal state'.*


```
AG(USV2CA -> AX USV2RS)

0
<CS,DCS,DCS,DCS,UT,GC,DNC,NF,DNF,UT,MEGC,MEG,MECC,MEC,s8,PP,SB,CL,DCS,
1
<CS,DCS,DCS,DCS,UT,SC,DGC,NF,DNF,UT,LEGC,MEG,MECC,VLEC,s8,SW,SB,CS,DCL
2
<CS,DCS,DCS,DCS,UT,SC,DSC,NF,DNF,UT,VLEGC,LEG,HECC,VLEC,s10,SW,RE,CS,DC
3
<CS,DCS,DCS,DCS,UT,NC,DSC,NF,DNF,UT,LEGC,VLEG,HECC,VLEC,s10,LC,RE,CS,DC
4
<CS,DCS,DCS,DCS,UT,GC,DNC,NF,DNF,UT,MEGC,LEG,HECC,VLEC,s10,LC,DP,CL,DC
5
<CS,DCS,DCS,DCS,UT,SC,DGC,NF,DNF,UT,HEGC,MEG,MECC,VLEC,s10,LC,AR,CL,DC
6
<CS,DCS,DCS,DCS,UT,NC,DSC,NF,DNF,UT,HEGC,HEG,MECC,VLEC,s10,LC,AR,CS,DC
7
```

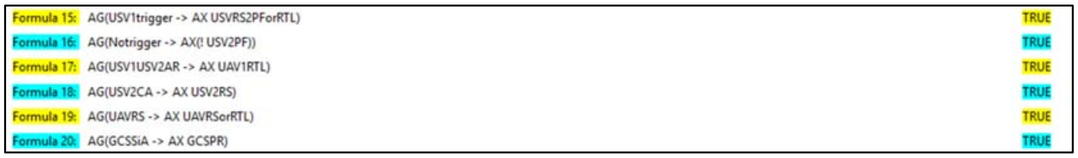
```
Agent FaultTrigger1
 state = 0
Agent USV1
 state = AR
Agent Communicationdetector2
 state = DCS
Agent AIS2
 state = DSC
Agent FaultDetector2
 state = DNF
Agent USV2
 state = CA
Agent UAV
 state = SB
Agent UAVUSV1Communicationdetector
 state = DCS
Agent UAVUSV2Communicationdetector
 state = DCS
Agent USV1USV2Communicationdetector
 state = DCL
-------- State 8 --------
Agent CommunicationTrigger1
 state = 0
Agent FaultTrigger1
 state = 0
Agent USV1
 state = AR
Agent Communicationdetector2
 state = DCS
Agent AIS2
 state = DSC
Agent FaultDetector2
 state = DNF
Agent USV2
 state = CA
Agent UAV
 state = SB
Agent UAVUSV1Communicationdetector
 state = DCL
Agent UAVUSV2Communicationdetector
 state = DCS
Agent USV1USV2Communicationdetector
 state = DCL
```

☐ Environment	☐ UAVUSV1Communication
☐ Communication1	☑ UAVUSV1Communicationdetector
☐ CommunicationRecorder11	☐ UAVUSV2Communication
☐ CommunicationRecorder12	☑ UAVUSV2Communicationdetector
☐ CommunicationRecorder13	☐ USV1USV2Communication
☑ CommunicationTrigger1	☑ USV1USV2Communicationdetector
☐ Collision1	
☐ AIS1	
☐ Fault1	
☐ FaultRecorder11	
☑ FaultTrigger1	
☐ EGC1	
☐ EGM1	
☐ ECC1	
☐ ECM1	
☐ Battery1	
☐ GCS1	
☑ USV1	
☐ Communication2	
☑ Communicationdetector2	
☐ Collision2	
☑ AIS2	
☐ Fault2	
☑ FaultDetector2	
☑ USV2	
☐ UAVFault	
☐ UAVFaultDetector	
☐ UAVCommunication	
☐ UAVCommunicationdetector	
☑ UAV	

Fig. 7. 58. The counterexample of Formulae 18

However, from the counterexample, we found that when USV2 just avoided a collision risk, it stays at the collision avoidance state instead of transiting to the routing signal state because the communication between USV1 and USV2 is lost. In other words, when the communication between USV1 and USV2 is lost, it cannot be confirmed whether USV1 has experienced a fault event or is lost permanently, so USV2 cannot make the decision to transit to the routing signal state or taking the place of USV1 to follow the path. In terms of maintaining the fault tolerance property and reliable communication, USV2 should transit to the routing signal state from the collision avoidance state when there is no communication between USV1 and USV2. Because the routing signal state helps to improve the communication signal between USV1 and USV2 when the communication is lost. Moreover, when the communication becomes normal and USV1 is confirmed to abandon the mission, USV1 can still transit to the path following state from the routing signal state.

Therefore, the transition condition from the collision avoidance state to the routing signal state will be, '*When USV1 has no fault event or is not lost permanently,* *USV2 hasn't detected the give-way collision and the fault event and the communication between USV2 and GCS is normal, USV2 will transit to the routing signal state*'. According to the new transition condition, USV2 will transit to the routing signal state even when USV2 cannot receive the information determining whether if USV1 has abandoned the mission due to its fault or a permanent lost problem. By modifying the ISPL code of USV2 according to the corrected Kripke model, we get the '*TRUE*' result of Formulae 18, as shown in Fig. 7. 59.



```
Formula 15:  AG(USV1trigger -> AX USVRS2PForRTL)      TRUE
Formula 16:  AG(Notrigger -> AX(! USV2PF))            TRUE
Formula 17:  AG(USV1USV2AR -> AX UAV1RTL)             TRUE
Formula 18:  AG(USV2CA -> AX USV2RS)                  TRUE
Formula 19:  AG(UAVRS -> AX UAVRSorRTL)               TRUE
Formula 20:  AG(GCSSiA -> AX GCSPR)                   TRUE
```

Fig. 7. 59. The verification result of the corrected Kripke model

7.6 Summary

This research tackled the problems of model checking using the MCMAS for verifying the decision-making behaviours of a long endurance USV. The Kripke model and computational tree logic were applied to construct the model and specifications of the system of interest.

In the first scenario, the USV decision-making system needs to make the proper decision under several concurrent environmental problems including communication lost, collision risk and fault event. The decision-making behaviours and environmental uncertainties were specified and decision-making system were verified using MCMAS.

The second scenario extended the first scenario by incorporating energy factors. The USV considered in the second scenario has the capability to utilize multiple energy resources, including solar energy, wind energy and diesel energy. The energy generation module includes the solar panel, wind turbine and diesel generator. The states of the energy generation module are affected by the environmental energy generation condition, battery level and USV behaviour. The decision-making system took into account the energy generation and energy consumption for improving operational endurance. Both the safety properties and the long endurance properties of the decision-making behaviours under concurrent and non-deterministic uncertainties were finally verified using MCMAS.

For the third scenario, the second scenario was extended to become a multi-agent system, modelling the long endurance USV, the regular USV, the UAV and the GCS. This system has fault tolerance capability and includes a mesh network to increase the reliability of the communication network. The long endurance USV was commanded to collect data in a large coverage area. The UAV and the regular USV were commanded to route the signal between the long endurance USV and the GCS. All the autonomous vehicles and the GCS are configured as communication nodes. The multiple vehicles system and multiple environmental uncertainties were modelled and verified. The modelled environmental uncertainties include the communication channels of the mesh network, the fault event, the collision risk, the energy generation condition (solar irradiance and wind conditions) and the energy consumption condition (sea current state). Finally, the safety and long endurance properties of the long endurance USV and the fault tolerance property of the whole system were verified.

This research demonstrates how to use the model checking algorithm to model the multiple concurrent environmental uncertainties and multiple autonomous vehicles allowing verification of the decision-making system. Our research also shows how to use the counterexample to retrospect and improve the decision of the USV decision-making system. In the final scenario, thirty-five Kripke models were developed, including fifteen communication-related models, four collision risk-related models, seven fault event-related models, five energy-related models, and four autonomous systems-related models. The number of reachable states extended up to 3.68 billion with an execution time of 87.5 seconds. The model checker SMV consumed 79 hours and 33 minutes to verify the multi-agent system consisting of four UAVs by exploring 1.25809 million states (Choi, et al., 2015). In contrast, the MCMAS was able to explore 29,251 times more reachable states while only consuming 0.03% of computational time, demonstrating the strong computational efficiency of the model checker MCMAS. The final scenario used the USV models and uncertainties models developed in the first and second scenarios with minor modifications, showing the scalability of the MCMAS.

8 Conclusions and Future Work

8.1 Conclusions

In Chapter Two, four path following algorithms, namely, the Carrot Chasing algorithm, the Nonlinear Guidance Law algorithm, the Pure pursuit and LOS algorithm, and the Vector Field algorithm, were implemented into the dynamic model of C-Enduro USV. The wind and sea current disturbances were also modelled in this research. The cross-track error and the control effort of four path following algorithms were compared in the presence of the wind and sea current disturbances. The results could be used for choosing which is the most suitable path following algorithm for executing different missions efficiently.

In Chapter Three, a Voronoi-Visibility roadmap-based shortest path planning algorithm was proposed. The Voronoi diagram is well known for its computational efficiency and its capability to keep the vehicle maximum clearance distance from the obstacles, however, the generated path is far from optimal and needs to be refined. The Visibility graph may generate an optimal path but its computational efficiency is low. The proposed algorithm integrates the advantages of the Voronoi diagram and the Visibility graph. The proposed algorithm includes the collision-free Voronoi roadmap generation, the Voronoi shortest path generation, the Visibility graph generation and Voronoi-Visibility shortest path generation. The proposed algorithm was compared with the VM (Voronoi path planning algorithm refined by minimizing the number of waypoints) algorithm in ten missions in the Singapore Strait. The results showed that the proposed VV path planning algorithm saved the overall mission length by up to 13.89% compared with the VM path planning algorithm. By comparing the computational time of the Voronoi, VM and the VV path planning algorithms, it may be concluded that although the VV method required more time than the Voronoi and VM algorithms, it still kept the same level of the computational efficiency as that of the Voronoi path planning algorithm. The simulation results off the coast of Croatia showed the flexibility of the proposed algorithm in dealing with different geographical scenarios. Moreover, the proposed VV algorithm may ensure the safety of the USV by keeping the USV a configurable clearance distance from the coastlines.

In Chapter Four, a Voronoi-Visibility roadmap-based energy efficient path planning algorithm was proposed by taking the sea current data into account. In the proposed approach, the Voronoi diagram, Visibility graph, Dijkstra's search and energy consumption function were integrated to generate the energy efficient path, which allows the USVs to avoid the obstacles while saving energy. The Voronoi-Visibility energy efficient path planning algorithm and the Voronoi-Visibility shortest path planning algorithm were simulated in the same geographical locations (Singapore Strait and Croatia islands) but with different mission parameters. By analysing the impacts of the mission time, the travelling speed and the sea current state on the results, it may be concluded that in the favourable sea current situation, low travelling speed situation, and fast changing sea current situation, it is easier for the proposed VV energy efficient path planning algorithm to save a higher percentage of the total energy than the shortest path. In Mission No.3, the results showed that the energy efficient path saved up to 52.84% more energy than the shortest path. By comparing the computational times between the proposed algorithm and the Voronoi energy efficient path planning algorithm, we may find the proposed VV algorithm still keeps the same level of computational efficiency as that of the Voronoi energy efficient path planning algorithm.

In Chapter Five, the proposed Voronoi-Visibility energy efficient path planning algorithm was extended by taking the time varying sea currents into consideration. The new energy efficient path planning algorithm integrated the Voronoi diagram, Visibility graph, Dijkstra's search algorithm and Genetic Algorithm (GA). The coastline expanding algorithm and Voronoi algorithm were first applied to generate a Voronoi collision-free roadmap. Then the Dijkstra's search algorithm and Visibility algorithm were implemented to generate the first generation of the GA algorithm. The first generation of the paths were used for generating new paths by applying the proposed selection operator, crossover operator and mutation operator. A fitness function was proposed to evaluate the fitness value of the new generated path under the time varying sea current state. The paths having the lowest fitness values are kept in each generation to generate new paths by using the proposed crossover operator and mutation operator. When the termination condition was satisfied, the iteration stops and the best individual is selected. The new energy efficient path planning algorithm was finally compared with the previously proposed Voronoi-Visibility energy efficient path planning algorithm in three Singapore missions. The results showed that the GA based energy efficient path planning may save up to 27.68% of the energy during the entire mission duration, which demonstrated that the proposed algorithm generated the energy optimal path for missions with time varying sea currents.

In Chapter Six, the geometric collision avoidance algorithm based on COLREGS was proposed for supporting USV long endurance operation. The collision avoidance algorithm includes the collision detection algorithm, the decision-making algorithm, the collision resolution algorithm and the resolution guidance algorithm. The collision detection algorithm was used to detect any potential collisions. The decision-making algorithm took the COLREGS into account, to decide whether the USV should alter its course angle or maintain its course. If the USV was required to alter its course angle, the collision resolution and resolution guidance algorithm navigated the USV through the new course, avoiding any intruders. The proposed algorithm was simulated and validated under single intruder scenario, multiple dynamic intruders scenario and multiple static-dynamic mixed intruders scenario.

Lastly, in Chapter Seven, this thesis tackled the problems of using model checking method for verifying the decision-making behaviours of multi-agent system under multiple concurrent uncertainties. In Section 7.3, the uncertainties, the decision-making systems of USV, and the decision-making system of GCS, were modelled using the Kripke modelling method. The external uncertainties, including the communication state, the traffic information, and the malfunction event, were modelled as non-deterministic and concurrent factors. This modelling method made it possible to verify the property of the autonomous system under multiple concurrent uncertainties. The Kripke models were translated into ISPL code, which is the language of the model checker MCMAS. The decision-making systems of the USV and GCS were verified successfully. The counterexample of

Formula 2 showed the designed decision-making system of the USV did not consider the situation where the give-way collision risk and the communication loss states happen simultaneously. By adding the state transition condition, ‘when the USV detects the give-way collision risk, the USV will transit from path following state to station keeping state, even when the communication signal is lost’, the corrected model was verified again and the Formula 2 acquired the ‘*TRUE*’ result. In Section 7.4, a long endurance USV, equipped with solar panel, wind turbine, and diesel generator was modelled and verified. Additional uncertainties, namely, the energy consumption condition and the energy generation condition, were added to the model of Section 7.3. The decision-making system of the USV was designed to take into account the energy consumption, energy generation and battery level to make the decision of transiting its states. When the battery level is low, the USV should maintain station keeping state to save energy and trigger the diesel generator to charge the battery. When the battery level is high and the energy generation is greater than the energy consumption, the USV will make the decision to increase its travelling speed. The safety property and the long endurance property were verified by using MCMAS. In Section 7.5, to improve the mission completion, multiple autonomous systems, including the long endurance USV, regular USV, a UAV, and GCS, were simulated in one of the test case scenarios. A wireless mesh network was built using each vehicle as a signal routing node to increase the signal coverage and improve the signal reliability. When the long endurance USV was executing the mission normally, the regular USV and UAV route the signal. However, when the long endurance USV has a malfunction or was detected to be lost permanently, the regular USV will take the place of the long endurance USV to continue to execute the mission (fault tolerance property). The models of Section 7.3 and Section 7.4 were modified and added into the multi-agent system of Section 7.5, and the wireless mesh network was also modelled. The multi-agent system and the multiple concurrent uncertainties were finally verified, and the counterexample of Formula 18 showed the model checker found the system design mistake. When the regular USV completes the collision avoidance manoeuvre, the communication signal between the regular USV and the long endurance USV is lost, as the regular USV does not acquire the information of the long endurance USV if it is lost permanently or has malfunctioned. As a result, the regular USV cannot decide to continue to route the signal or replace the long endurance USV. From a safety point of view, when the communication is lost, the regular USV should transit to the routing signal state because this transition will help increase the stability of the signal. Moreover, when the long endurance USV is lost or has malfunctioned, the regular USV may still transit to the path following state from the routing signal state. The system model was modified based on the analysis and Formula 18 finally acquired the '*TRUE*' result. Chapter 7 presented the method for modelling the multi-agent system and the multiple concurrent uncertainties so that the model checker may be used for verifying desirable properties. Chapter 7 also showed how to use the counterexample to retrospect and improve the design of the decision-making system.

8.2 Future Work

In the energy efficient path planning algorithms of this research, the USV was assumed to keep a constant speed. In future work, the path planning algorithms developed in this research may be modified to calculate the time-saving path under the assumption that the USV is travelling at full throttle instead of keeping a constant speed. This may be realized by changing the energy consumption weight to the time consumption weight. The traffic data will also be useful to be integrated into the path planning algorithm since the crowded traffic will delay the journey of the USV. For time-saving purpose, the crowded area should be avoided. In terms of the safety, the area with high speed sea current or gale also needs to be avoided and this can be realized by defining a safety weight of each candidate path. The dynamic water depth data may also be necessary to be considered in the path planning algorithm to improve the safety of the USV, for instance, the shallow water may be dangerous for the large and heavy ships those have heavy drafts. The path planning algorithm may take the speed changing situation into consideration, because when the energy generation is sufficient, it will be necessary to accelerate so as to maximize the use of the renewable energy. However, taking into account the changing speed mission, it will increase the dimension of the problem, a computationally efficient searching algorithm needs to be developed. Once the operator/GCS plans the path, the USV will execute the path following command, however, the mission may be delayed by crowded traffic or malfunction. In order to keep the USV follow the up-to-date energy efficient path, the criteria about when the GCS should plan a new path needs to be defined. In the case of a high-speed vessel, the dynamics of the vessels should be taken into consideration, requiring application of the path smoothing algorithm. Moreover, the other environmental data including the wind or solar irradiance data may be necessary to take into account when the USV is equipped with the wind turbine and solar panels. For the long-range mission of a vast area, it might be impossible to calculate the whole path because the weather data and the traffic data cannot be acquired in advance. In this case, path re-planning algorithm will be necessary to develop. One solution is to divide the path using equidistant points and develop the energy efficient paths between the equidistant segments, which will also save the computational efficiency when dealing with a shorter path. To improve the efficiency in some particular missions, the path planning algorithms considering the formation of multiple USVs may be also useful to develop.

In terms of the collision avoidance algorithm, the scenario may be more complex than the simulated scenarios. The other intruders may not follow the rules and emergent action may be integrated into the decision-making algorithm to keep the safety of the USV. When avoiding the intruders, the decision-making algorithm may break the COLREGS while there are other intruders nearby. In some scenarios, one big manoeuvre is better than a sequence of small manoeuvres. Therefore, a more elaborate decision-making algorithm requires to be made. The collision avoidance algorithm may also take into account the dynamic capacity of the USV to make the proper decision. The sensor fusion algorithm, fusing the information from the radar and camera, is also necessary to develop.

The results of verifying the decision-making system in the final scenario showed that the number of reachable states extended to 3.68 billion with an execution time of 87.5 seconds, which implies that the model checker MCMAS may be used for verifying the scenario with more agents and more concurrent environmental factors in a practical time. One possible scenario is that the USV may be used for collecting data from many marine beacons and buoys in the sea, which may be defined as the end users of the wireless mesh network. Unlike the routers, the end users can sleep and wake only when approached by the router, making it possible to save energy and collect the data for a long time. The partial wireless mesh network may also be used in some complex scenarios. Another possible scenario is the cooperation between the buoys, USVs, UUVs, UAVs and GCS. Since the buoys and UUVs will be allocated with different kind of tasks and they will also encounter various uncertainties, it will be challenging to model and verify such a complex system with so many different agents, behaviours and uncertainties. Moreover, the models of malfunctions need to be added. The malfunction parts may include propeller, engine, battery, wind turbine, diesel engine, radar, camera, software, etc.

The proposed path planning, path following, collision avoidance and decision-making algorithms need to be implemented into the real vehicles to validate. Some other control algorithms, such as Model Predictive Control (MPC) and backstepping control, can also be developed to compare the performance with PID control. For the missions with a team of USVs, a software platform, supporting the communication and control of multiple agents, need to be implemented. Two open-source operating systems, Robot Operating System (ROS) and Mission Oriented Operating Suite (MOOS), support the operation of multiple agents. ROS is designed to support the control of general robots. MOOS is developed to support the control of autonomous marine vehicles such as the cooperative mission between USVs and UUVs. In the cooperative mission, as the quality of communication is critical for the safety of the vehicle, wireless mesh network is worth to be implemented to improve the stability of the communication and keep the fault tolerance capability.

References

Agarwal, A. & Lermusiaux, P. F. J., 2011. Statistical field estimation for complex coastal regions and archipelagos. *Ocean Modelling,* Volume 40(2), pp. 164-189.

Almeida, C. et al., 2009. *Radar based collision detection developments on USV ROAZ II.* Oceans 2009-Europe (pp. 1-6). IEEE..

Alvarez, A., Caiti, A. & Onken, R., 2004. Evolutionary path planning for autonomous underwater vehicles in a variable ocean. *IEEE Journal of Oceanic Engineering,* Volume 29(2), pp. 418-429.

Amato, N. M. & Wu, Y., 1996. A randomized roadmap method for path and manipulation planning. *Robotics and Automation,* Volume 1, pp. 113-120.

Ashrafiuon, H., Muske, K., McNinch, L. & Soltan, R., 2008. Sliding-Mode Tracking Control of Surface Vessels. *IEEE Transactions on Industrial Electronics,* Volume 55(11), pp. 4004-4012.

ASV Global Ltd, 2012. *ASV awarded Thales contract for re-configurable USV.* [Online]
Available at: https://www.asvglobal.com/asv-awarded-thales-contract-for-re-configurable-usv/
[Accessed 1 July 2014].

ASV Global Ltd, 2013. *ASV to Build USV under SBRI Funding.* [Online]
Available at: https://www.asvglobal.com/asv-to-build-usv-under-sbri-funding/
[Accessed 08 June 2016].

ASV Global Ltd, 2013. *Product Information.* [Online]
Available at: https://www.asvglobal.com/product/c-enduro/
[Accessed 5 May 2016].

ASV Global Ltd, 2015. *C-Worker.* [Online]
Available at: https://www.asvglobal.com/product/c-worker-4/
[Accessed 5 May 2017].

ASV Global Ltd, 2016. *ASV unmanned marine systems.* [Online] Available at: https://www.asvglobal.com/ [Accessed 07 June 2016].

Béatrice, B. et al., 2001. *Model Checking.* Berlin Heidelberg: In Systems and Software Verification, pp. 39-46. Springer.

BEN LINE AGENCIES, 2014. *10 Important Things To Do During Ship Collision Accident.* [Online] Available at: http://www.benlineagencies.com/article_view.php?id=1 [Accessed 07 June 2016].

Benavides, F., Tejera, G., Pedemonte, M. & Casella, S., 2011. *Real path planning based on genetic algorithm and Voronoi diagrams.* In Robotics Symposium, 2011 IEEE IX Latin American and IEEE Colombian Conference on Automatic Control and Industry Applications (LARC), pp. 1-6. IEEE..

Bhattacharya, P. & Gavrilova, M. I., 2008. Roadmap-based path planning-Using the Voronoi diagram for a clearance-based shortest path. *IEEE Robotics & Automation Magazine,* Volume 15, no. 2.

Bibuli, M., Caccia, M. & Lapierre, L., 2007. Path-following algorithms and experiments for an autonomous surface vehicle. *IFAC Proceedings Volumes,* Volume 40(17), pp. 81-86.

Blanke, M., 1981. *Blanke, M. (1981). Shi Propulsion Losses Related to Automated Steering and Prime Mover Control, phD thesis, The Technical University of Denmark, Lyngby.,* s.l.: s.n.

Brat, G. & Jonsson, A., 2005. *Challenges in verification and validation of autonomous systems for space exploration.* 2005 IEEE International Joint Conference.

Bryson, A. E. & Ho, Y. C., 1975. Applied optimal control. In: Hemisphere, New York: s.n., p. 458.

Buniyamin, N., Wan Ngah, W., Sariff, N. & Mohamad, Z., 2011. A simple local path planning algorithm for autonomous mobile robots. *International journal of systems applications,* Volume 5, no. 2, pp. 151-159.

Caccia, M, et al., 2005. Sampling sea surfaces with SESAMO: an autonomous craft for the study of sea-air interactions. *IEEE robotics & automation magazine,* Volume 12(3), pp. 95-105.

Caccia, M. & Veruggio, G., 2000. *Guidance and control of a reconfigurable unmanned underwater vehicle.* Control Engineering Practice 8, no. 1.

CALZONI, 2014. *http://www.calzoni.com/products-services/special-naval-applications_3/unmanned-surface-vehicles_14.* [Online] Available at: http://www.calzoni.com/products-services/special-naval-applications_3/unmanned-surface-vehicles_14
[Accessed 01 July 2014].

Campbell, S., Naeem, W. & Irwin, . G. W., 2012. A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance manoeuvres. *Annual Reviews in Control,* Volume 36, no. 2, pp. 267-283.

Cao, C. et al., 2007. *Stabilization of cascaded systems via L1 adaptive controller with application to a UAV path following problem and flight test results.* In American Control Conference, 2007. ACC'07, pp. 1787-1792. IEEE.

Carpin, S. & Pillonetto, G., 2005. Motion planning using adaptive random walks. *IEEE Transactions on Robotics,* 21(1), pp. 129-136.

Carroll, K. P. et al., 1992. *AUV path planning: an A* approach to path planning with consideration of variable vehicle speeds and multiple, overlapping, time-dependent exclusion zones.* Proceedings of the 1992 Symposium on (pp. 79-84). IEEE.

Choi, J., 2012. *Model Checking for Decision Making Behaviour of Heterogeneous Multi-Agent Autonomous System,* s.l.: Phd Thesis, Cranfield University.

Choi, J., Kim, S. & Tsourdos, A., 2015. Verification of heterogeneous multi-agent system using MCMAS. *International Journal of Systems Science ,* Volume 46, no. 4 , pp. 634-651.

Choi, J., Tsourdos, A. & White, B., 2010. *Verification of Multi-Agent Systems using Formal Approach.* AIAA Guidance, Navigation, and Control Conference.

Clarke, E. M., 1997. *Model checking.* Berlin: Foundations of Software Technology and Theoretical Computer Science. Lecture Notes in Computer Science, vol 1346..

Clarke, E. M., Grumberg, O. & Peled, D., 1999. *Model checking.* s.l.:MIT press.

Cockcroft, A. & Lameijer, J., 2011. *A Guide to the collision avoidance rules.* 7 ed. s.l.:Elsevier.

Colito, J., 2007. *Autonomous mission planning and execution for unmanned surface vehicles in compliance with the marine rules of the road,* Washington: University of Washington.

Conte, G., Duranti, S. & Merz, T., 2004. *Dynamic 3D path following for an autonomous helicopter.* In Proc. of the IFAC Symp. on Intelligent Autonomous Vehicles, pp. 5-7.

Curcio, J., et al., 2005. *Experiments in moving baseline navigation using autonomous surface craft.* OCEANS, 2005. Proceedings of MTS/IEEE, pp. 730-735. IEEE, 2005..

Defense Media Network, 2013. *Zycraft Independent Unmanned Surface Vehicle (IUSV).* [Online]
Available at: http://www.defensemedianetwork.com/stories/zycraft-independent-unmanned-surface-vehicle-iusv/
[Accessed 07 June 2016].

Defense Update, 2005. *Stingray.* [Online]
Available at: http://defense-update.com/products/s/stingray.htm
[Accessed 8 July 2014].

Defense Update, 2007. *Interceptor - Unmanned Surface Vessel Unveiled.* [Online]
Available at: http://defense-update.com/products/i/interceptor.htm
[Accessed 1 August 2014].

Dowek, G. & Munoz, C., 2007. *Conflict detection and resolution for 1, 2,... N aircraft.* In 7th AIAA ATIO Conf, 2nd CEIAT Int'l Conf on Innov and Integr in Aero Sciences, 17th LTA Systems Tech Conf; followed by 2nd TEOS Forum (p. 7737)..

ECA Group, 2014. *ECA Group confirms its role as a global player in Unmanned Surface Vehicles.* [Online]
Available at: http://www.ecagroup.com/en/financial/eca-group-confirms-its-role-global-player-unmanned-surface-vehicles
[Accessed 1 July 2014].

Eichhorn, M., 2015. Optimal routing strategies for autonomous underwater vehicles in time-varying environment. *Robotics and Autonomous Systems,* Volume 67, pp. 33-43.

Fossen, T. I., 1994. *Guidance and control of ocean vehicles.* s.l.:John Wiley & Sons Inc..

Galdino, A. L., Munoz, C. & Ayala-Rincón, M., 2007. *Formal verification of an optimal air traffic conflict resolution and recovery algorithm.* Berlin Heidelberg, In International Workshop on Logic, Language, Information, and Computation, pp. 177-188..

Garau, B., Alvarez, A. & Oliver, G., 2005. *Path planning of autonomous underwater vehicles in current fields with complex spatial variability: an A* approach.* In Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on (pp. 194-198). IEEE..

Garau, B., Alvarez, A. & Oliver, G., 2005. *Path planning of autonomous underwater vehicles in current fields with complex spatial variability: an A* approach.* Proceedings of the 2005 IEEE International Conference.

Garau, B. et al., 2014. Path planning for autonomous underwater vehicles in realistic oceanic current fields: Application to gliders in the western mediterranean sea. *Journal of Maritime Research,* Volume 6, no. 2, pp. 5-22.

Gasquet, O., Herzig, A., Said, B. & Schwarzentruber, F., 2014. *Model Checking.* s.l.:Springer Basel.

Ghosh, S. K. & Mount., D. M., 1991. An output-sensitive algorithm for computing visibility graphs. *SIAM Journal on Computing,* Volume 20, no. 5, pp. 888-910.

Goldberg, D. E. & Holland, J. H., 1988. Genetic algorithms and machine learning." Machine learning. In: *Machine learning.* s.l.:s.n.

Hsu, D., Latombe, J. & Motwani, R., 1997. Path planning in expansive configuration spaces. *Robotics and Automation,* Volume 3, pp. 2719-2726.

Humphrey, L., 2012. *Model checking UAV mission plans.* AIAA Modeling and Simulation Technologies Conference.

Humphrey, L. R. & Patzek, M. J., 2013. *Model checking human-automation UAV mission plans.* AIAA Guidance, Navigation, and Control (GNC) Conference.

Huth, M. & Ryan, M., 2004. *Logic in Computer Science: Modelling and reasoning about systems.* s.l.:Cambridge university press.

Ibarra-Zannatha, J., Sossa-Azuela, J. & Gonzalez-Hernandez, H., 1994. A new roadmap approach to automatic path planning for mobile robot navigation. *Systems, Man, and Cybernetics,* Volume 3, pp. 2803-2808.

Isherwood, R. M., 1973. *Wind resistance of merchant ships.* RINA Supplementary Papers, 115.

Jørgensen, K. A. N., 2011. *State Estimation with Wave Filtering for an Unmanned Surface Vehicle: by utilizing acceleration feedback.* Master's thesis, Institutt for teknisk kybernetikk.

Kaluđer, H., Brezak, M. & Petrović, I., 2011. *A visibility graph based method for path planning in dynamic environments.* , MIPRO, 2011 Proceedings of the 34th International Convention.

Khatib, O., 1987. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation,* 3(1), pp. 43-53.

Kim, J., Pearce, R. A. & Amato, N. M., 2003. *Extracting optimal paths from roadmaps for motion planning.* Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference.

KL Security Review, 2010. *Singapore develops Unmanned Combat Surface Vehicle.* [Online]
Available at: http://www.klsreview.com/HTML/2009Jan_Jun/20090608_02.html
[Accessed 8 July 2014].

Koay, T.-B. & Chitre, M., 2013. *Energy-efficient path planning for fully propelled AUVs in congested coastal waters.* 2013 MTS/IEEE.

Konukoglu, E., Sermesant, M., Clatz, O. & Peyrat, J.-M., 2007. A recursive anisotropic fast marching approach to reaction diffusion equation: Application to tumor growth modeling. *Information processing in medical imaging,* pp. 687-699.

Koren, Y. & Borenstein, J., 1991. *Potential field methods and their inherent limitations for mobile robot navigation.* 1991 IEEE International Conference.

Kothari, M., Postlethwaite, I. & Gu, D.-W., 2010. A suboptimal path planning algorithm using rapidly-exploring random trees. *International Journal of Aerospace Innovations,* 2(1), pp. 93-104.

Kuffner, J. & Latombe, J., 2000. Interactive manipulation planning for animated characters. *Computer Graphics and Applications,* pp. 417-418.

Kularatne, D., Bhattacharya, S. & Hsieh, M. A., 2016. *Time and Energy Optimal Path Planning in General Flow.* robotics proceedings.

Kuwata, Y., Wolf, M. T., Zarzhitsky, D. & Huntsberger, T. L., 2011. *Safe maritime navigation with COLREGS using velocity obstacles.* In Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, pp. 4728-4734. IEEE.

Kuwata, Y., Wolf, M. T., Zarzhitsky, D. & Huntsberger, T. L., 2014. Safe maritime autonomous navigation with COLREGS, using velocity obstacles. *IEEE Journal of Oceanic Engineering,* 39(1), pp. 110-119.

Ladkin, P. B. & Leue, S., 1992. *An analysis of Message Sequence Charts.* Universität Bern: Institut für Informatik und Angewandte Mathematik.

Larson, J. et al., 2007. *Advances in autonomous obstacle avoidance for unmanned surface vehicles.* SPACE AND NAVAL WARFARE SYSTEMS CENTER SAN DIEGO CA.

Lee, S., Kwon, K. & Joh, J., 2004. A fuzzy logic for autonomous navigation of marine vehicles satisfying COLREG guidelines. *International Journal of Control, Automation, and Systems,* Volume 2(2), pp. 171-181.

Lee, T. et al., 2015. Energy efficient path planning for a marine surface vehicle considering heading angle. *Ocean Engineering,* Volume 107, pp. 118-131.

Lerda, F. & Visser, W., 2001. *Addressing dynamic issues of program model checking.* Berlin Heidelberg, International SPIN Workshop on Model Checking of Software.

Leue, S., Mehrmann, L. & Rezai, M., 1998. *Synthesizing ROOM models from message sequence chart specifications.* s.l.:University of Waterloo.

Liao, Y., Wan, L. & Zhuang, J., 2011. Backstepping dynamical sliding mode control method for the path following of the underactuated surface vessel. *Procedia Engineering,* Volume 15, pp. 256-263.

Liquid Robotics, 2013. *Liquid Robotics.* [Online] Available at: https://www.liquid-robotics.com/ [Accessed 8 July 2014].

Liu, Z., Zhang, Y., Yu, X. & Yuan, C., 2016. Unmanned surface vehicles: An overview of developments and challenges. *Annual Reviews in Control ,* Volume 41, pp. 71-93.

Li, Y., Chen, H., Er, M. J. & Wang, X., 2011. Coverage path planning for UAVs based on enhanced exact cellular decomposition method. *Mechatronics,* Volume 21, no. 5, pp. 876-885.

Lloyd’s Register Rulefinder, 2005. *COLREGS - International Regulations for Preventing Collisions at Sea,* s.l.: s.n.

Loe, G., 2008. *Collision Avoidance for Unmanned Surface,* s.l.: Master thesis, Norwegian University of Science and Technology (NTNU).

Lolla, T., Haley, J. & Lermusiaux, P. F. J., 2015. Path planning in multi-scale ocean flows: Coordination and dynamic obstacles. *Ocean Modelling,* Volume 94, pp. 46-66.

Lolla, T., HaleyJr., P. J. & Lermusiaux, P. F. J., 2014. Time-optimal path planning in dynamic flows using level set equations: realistic applications. *Ocean Dynamic,* Volume 64(10), pp. 1399-1417.

Lolla, T. et al., 2012. Path planning in time dependent flow fields using level set methods. *Robotics and Automation,* Issue IEEE, pp. 166-173.

MahmoudZadeh, S., Powers, D. & Yazdani, A., 2016. *Differential Evolution for Efficient AUV Path Planning in Time Variant Uncertain Underwater Environment.* arXiv preprint arXiv:1604.02523 (2016)..

Manley, J.E, 2008. *September. Unmanned surface vehicles, 15 years of development.* OCEANS 2008 (pp. 1-4). IEEE.

Marco, B., Gabriele, B., Caccia, M. & Lapierre, L., 2012. *A collision avoidance algorithm based on the virtual target approach for cooperative unmanned surface vehicles.* CDC'2012: 51st Conference on Decision and Control.

Masehian, E. & Amin-Naseri, M. R., 2004. A voronoi diagram-visibility graph-potential field compound algorithm for robot path planning. *Journal of Field Robotics,* Volume 21, no. 6, pp. 275-300.

Michael, Q. M., Bak, T. & Zamanabadi, R. I., 2004. *Multi-robot planning: A timed automata approach.* 2004 IEEE International Conference.

Mirebeau, J.-M., 2014. Anisotropic fast-marching on Cartesian grids using lattice basis reduction. *SIAM Journal on Numerical Analysis,* Volume 4 , pp. 1573-1599.

Molnar, L. & Veres, S. M., 2009. *Verification of autonomous underwater vehicles using formal logic.* Control Conference (ECC).

Motwani, A, 2012. *A survey of uninhabited surface vehicles,* Plymouth: Marine and Industrial Dynamic Analysis, School of Marine Science and Engineering Plymouth University.

Murphy, R, Stover, S, Pratt, K & Griffin, C, 2006. *Cooperative damage inspection with unmanned surface vehicle and micro unmanned aerial vehicle at hurricane wilma.* Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on (pp. 9-9). IEEE..

Naeem, W., Irwin,, G. W. & Yang, A., 2012. COLREGs-based collision avoidance strategies for unmanned surface vehicles. *Mechatronics,* 22(6), pp. 669-678..

Naeem, W., Sutton, R. & Chudley, J., 2006. *Modelling and control of an unmanned surface vehicle for environmental monitoring.* UKACC International Control Conference.

Naval Technology, 2013. *Protector Unmanned Surface Vehicle (USV), Israel.* [Online]
Available at: http://www.naval-technology.com/projects/protector-unmanned-surface-vehicle/
[Accessed 07 June 2016].

Naval Technology, 2014. *Fleet-Class Common Unmanned Surface Vessel (CUSV), United States of America.* [Online]
Available at: http://www.naval-technology.com/projects/fleet-class-common-unmanned-surface-vessel-cusv/
[Accessed 1 July 2014].

NAVALDRONES, 2017. *Exclusive Content on Unmanned Naval Systems.* [Online]

Available at: http://www.navaldrones.com/Unmanned-Surface-Vehicles.html [Accessed 1 January 2017].

Nelson, D. R., Barber, B., McLain, T. W. & Beard, R. W., 2007. Vector field path following for miniature air vehicles. *IEEE Transactions on Robotics,* 23(3), pp. 519-529.

NEW ATLAS, 2008. *Solar powered, Linux brained autonomous robot sailboat aims to conquer the Atlantic.* [Online]
Available at: http://newatlas.com/solar-power-linux-brain-autonomous-robot-sailboat-aims-to-conquer-the-atlantic/10517/
[Accessed 1 July 2014].

Nicollin, X., Sifakis, J. & Yovine, S., 1992. Compiling real-time specifications into extended automata. *IEEE transactions on Software Engineering ,* Volume 18, no. 9 , pp. 794-804.

Niu, H., Lu, Y., Savvaris, A. & Tsourdos, A., 2016. *Efficient path following algorithm for unmanned surface vehicle.* Shanghai, OCEANS 2016, IEEE.

Niu, H., Lu, Y., Savvaris, A. & Tsourdos, A., 2016. *Efficient Path Planning Algorithms for Unmanned Surface Vehicle.* Norway, IFAC-PapersOnLine.

Northrop Grumman, 2011. *Northrop Grumman Awarded Unmanned Surface Vessel Contract From DARPA.* [Online]
Available at: http://news.northropgrumman.com/news/releases/northrop-grumman-awarded-unmanned-surface-vessel-contract-from-darpa
[Accessed 1 August 2014].

NORTHROP GRUMMAN, 2014. *AQS-24B Minehunting System.* [Online]
Available at:
http://www.northropgrumman.com/Capabilities/ANAQS24/Pages/default.aspx
[Accessed 1 July 2014].

Park, S., Deyst, J. & How, J. P., 2007. Performance and lyapunov stability of a nonlinear path following guidance method. *Journal of Guidance, Control, and Dynamics,* 30(6), pp. 1718-1728.

Pecheur, C., 2000. *Verification and validation of autonomy software at NASA,* NASA.

Pehlivanoglu, Y. V., 2012. A new vibrational genetic algorithm enhanced with a Voronoi diagram for path planning of autonomous UAV. *Aerospace Science and Technology,* 16(1), pp. 47-55.

Perera, L., Carvalho, J. & Soares, C., 2009. *Autonomous guidance and navigation based on the COLREGs rules and regulations of collision avoidance.* Proceedings of the International Workshop “Advanced Ship Design for Pollution Prevention.

Petres, C., Pailhas, Y., Petillot, Y. & Lane, D., 2005. Underwater path planing using fast marching algorithms. *Oceans 2005-Europe,* 2(IEEE), pp. 814-819.

Phanthong, T. et al., 2014. Application of A* algorithm for real-time path re-planning of an unmanned surface vehicle avoiding underwater obstacle. *Journal of Marine Science and Application,* Volume 13, no. 1, pp. 105-116.

Plymouth University, 2011. *Springer 2 - unmanned surface vehicle.* Plymouth, https://www.plymouth.ac.uk/research/autonomous-marine-systems/springer-2.

QinetiQ Target Systems , 2014. *Barracuda™ USV-T.* [Online] Available at: http://targetsystems.qinetiq.com/en-ca/products-and-services/naval-targets/barracuda-usv-t/
[Accessed 1 July 2014].

QinetiQ Target Systems, 2013. *Hammerhead™ USV-T.* [Online] Available at: http://targetsystems.qinetiq.com/en-ca/products-and-services/naval-targets/hammerhead-usv-t/
[Accessed 1 August 2014].

Queen's University Belfast, 2014. *Unmanned Marine Vehicles.* [Online] Available at: http://www.qub.ac.uk/research-centres/EPIC/Research/EnvironmentalandMonitoringControl/UnmannedMarineVehicles/
[Accessed 8 July 2014].

Reniers, M. A., 1999. *Message sequence chart. Syntax and semantics,* Eindhoven: Technische Universiteit Eindhoven.

RESEARCH COUNCILS UK, 2015. *Unmanned Safe Maritime Operations Over The Horizon (USMOOTH).* [Online] Available at: http://gtr.rcuk.ac.uk/projects?ref=102303 [Accessed 01 January 2017].

Savvaris, A., Niu, H., Oh, H. & Tsourdos, A., 2014. *Development of Collision Avoidance Algorithms for the C-Enduro USV.* Cape Town, IFAC Proceedings Volumes 47, no. 3.

Sea Robotics, 2014. *Smart USVs.* [Online] Available at: http://www.searobotics.com/products/smart-usvs [Accessed 1 August 2014].

Shin, H.-S.et al., 2008. *UAV conflict detection and resolution for static and dynamic obstacles.* In AIAA Guidance, Navigation and Control Conference and Exhibit, p. 6521. 2008..

Sirigineedi,, G., Tsourdos, A. & White, B. A., 2011. Kripke modelling and verification of temporal specifications of a multiple UAV system. In: *Annals of Mathematics and Artificial Intelligence.* s.l.:s.n., pp. 31-52.

Sirigineedi, G., Tsourdos , A., Zbikowski, R. & White, B. A., 2009. *Modelling and Verification of Multiple UAV Mission Using SMV.* Eindhoven, The Netherlands: Proceedings FM-09 Workshop on Formal Methods for Aerospace.

Sirigineedi, G., Tsourdos, A., White, B. & Zbikowski, R., 2009. *Towards verifiable approach to mission planning for multiple UAVs.* AIAA Infotech@ Aerospace Conference and AIAA Unmanned, Unlimited Conference.

Sirigineedi, G., Tsourdos, A., White, B. & Zbikowski, R., 2010. *Kripke modelling and model checking of a multiple UAV system monitoring road network.* AIAA Guidance, Navigation, and Control Conference.

SNAME, 1950. *Nomenclature for treating the motion of a submerged body through a fluid.* The Society of Naval Architects and Marine Engineers, Technical and Research Bulletin.

Song, R., Liu, Y. & Bucknall, R., 2017. A multi-layered fast marching method for unmanned surface vehicle path planning in a time-variant maritime environment. *Ocean Engineering,* Volume 129, pp. 301-317.

Statheros, T., Howells, G. & Maier, K., 2008. Autonomous ship collision avoidance navigation concepts, technologies and techniques. *Journal of Navigation,* Volume 61(1), pp. 129-142.

Sujit, P., Saripalli, S. & Sousa, J., 2014. Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicless. *IEEE Control Systems,* Volume 34(1), pp. 42-59.

Suresh, J., Tsourdos, A., Żbikowski, R. & White, B., 2006. Kripke modelling approaches of a multiple robots system with minimalist communication: a formal approach of choice. *International journal of systems science,* pp. 37(6), pp.339-349.

Tadao, M., 1989. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE,* Volume 77, no. 4, pp. 541-580.

Teo, K., Ong, K. & Lai, H., 2009. *Obstacle detection, avoidance and anti collision for MEREDITH AUV.* OCEANS 2009, MTS/IEEE Biloxi-Marine Technology for Our Future: Global and Local Challenges.

Tidetech, 2014. *Data.* [Online] Available at: https://www.tidetech.org/data/ [Accessed 01 June 2014].

Wang, B., Dong, X. & Chen, B. M., 2010. *Cascaded control of 3D path following for an unmanned helicopter.* In Cybernetics and Intelligent Systems (CIS), 2010 IEEE Conference on (pp. 70-75).

Wein, R., Van den Berg, J. P. & Halperin, D., 2007. The visibility–Voronoi complex and its applications. Volume 6, no. 1 , pp. 66-87.

Wu, B., Wen, Y., Huang, Y. & Zhu, M., 2013. *Research Of Unmanned Surface Vessel (USV) Path-Planning Algorithm Based on ArcGIS.* ICTIS 2013: Improving Multimodal Transportation Systems-Information, Safety, and Integration.

Yilmaz, N. K., Evangelinos, C., Lermusiaux, P. F. & Patrikalakis, N. M., 2008. Path planning of autonomous underwater vehicles for adaptive sampling using mixed integer linear programming. *IEEE Journal of Oceanic Engineering,* Volume 33, no. 4, pp. 522-537.

Zadeh, S. M., Powers, D. M., Sammut, K. & Yazdani, A., 2016. *Differential evolution for efficient AUV path planning in time variant uncertain underwater environment.* arXiv preprint arXiv:1604.02523 (2016)..

Zhang, Y., Zhang, L. & Zhang, X., 2008. *Mobile robot path planning base on the hybrid genetic algorithm in unknown environment.* In Intelligent Systems Design and Applications, 2008. ISDA'08. Eighth International Conference on, vol. 2, pp. 661-665. IEEE, 2008..

Zhuang, J.-y., Su, Y.-m., Liao, Y.-l. & Sun, H.-b., 2011. Motion planning of USV based on marine rules. *Procedia Engineering,* Volume 15, pp. 269-276.

ZyCraft, 2014. *USV VIGILANT.* [Online] Available at: http://www.zycraft.com/products/iusv-vigilant/ [Accessed 1 July 2014].

APPENDICES

Appendix A Civil USVs

USV names	Manufacturer	Application	Specification and Characteristic
ASV Roboat (NEW ATLAS, 2008)	The Austrian Society of Innovative Computer Sciences	Breaking the record for the longest journey made by a fully autonomous sailboat, all while collecting data on a Baltic Sea porpoise.	Boat type laerling Length 3.75 m Total displacement: 300kg Sail area: 3.5 m^2 main + 1.9 m^2 jib Self-righting design Computer: Mini-ITX with linux, 800 MHz / 512 MB RAM, 4 GB CF Card, Control software in Java and C++ Communication: Wifi, GPRS/UMTS, Iridium satellite modem Sensors: Position and speed over ground (GPS), Speed though water, Battery voltage and power consumption, Ultrasonic wind sensor (direction and speed), Tilt-compensated compass (heading and heel), humidity, air and water temperature, air pressure, depth Actuators: Sheet linear drive, Rudder gear, balanced Energy balance: 35W avg.power consumption, 1.5m^2 solar panels (285 Wp), Direct Methanol Fuel Cell (65W), 4.6kWh lithium ion batteries Characteristics： Wind as the only source of propulsion No remote control Entire control system is on board Collision avoidance Energy self-sufficient
Catamaran USVs (Sea Robotics, 2014)	sea robotics unmanned systems	Survey, Surveillance, Science, Research, Navigation and Communication Aide, ROV support, AUV tender, Arctic Operations.	Catamaran 11 meters open water USV Specification: Long Endurance Offshore Diesel Electric Hybrid USV with substantial Payload, Up to 60 days endurance. Catamaran 2.0-4.0 meters collapsible USV

			Specification: All the features of a larger USV, yet ships in a pallet crate. Assembled on site from light weight parts. Catamaran 4.0 meters general purpose USV Specification: Small but capable with numerous applications specific designs, highly maneuverable, high efficiency hulls. Catamaran 5.7 meters general purpose USV Specification: Large payload deck, Long endurance on battery, numerous alternate energy storage options, highly maneuverable, high efficiency hulls. Specification:1.5 m,5 m,7 m,and 9 m high speed USVs. Unmanned and manned configurations. Security, Surveillance, target/test applications.
C-Enduro USV (ASV Global Ltd, 2013)	ASV Ltd	Data collection and carry out traditional survey	Length: 4.2m Beam: 2.4m Draft: 0.4m Speed: up to 6.5 knots Solar Panel: Generating 1200 W energy Diesel generator system: Providing a peak charging power of 2.5KW Wind turbine: 720W Long endurance USV with energy harvesting technology combined with an efficient self-right hull.
China Unmanned Surface Vehicle (NAVALDRONES, 2017)	China’s Ministry of Transport and Maritime Bureau	Hydrographic surveys of shoals and reefs	Maximum speed: 18 knots Commercial radar Forward looking sonar Multi-beam side scan sonar for underwater survey work It can be controlled remotely or navigate autonomously.
C-Worker USV (ASV Global Ltd, 2015)	ASV Ltd	Carry out a variety of offshore and inshore survey tasks	Length: 4.17m Beam: 1.58m Draft: 0.41m Speed: Up to 7 knots

			Endurance: Up to 48 knots C-Worker 4's waterjet propulsion system makes it an ideal solution for shallow water survey. The vehicle has the ability to integrate a range of standard survey payloads including ADCP, USBL, PAM and CTD.
Inspector MK2 (ECA Group, 2014)	ECA robotics	Data Acquisition & Post-processing of subsea images: Sediment analysis, Bathymetry, Magnetic mapping, Imagery, 3D Post-processing	Mission oriented Sensors : Multi Beam Echo Sounder, Sub Bottom Profiler, Side Scan Sonar, Magnetometer, EDGETECH 4600 Unmanned Surface Vehicle INSPECTOR features a very shallow draught and one patented rotary bow arm for shallow & very shallow water survey, detection and object classification.
Mistral USV (NAVALDRONES, 2017)	MSHIPCO, LLC	Mistral's cargo bay area is designed to carry large modular sensor packages, passengers or equipment.	The USV can be optionally manned and is designed to operate from the well deck of a larger mothership without the need of a trailer. The USV's Command & Control (C2) has a self-tuning autopilot designed to simplify operations with autonomous features for coordinated operations, fail-safe scenarios, and pre-programmed objectives.
Powervent USV (NAVALDRONES, 2017)	Spatial Integration Systems, Inc.	Powervent (AMN1) is an experimental unmanned surface vessel to test the integration between Multi-Platform Sensor System (MPSS) and USV	Length: 11 meters Sensors: 360 degrees high-resolution cameras, LIDAR, and radar, integrated with multi-sensory data fusion, multi-dimensional processing algorithms, and artificial intelligence technology.
Tianxiang One USV (NAVALDRONES, 2017)	Shenyang Shin Kong Corporation, a susidiary of China Aviation Science and Industry Corporation.	The vessel is designed for meteorological survey and conducted those operations in support of the sailing competition for the 2008 Olympic Games in Beijing.	Length: 6.5 m Tianxiang can be operated autonomously or with a ground-based operator.
Wave Glider SV2 and SV3 (Liquid Robotics, 2013)	Liquid Robotics	With its ability to stay out at sea for long durations through all weather conditions, and communicate real-time data from the surface of the ocean, the Wave Glider SV Series is helping to advance mankind's ability to observe and understand the world's oceans.	Wave Glider SV2: Payload volume 40L Payload weight 18 kg High performance payload computers Towing ability: capable Wave Glider SV3: Payload volume 93L Payload weight: 45 kg High performance payload computers High speed networking (WIFI, cellular) Towing ability: optimized

			Designed for high deck ship recovery The Wave Glider SV Series are the first unmanned autonomous marine robots to use the ocean's endless supply of wave energy for propulsion (no manpower, no emissions, no refueling).

Appendix B Military USVs

USV names	Manufacturer	Application	Specification and Characteristic
Anti-Submarine Warfare (ASW) Continuous Trail Unmanned Vessel (ACTUV) (Northrop Grumman, 2011)	Northrop Grumman	An autonomous vessel that can perform both ASW tracking and intelligence, surveillance and reconnaissance functions.	High-fidelity surface-navigation sensors and system constraints would help ensure compliance with the Convention on the International Regulations for Preventing Collisions at Sea (COLREGS) and with maritime law.
AN/AQS-24A Airborne Mine hunting System (NORTHROP GRUMMAN, 2014).	Northrop Grumman	Mine hunting	High-Resolution Sonar: High speed (18 knot) operation 3" x 3" resolution Object detection and classification Target box cuing Laser Line Scanner: <1" resolution High speed operation Optical imagery for target I.D. Field proven/COTS based High-resolution, side scan sonar for real time detection, localization and classification of bottom and moored mines at high area coverage rates; Laser line scanner provides precision optical identification of underwater mines and other objects of interest; Simultaneous operation of Laser with sonar provides gap-filling capability and tactically advanced Target Detect and Target ID modes; Thousands of tow hours logged from a variety of platforms;
Barracuda TM USV-T (QinetiQ Target Systems , 2014)	Meggitt Training Systems Canada	Mine sweeping operations and mobile, inshore undersea warfare.	Length: 7.24 m Boat beam: 2.75 m

			Weight: 2,074 kg Engine: D4, Volvo turbocharged, aftercooled, inline 4-cylinder, marine diesel Engine Performance: 225 horsepower Fuel Capacity: 208 litres (46 Imp gallons) (55 U.S. gallons) Payloads: Scanning projectile Impact Evaluation System (SPIES) Active Radar Augmentation System Passive Radar Augmentation (20 – 500 m^2, I-Band) Visual Augmentation (Smokes, Flags, Flares, Strobes) Capable of carrying auto-winch Operating Temperature: -30° to +50°C (+15° to +122°F) Storage Temperature: -40° to +60°C (-40° to +140°F Maximum Speed: 36+ knots in SS3 Speed/endurance/range: 36+ knots, 5 hrs, 180 nm 20 knots, 15 hrs, 300 nm Payload/speed: 500 lbs/35 knots Control TM Range: Over-the-horizon (unlimited distance) tested to 6000 km Video TM Range: >10 nm (subject to control station antenna height) Control System: Universal Target Control Station (UTCS) – STANAG 4856 compliant
Calzoni unmanned surface vehicle U-RANGER (CALZONI, 2014)	Calzoni, which is a leading company in the Aerospace & Defence market providing solutions for Marine Handling & Lighting Solutions.	Minesweeping operation	The Mine-hunting Propulsion System, with two or three retractable azimuthal thrusters, provides an unparalleled manoeuvrability to the vessel in all the MCM operations. The vessel design with separated Mine-hunting Propulsion and the transit propulsion. The system includes a dedicated autopilot MHAP-20 specifically developed for Mine-hunting operation featuring specific algorithms to enhance dynamic positioning precision and minimize required power and hence noise

			Shock tested with explosion at sea (MIL – S – 901); Marginal residual magnetic signature; Extremely low underwater noise emission; Extremely high manoeuvrability level; Propeller thrust up to 3000 kg; Manual assisted: remote manual control by means of joystick/keyboard of the vehicle heading and thrust; Automatic: automatic track-keeping of planned tracks; Autonomous: automatic mode without radio-link.
Eclipse USV (NAVALDRONES, 2017)	Abu Dhabi boat builder Al seer Marine and 5G International of West Palm Beach, Florida	Serving as a surveillance or patrol craft	Length: 11 meters Maximum speed: 50 knots Hybrid propulsion system Twin turbo-charged Fiat 500 horsepower N67500 diesel engines Roll-Royce Kamewa A pair of 13kw electric motors Simrad high-resolution radar FLIR Bosch day/night cameras The USV can be controlled line-of-sight via radio link or over-the-horizon with a satellite link or perform autonomous missions
Edredon USV (NAVALDRONES, 2017)	Naval Academy in Gdynia, Gdansk University of Technology	The vehicle is designed with a modular approach to conduct a variety of maritime missions, which includes being weaponized with a machine gun or grenade launcher and modules are being considered which allows for the launch and recovering of an autonomous underwater vehicle for duties like mine-hunting.	Radar, cameras, infrared sensors, Yanmar ZT350 132kW/180 hp sterndrive engine, Endurance 30 hours depending on sea state The vehicle has several autonomous functions, including the ability to return to base upon loss of communications. Edredon is operated by a two man crew in a containerized control station.
Espadon Unmanned Surface Vehicle (NAVALDRONES, 2017)	Espadon is a demonstration program contracted for France's DGA (Directorate of armament) by DCNS, Thales,	Performing autonomous mine-hunting missions.	Length: 17 meters Weight (dry): 25 tons Body: Catamaran hulled ship The Espadon (French Swordfish) Project is comprised of an optionally-manned surface vessel named Sterenn Du (Black Star) and accompanying

	and ECA in 2009.		autonomous underwater vehicles (AUV). The mother ship was launched in 2010 and carries a towed sonar and autonomous underwater vehicles designed to detect and neutralize mines. Sterenn Du is capable of retrieving AUVs (including the Alister) via an unique towed capture device for recharging and data upload.
Fleet-class Common Unmanned Surface Vessels (CUSV) (Naval Technology, 2014)	Textron Systems Advanced Systems	Collaborative unmanned mine-hunting and mine-neutralization operations	Utilizing the U.S. Navy's tactical decision aid — called MEDAL, deployed an L-3 Klein 5000 V2 Side Scan Sonar to mine hunt the suspected minefield, The CUSVs were outfitted with Harris SeaLancetTM RT-1944/U data links, which relayed all sonar and vehicle control information to a shore-based AAI Universal Command & Control Station (UCCS). featuring commercial off-the-shelf modular open architecture, a reconfigurable payload bay, and high tow force capability. Its maritime command and control system is based upon AAI Unmanned Aircraft Systems' combat-proven One System(R) architecture. Sliding autonomy, enabling autonomous and man-in-the-loop vessel operations, and non-lethal weapon common payload command and control.
Globida USV (NAVALDRONES, 2017)	Sponsored by the Scientific and Technological Research Council in Turkey and manufactured by Global Teknik		Length: 4 m Endurance: 120 hours on a 300 liter gas tank Radar, sonar, and night vision camera This vehicle can be controlled remotely or run autonomously.
Hammerhead TM USV-T (QinetiQ Target Systems, 2013)	Meggitt Training Systems Canada	Simulating a Fast Inshore Attack Craft (FIAC) in a multi-vehicle swarm of up to 40 vessels simultaneously. The Hammerhead TM also excels in replicating Fast Attack Crafts (FAC) naval threats.	Hull Length: 5.2m Boat beam: 1.4 m Weight: 900 kg Engine: MerCruiser 3.0L Engine Performance: 135 horsepower Fuel Capacity: 161 liters Payloads: Visual Augmentation (Smokes, Flags, Flares, Strobes) Passive Radar Augmentation (20 - 500m^2, I-Band) Active Radar Augmentation (RF-SAS) System Video TM

			Operating Temperature: -10° to +40°C (+15° to +122°F) Storage Temperature: -20° to +60°C (-40° to +140°F) Maximum Speed: 35+ knots in Sea State 3 > 35 knots in Sea State 2 (prop dependant) Speed/endurance/range: 12 hours @2400 RPM 10knots 8 hours @3050 RPM 20knots 5 hours @4100 RPM 30knots Payload/speed: 500 lbs/35 knots Control System: MTSC UTCS (STANAG 4586 compliant) Optional Video TM Range 5nm (subject to CS antenna height)
Interceptor (Defense Update, 2007)	The Interceptor unmanned surface vessel (USV) is developed under cooperation between AAI, and two U.S. - Marine Robotic Vessels International (MRVI) and Sea Robotics Company (SRC).	Security and public service applications such as anti-piracy patrol, harbour security and oil rig surveillance.	Length: 6.5 meters Engine: 266 Steyr multi-fuel Propulsion: Hamilton waterjet The Interceptor can operate autonomously commanded by an on-board mission computer and navigation system and it can also operate collision avoidance by applying onboard sensors. The control system is compatible with the same protocol used with AAI Corp's UAV systems (Shadow and Aerosonde) to provide a seamless integration between unmanned aerial and surface vehicles.
Inspector MK1 (ECA Group, 2014)	ECA robotics	It has been developed for the training of Navy crews in self defence against asymmetrical threats and it also serves for development and certification of weapon	Inspector MK1 is a 7meters RHIB, easy to deploy Unmanned Surface Vehicle.
Modular Unmanned Surface Craft Littoral (MUSCL) (NAVALDRONES, 2017)	U.S. Naval Surface Warfare Center Panama City Division	MUSCL's primary mission is Intelligence, Surveillance, and Reconaissance (ISR) in the riverine environment.	Length: 6 feet Width: 2 feet Maximum: knots Endurance: Less than 3 hours Weight: 85 pounds

			MUSCL is a two man-portable vessel designed to be carried aboard the U.S. Navy's Riverine Patrol Boat (RPB) and Riverine Assault Craft (RAC).
Piraya USV (NAVALDRONES, 2017)	Kockums AB, the Swedish subsidiary of ThyssenKrupp Marine Systems.	Patrolling costal waters	Length: 4 m Beam: 1.4 m Weight: 400 kg Maximum: 20 knots Arc-mounted video/IR cameras CBRN-sensors Modular flexibility and the possibility to integrate controls with other USV systems and legacy control and communication systems are important features
Protector Unmanned Surface Vehicle Israel (Naval Technology, 2013)	Rafael Advanced Defence Systems.	Performing several mission critical operations without revealing its identity to hostile source and minimizing safety risk to the sailors and armed forces by avoiding them to be directly in contract with potentially mission critical operations.	Length: 11 m Payload: An enhanced remotely controlled water canon system for non-lethal and fire fighting capabilities, spotlight-N multi-sensor electro-optical (EO) image system, Spike LR missiles, comprehensive command and control system, non-line-of-sight (NLOS) communication, advanced stabilization, public address (PA) system, and light projector Sensor: Several sub-systems including a radar, an electro-optical director (EOD), and a 360º panoramic camera Weapon system: The weapon system is integrated with a Toplight electro-optic system to detect and target the threats. Engine: two Caterpillar C7 diesel engines The 9m version of the Protector USV is equipped with a single diesel engine and water jet propulsion.
Re-configurable USV (ASV Global Ltd, 2012)	Thales UK and Autonomous Surface Vehicle (ASV)	To be a stable platform with excellent slow speed and towing capabilities, and highly reliable & cost effective.	Length: 11.5m Beam: 3.5m Maximum speed: 25 knots Deployable from military platforms, craft of opportunity and from shore/harbour Air transportable;

			Payload flexibility for all MCM systems – unmanned underwater vehicles, towed sonar, disposal systems, minesweeping; Stable platform with excellent slow speed and towing capabilities; Highly reliable & cost effective;
Seastar USV (NAVALDRONES, 2017)	Aeronautics Defence Systems	The USV is designed to conduct port security, ISR, coastal patrol, target designation, and other missions.	Length: 11m Width: 3.5 m Weight: 6000 kg Fuel capacity: 1000 liters Marine Diesel Engines: 2* 470 Hp Maximum speed: 45 knots Endurance: 300 nautical mile mission plus 10 hours on-station time Payload: Day/Night (EO/IR) sensors, Target acquisition sensors, ESM/ECM ELINT/COMINT, Sonar, ublic address system. Non-Lethal / Weapon Systems (Water/Noise/Stun), Maritime stabilized Gun and Fire control system SeaStar is a versatile platform for various applications.
Silver Marlin USV (NAVALDRONES, 2017)	Elbit Systems	The USV is designated for intelligence, surveillance and reconnaissance (ISR) missions, force protection/anti-terror missions, anti-surface and anti-mine warfare, search and rescue missions, port and waterway patrol as well as electronic warfare.	Length: 10.67 m Weight: 4,000 kg Payload: 2,5000 kg Endurance: 24 hours Maximum speed: 45 knots Compact Multi-Purpose Advanced Stabilized System (Compass) sensor turret CCD TV camera 35 micron FLIR Laser: Aiming, Rangefinder, Designator and Target Illuminator Small boat detection – 6km; medium -16km; aircraft – 15km Range of 500 n miles 7.62 mm Overhead Remote Control Weapon System – day/night and fire-on-the-move.

Stingray USV (Defense Update, 2005)	Elbit Systems	The USV is designed for homeland security and coast guard applications including clearing shipping lanes and underwater search missions. Potential naval combat applications include target identification, Intelligence reconnaissance and surveillance (ISR) missions. Other applications include EW and ELINT.	Maximum speed: 40 knots Endurance: 8 hours Payload weight: 150 kg Payload: day and night electro-optical stabilized payload, cruise sensors, and a stabilization system which prevents capsizing. The USV is controlled from a portable control station, from which operators can monitor and operate the mission payloads and perform mission planning.
Venus USV (KL Security Review, 2010)	Singapore Technologies Electronics Limited (ST Electronics)	This unmanned combat surface vehicle uses the Venus USV as combat platform. It can be modified based on combat missions and fitted with a variety of task modules for the specific combat mission. The combat missions may include fleet escorting, anti-mines, precision shooting and anti-submarine missions.	Length: 9 m Maximum speed: 35 knots Endurance: 8 hours Hull material: strong synthetic Carrying capability: 1000 kg
Zycraft Independent Unmanned Surface Vehicle (IUSV) (ZyCraft, 2014)	Singapore-based Zycraft, which is a company offering unique high speed remotely-operating independent unmanned surface vehicles (IUSV) for high endurance maritime missions.	IUSV is designed for open ocean missions to support naval force or provide merchant ship escort through pirate-prone waters.	Length: 16.5 m Beam: 3.6 m Maximum Speed: >30 knots Cruise Speed: 12 knots Payload Fuel and Cargo: 7,000 kg Full Load Weight: 13,000 kg Lightship Weight: 6,000 kg Range @ Cruise: 1,5000 nm (Standard Package) Patrol Endurance: >30 days

Appendix C Research USVs

USV names	Manufacturer	Application	Specification and Characteristic
AEOS-1 USV (Murphy, R, et al., 2006)	Centre for Ocean Technology (COT) of the College of Marine Science, University of South	Cooperating with UAV to show the state of underwater structures, schools of small fish swimming, and find the railings from the collapsed section of a pier.	AEOS-1 USV is equipped with a Sound Metric Dual frequency Identification Sonar (DIDSON) for

	Florida (USF), St. Petersburg		
Kayak USVs (Curcio, J., et al., 2005)	MIT	Supporting the moving longbaseline navigation of UUVs and used as the network nodes in naval application	
ROAZ II USV (Almeida, et al., 2009)	Autonomous Systems Laboratory, Instituto Superior de Engenharia do Porto	ROAZ II USV is developed for research purpose, namely design issues, navigation and control problems, multiple robot coordination and environmental perception questions.	Radar based collision avoidance
SeaFox USV	Virginia Centre for Autonomous Systems	Building a detailed map of the area and acquire high-resolution imagery that can inform mission planning and execution	SeaFox USV is an aluminium hulled, rigid inflatable boat
SESAMO USV (Caccia, M, et al., 2005)	Institute of Intelligent Systems for Automation, Genova, Italy	Oceanographic research	The stability, payload capacity and ease of deck access make catamarans a compelling choice for academic USVs
Springer USV (Plymouth University, 2011)	Plymouth University, UK	The objective of this research programme is to design and build a new advanced intelligent integrated navigation and autopilot (IINA) system with adaptive capabilities, which will be complemented with a visual SLAM algorithm that will enable continuous operation even in the face of a loss of GPS signal. Autopilot based on a combination of on-line closed loop identification (CLID) and model predictive control (MPC) will be developed, and data fusion for navigation using interval Kalman filters (IKF) will be explored	Length: 4 m Width: 2.3 m Displacement: 0.6 tonnes Propulsion system: two propellers powered by a set of 24V 74lbs MINN KOTA RIPTIDE transom mount saltwater trolling motors Hull: medium waterplane twin hull (MWATH) Leak sensors