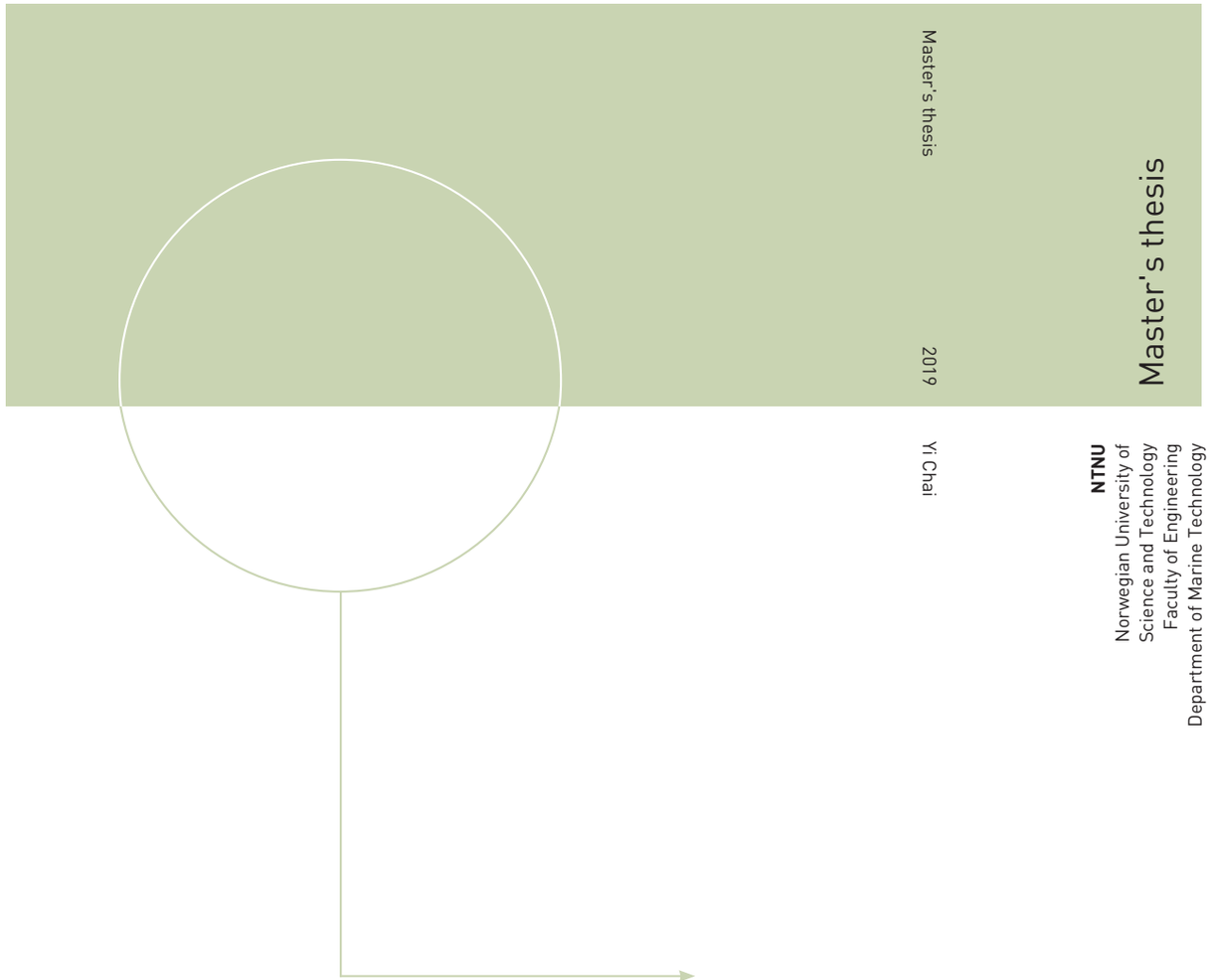

Master's thesis

2019

Yi Chai

NTNU
Norwegian University of
Science and Technology
Faculty of Engineering
Department of Marine Technology

Yi Chai

Hybrid Collision Avoidance with Moving Obstacles

June 2019

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Hybrid Collision Avoidance with Moving Obstacles

Yi Chai

Marine Technology
Submission date: June 2019
Supervisor: Vahid Hassani, IMT

Norwegian University of Science and Technology
Department of Marine Technology

Problem Description

Background:

Over the past few years, the maritime sector has witnessed an increasing interest in use of autonomous ships and in particular Autonomous Surface Vehicles (ASV) in complex applications with high associated risks. The development of autonomous ships provides much more costeffective and environmentally friendly vessel types. One of the important aspects in design and operation of ASVs are guidance and navigation in congested waterways.

This thesis proposal aims to contribute to the development of a new hybrid COLAV method based on the integration of deliberate path planning and reactive collision avoidance, that incorporates the dynamics of the vehicles and is able to generate a collision-free trajectory handling both static and moving obstacles.

Work Description:

1. Investigate research backgrounds on the following aspects:

   - Modeling of the surface vessel
   - Bézier curve and differential flatness property
   - Reactive dynamic window method
   - PLOS path tracking algorithm

2. Develop a hybrid COLAV method based on deliberate and reactive methods, that shall:

   - generate a global desired path towards goal, taking dynamics of ASV and static obstacles into account.
   - react to rapidly changing dynamic obstacle to avoid collision while aligning with global path.
   - include the development of interface between deliberate and reactive COLAV methods.

3. Implement the proposed hybrid COLAV algorithm in PYTHON.

4. Evaluate the performance and robustness of the algorithm through numerical simulations.

Abstract

A considerable amount of work has been done in the field of Autonomous Surface Vehicles (ASV) over the past few decades. Autonomous path planning and collision avoidance (COLAV) are essential for ASV navigating in unknown or partially known environment with both static and moving obstacles in the vicinity of the vehicle. This thesis proposes a hybrid COLAV approach based on the integration of a global path planning algorithm and a reactive collision avoidance approach to address the COLAV issue with the presence of the moving obstacles.

Bézier curves are exploited as the basis for global path planning, which is formulated within the framework of optimization, involving constraints like continuity, boundary conditions, static obstacles, etc. Property of differential flatness is used to assign a cost to each path that reflects the dynamic capabilities of the vehicle, yielding the optimal path by minimizing the objective function. In addition, a combined pure pursuit and line-of-sight (PLOS) steering law is implemented to follow the global path.

As a reactive collision avoidance technique, dynamic window (DW) algorithm is employed to search for optimal velocity pairs which ensure collision-free trajectory handling both static and dynamic obstacles. Extensive modification has been done to adapt original DW algorithm to the hybrid COLAV method by incorporating a path alignment function into the objective function. In particular, the interface between the deliberative and reactive method is developed, enabling the vehicle to simultaneously track the generated global path towards the given goal and avoid local collision.

The performance of this hybrid COLAV method has been evaluated through numerical simulations, and the hybrid COLAV method performs very well handling both static and moving obstacles. Besides, robustness to noisy measurement has also been tested by considering Gaussian noise with different standard deviations. Moreover, the major downside of the reactive DW algorithm, high sensitivity to local minima, is addressed with the guidance of the global path. However, for future work, it should be further investigated to perform maneuvers compliant with the International Regulations for Preventing Collisions At Sea (COLREG).

Preface

This master thesis has been written as a compulsory part of the Master's degree in Marine Technology at Norwegian University of Science and Technology (NTNU) during the spring semester of 2019. The work done on this thesis has been challenging but rewarding. After months of hard work, it eventually paid off, which has been a precious and unforgettable experience of my life.

I would like to express my great gratitude to my supervisor Prof. Vahid Hassani for his generous help and support throughout the work of this thesis. Besides, I would like to thank my fellow students, Luhao Shi and Yuan Tian, for their company and interesting discussion during the semester. Last but not least, I would also like to give a special thanks to my parents for their incredible support and encouragement.

Table of Contents

List of Tables

List of Figures

Abbreviations

ASV	=	Autonomous Surface Vehicle
AUV	=	Autonomous Underwater Vehicle
CCD*	=	Complete Coverage D*
COLAV	=	Collision Avoidance
COLREG	=	International Regulations for Preventing Collisions at Sea
DOF	=	Degrees of Freedom
DW	=	Dynamic Window
GNC	=	Guidance, Navigation and Control
GNSS	=	Global Navigation Satellite System
LOS	=	Line of Sight
MPC	=	Model Predictive Control
NED	=	North-East-Down
PLOS	=	Combined Pure Pursuit and LOS
PRM	=	Probabilistic Roadmap
RRT	=	Rapidly-Exploring Random Tree
SQP	=	Sequential Quadratic Programming
USV	=	Unmanned Surface Vehicle
VHF	=	Vector Field Histogram
VO	=	Velocity Obstacles

Chapter 1

Introduction

This introduction section will present the relevant background, motivation and main contributions that have been done so far, in order to give a general overview of frameworks and techniques that are studied and implemented in this thesis.

1.1 Motivation and Background

Over the past few decades, the field of autonomous vehicles has witnessed an ever increasing interest in motion planning, especially for Autonomous Surface Vehicle (ASV), also known as Unmanned Surface Vehicle (USV), which can operate on the surface of water without human intervention. ASV can be used to perform a variety of missions without a crew in hazardous environment, which declines the risk of causalities to a great extent. Some research in collision avoidance and other operation has been conducted to ensure safe and reliable control for ASV. Relevant technologies have applied ASV as a viable platform, to marine operations and the application areas where they contribute (Manley, 2008).

Compared to other vehicles, USV has the advantages of lower operational cost, more reliable safety and wider scope of operation. The upsides above motivate the development in path planning and collision avoidance (COLAV) for ASV to guarantee considerable reliability and manoeuvrability. Path planning algorithm generates geometric paths, from the initial point to the target, while passing through a couple of waypoints based on the collision-free goal. However, it is barely possible for ASV to precisely follow a preplanned path due to limited information about the environment, on-board sensors with limited range, speed and acceleration constraints, and disturbance in vehicle state and sensor data (Goerzen et al., 2010).

Figure 1.1: ASV Global C-Worker for offshore operations (ASV, 2019)

When it comes to motion planning for Autonomous Surface Vehicles (ASV), collision avoidance (COLAV) is an essential issue that raises great concerns. The most popular techniques in the COLAV field are generally divided into two types, deliberate and reactive methods. ASV is required to make real-time response while navigating in the unknown and cluttered dynamic environment. Hence, a reactive COLAV method that receives data and information of the immediate environment from sensors, significant contributes to local collision avoidance in the presence of both static and dynamic obstacles. Reactive COLAV methods are widely used due to low demand for computing capabilities, while it still suffers the risk of being trapped in local minima.

Due to the limitations of reactive method, for instance, high sensitivity to local minima, it is not adequate to solely employ reactive COLAV guiding an ASV to goal. Deliberate COLAV methods that rely on the information of the complete environment, become necessary to generate a global path used as guidance for reactive method. Deliberate method is normally referred to as path planning, aiming to find a route from start to goal point and is less likely to fail when encountering local minima. Nevertheless, the main drawback of deliberate method is the demand for long computing time, which makes it no longer applicable to rapidly changing dynamic environment. As a consequence, it is guaranteed to yield a collision-free trajectory towards goal by adopting a hybrid COLAV approach based on the integration of deliberate and reactive method.

1.2 Previous Work

Over the past years, a great variety of methods regarding collision avoidance (COLAV) have been developed. Deliberate COLAV methods, including graph search algorithms

such as A* (Hart et al., 1968) and D* (Stentz et al., 1995), rapidly-exploring random trees (RRT) (LaValle, 1998) and probabilistic roadmap (PRM) (Kavraki et al., 1994) are widely used in the field of path planning. In addition to the above methods, spline-based constrained optimization is also employed to generate global paths. (Hassani and Lande, 2018) presents an effective path planning technique that incorporates the dynamics of USV based on Bézier curves and differential flatness.

As for reactive COLAV methods, velocity obstacles (VO) method (Fiorini and Shiller, 1998) is one of them, intended for motion planning to avoid static and moving obstacles in the velocity space. (Ge and Cui, 2002) proposed a new approach based on potential field for motion planning of robots in a dynamic environment with moving goals and obstacles. Additionally, dynamic window algorithm is one of the existing reactive COLAV approach, originally designed for robots with first order nonholonomic constraints (Fox et al., 1997). A modified DW algorithm presented in (Eriksen et al., 2016), is adapted and tested for autonomous underwater vehicles (AUV) with second-order nonholonomic constraints.

Extensive research in hybrid COLAV method has also been carried out to solve the existing issues in both reactive and deliberate approaches. (Seder and Petrovic, 2007) proposes an improved dynamic window algorithm incorporated with a focused D* search algorithm, such that the vehicle is less likely to be trapped in local minima. Furthermore, (Serigstad et al., 2018) introduces a hybrid dynamic window approach, functions as an interface combined with deliberate path planning approach which generates time parameterized trajectories, yielding the result that vehicle is able to avoid local minima. In (Lopes et al., 2016), A* algorithm is used as a path planner while a modified dynamic window (DW) algorithm is implemented enabling the vehicle to avoid dynamic obstacles.

1.3 Contributions

- Present a review of existing COLAV methods and the comparison of them, including both reactive and deliberate COLAV techniques.
- Based on previous work (Hassani and Lande, 2018), path planning algorithm is formulated using Bézier curves within the framework of optimization, which is employed as a deliberate COLAV method in this thesis.
- Differential flatness of the mathematical model has been demonstrated, taking the dynamics of ASV into account, such that smooth and continuous trajectory can be followed by the underactuated surface vessel.
- In order to incorporate dynamic obstacles, a modified dynamic window algorithm is proposed, enabling the vehicle to perform real-time actions and avoid the collision with moving obstacles.
- To combine global pre-defined paths generated by Bézier curves with dynamic window algorithm, a combined pure pursuit and LOS path tracking algorithm is intro-

duced, which contributes to the extensive development of interface between deliberate and reactive COLAV method.

- Last but not least, numerical simulations are carried out to test and demonstrate the performance of this hybrid method.

1.4 Outline

This thesis is organized as follows:

Chapter 2: This chapter presents the vehicle modeling, including kinematics and kinetics. Besides, chapter 2 also illustrates the guidance, navigation and control (GNC) system and elaborate different modules of GNC system. In particular, the principle of PLOS steering law is introduced.

Chapter 3: A global path planning based on Bézier curve and constrained optimization is presented. The basic principle and property of Bézier Curves are described in detail in this chapter. Moreover, differential flatness of the mathematical model for the surface vessel has been demonstrated with a simplified model.

Chapter 4: Existing motion planning algorithms applied in different situations are introduced and comprehensively compared. Dynamic Window algorithm exploited as reactive COLAV is highlighted in this chapter.

Chapter 5: In this chapter, a hybrid COLAV architecture is explained and the interface between reactive and deliberate COLAV method is also presented.

Chapter 6: This chapter presents a series of numerical simulation results concerning path planning and collision avoidance, which shows the performance and robustness of proposed hybrid COLAV method.

Chapter 7: Chapter 7 presents the conclusion of the work has been currently done and introduces the future work.

Chapter 2

Theoretical Backgrounds

2.1 Vessel Dynamics and Modeling

In this section, the investigation of vessel dynamics mainly focus on kinematics and kinetics, involving geometrical aspects of motion and relevant forces that generates force.

2.1.1 Kinematics

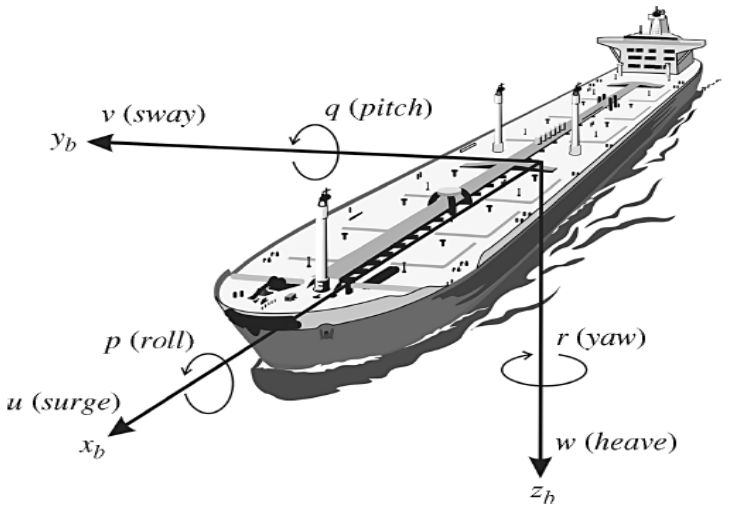

Figure 2.1: The 6DOF velocities u, v, w, p, q and r in the body-fixed reference frame (Fossen, 2011)

Geographic Reference Frames

NED: The NED coordinate system $n = (x_n, y_n, z_n)$ with origin o_n is referred to as North-East-Down system, relative to the Earth's reference. It is usually defined as the tangent plane on the surface of the Earth moving with the craft, but with axes pointing in different directions than the body-fixed axes of the vessel. For this system the x axis points towards true North, the y axis points towards East while the z axis points downwards normal to the Earth's surface. The location of n relative to e is determined by using two angles in terms of longitude and latitude, respectively.

BODY: The body-fixed reference frame $b = (x_b, y_b, z_b)$ with origin o_b is a moving coordinate frame that is fixed to the vessel. The position and orientation of the vessel are described relative to the inertial reference frame, while the linear and angular velocities of the craft should be expressed in the body-fixed coordinate system. The origin o_b is usually chosen to coincide with a point midships in the water line. In this frame the x_b axis is fixed in the vessel forward direction, y_b axis is fixed in the vessel starboard side, pointing towards east, and z_b is fixed in the vertical direction, directed from top to bottom. The centre of gravity is then located at $(x_G, 0, z_G)$ in body coordinates. In particular, this frame is usually used to calculate the motion and the loads acting on the vessel.

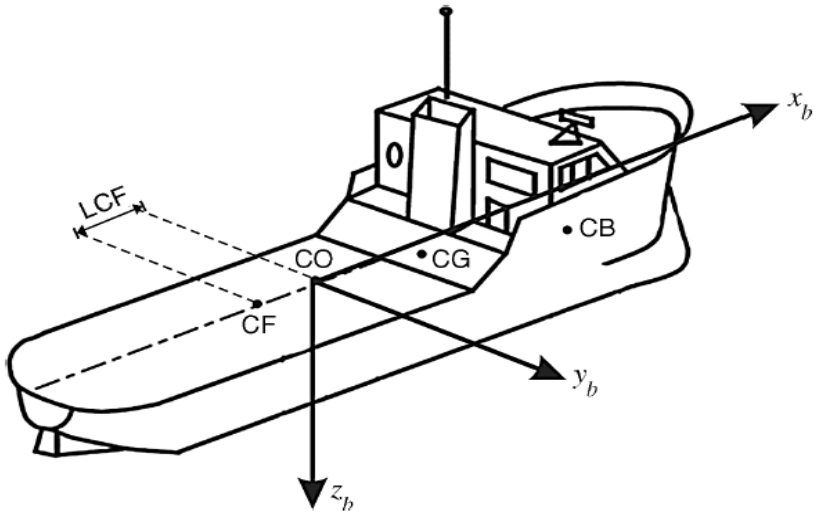

Figure 2.2: Body-fixed reference frame points (Fossen, 2011)

The following reference points are defined with respect to CO:
CG – Center of Gravity
CB – Center of Buoyancy
CF – Center of Flotation

Transformations between NED and BODY

The rotation matrix R between two frames a and b is denoted as R_a^b and it is useful when deriving kinematic equations of motion for a surface vessel. The linear velocity transformation the body-fixed frame to NED can be presented as,

$$R_b^n(\Theta_{nb}) = R_{z,\psi} R_{y,\theta} R_{x,\phi} \tag{2.1.1}$$

where $R_{z,\psi}$, $R_{y,\theta}$ and $R_{x,\phi}$ are rotation matrices about axis z_b, y_b, x_b, respectively.

Since in the case of horizontal motion, only surge, sway and yaw are taken into account such that rotation matrix about z axis is merely involved. Therefore, the transformation from BODY to NED can be implemented by the rotation matrix $R_z(\psi)$.

$$R(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1.2}$$

3DOF Kinematics

DOF, referred to as degree of freedom, consists of independent displacements and rotations, which are exploited to describe the displaced position and orientation of the vessel. A general body that can move freely in the 3D space has maximum 6 DOFs including three translational and three rotational components (Sørensen, 2012). In many applications, only the horizontal motions are taken into account, for instance, ships are often described in the horizontal plane only considering the motions, surge, sway and yaw (n = 3DOFs). And the 3DOF kinematics can be expressed as follows,

$$\dot{\eta} = R(\psi)v, \qquad R^{-1}(\psi) = R^T(\psi) \tag{2.1.3}$$

where the state vectors reduce to $\eta = [x, y, z]^T$ and $v = [u, v, r]^T$

2.1.2 Vessel Dynamics

In this thesis, a simplified mathematical model and control plant model is nevertheless sufficient to describe the main physical characteristics of the dynamic system (Sørensen, 2012). A nonlinear low-frequency horizontal plane model for maneuvering in surge, sway and yaw about zero vessel velocity is based on rigid-body kinetics, and can be presented as follows:

$$\begin{aligned} \dot{\eta} &= J(\eta)v \\ M\dot{v} + C(v)v + D(v)v + g(\eta) &= \tau \end{aligned} \tag{2.1.4}$$

where mass matrix, Coriolis and centripetal terms can be represented as

$$\begin{aligned} M &= M_A + M_{RB} \\ C(v) &= C_A(v) + C_{RB}(v) \end{aligned} \tag{2.1.5}$$

$D(v)$ is the damping matrix, and $J(\eta)$ is the kinematic transformation matrix. In the right-hand side of the equation, control forces and moments are denoted as τ. Here, we assume

that the mass distribution is homogeneous and xz-plane is symmetric, such that surge motion is decoupled from sway and yaw. The inertia matrix M consists of rigid body and added mass, and has the properties $M = M^T > 0$ and $\dot{M} = 0$. Furthermore, it is notable that the Coriolis and Centripetal matrix is characteristic of the skew symmetric matrix, that is $C(v) = -C^T(v)$.

Since the horizontal motion for the surface vessel can be described by the components in surge, sway and yaw, and the state vectors including position and velocity can then be reduced to $\eta = [x, y, \psi]^T$ and $v = [u, v, r]^T$, respectively, which implies that the dynamics associated with the motion in heave, roll and pitch are neglected. Therefore, the kinematic transformation matrix $J(\eta)$ is then identical to the rotation matrix in terms of 3DOF.

$$J(\eta) = R(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1.6}$$

Let the origin of body-fixed frame be fixed in the center line of the vessel at the point CO, the mass matrix and Coriolis term associated with the rigid-body kinetics then can be represented as,

$$M_{RB} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & mx_g \\ 0 & mx_g & I_z \end{bmatrix} \qquad C_{RB}(v) = \begin{bmatrix} 0 & 0 & -m(x_g r + v) \\ 0 & 0 & mu \\ m(x_g r + v) & -mu & 0 \end{bmatrix} \tag{2.1.7}$$

while mass matrix, Coriolis and Centripetal term associated with the added mass are then given as,

$$M_A = \begin{bmatrix} -X_{\dot{u}} & 0 & 0 \\ 0 & -Y_{\dot{v}} & -Y_{\dot{r}} \\ 0 & -Y_{\dot{r}} & -N_{\dot{r}} \end{bmatrix} \qquad C_A(v) = \begin{bmatrix} 0 & 0 & Y_{\dot{v}} v + Y_{\dot{r}} r \\ 0 & 0 & -X_{\dot{u}} u \\ -Y_{\dot{v}} v - Y_{\dot{r}} r & X_{\dot{u}} u & 0 \end{bmatrix} \tag{2.1.8}$$

The inertial matrix, Coriolis and Centripetal matrix then can be expressed as,

$$M_A = \begin{bmatrix} m - X_{\dot{u}} & 0 & 0 \\ 0 & m - Y_{\dot{v}} & mx_g - Y_{\dot{r}} \\ 0 & mx_g - Y_{\dot{r}} & I_z - N_{\dot{r}} \end{bmatrix} = \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & m_{23} \\ 0 & m_{23} & m_{33} \end{bmatrix} \tag{2.1.9}$$

$$C(v) = \begin{bmatrix} 0 & 0 & -m_{23} r - m_{22} v \\ 0 & 0 & m_{11} u \\ m_{23} r + m_{22} v & -m_{11} u & 0 \end{bmatrix} \tag{2.1.10}$$

The damping matrix $D(v)$ is composed of linear damping and nonlinear damping, given as

$$D(v) = D + D_n(v) \tag{2.1.11}$$

The linear damping matrix D in this expression is important for low-speed maneuvering and station keeping and can be represented as

$$D = \begin{bmatrix} -X_u & 0 & 0 \\ 0 & -Y_v & -Y_r \\ 0 & -N_v & -N_r \end{bmatrix} \tag{2.1.12}$$

2.2 Motion Control System

The motion control systems consist of three subsystems, namely guidance, navigation and control (GNC) systems. Guidance system refers to the determination of desired path or trajectory from the initial position to the goal, and the desired changes in reference position, velocity and acceleration of the surface vessel. The navigation system functions to determine the vessel's position, velocity and acceleration using global navigation satellite system (GNSS) and sensors such as accelerometers and gyroscopes. Given information from the navigation system and reference trajectory from the guidance system, the control system is used to determine the necessary control forces and moments to follow the reference trajectory with small error.

The preliminary step for designing a motion control system is to figure out what the control objective is. Generally, the objective is defined as setpoint regulation, trajectory-tracking control or path-following control (Fossen, 2011). As the most basic guidance system, the setpoint regulation uses a constant input as guidance with a controller, referred to as regulator. In trajectory-tracking control, the guidance system is designed to follow a desired trajectory, taking temporal constraints into consideration. By contrast to trajectory-tracking control, path-following control function as tracking a time-invariant predefined path. In addition, the path planning system is essential to the guidance system, since the output of the path planner functions as input to the guidance system. The path planner is used to design and generate a path, which is expected to be safe and feasible if followed without deviations. Typically, the output of the path planner are a set of waypoints or desired path.

2.2.1 Configuration Space in Motion Planning

In motion planning, configuration is a key concept, defined as a complete specification of the position of every point in the system (Spong et al., 2006). Thus the space of all configurations is defined as configuration space, also denoted as $\mathcal{C}$-space.

For a 3DOF surface vehicle considered in this thesis, the configuration consists of location and heading angle of the vehicle, denoted as $\eta = (x, y, \psi)^T$. And a 3-dimensional configuration space can be expressed as $\mathcal{C} = \mathbb{R}^2 \times SO(2)$. Besides, obstacles in the configuration space can be defined as a set, $\mathcal{O} = \mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_n$, where $\mathcal{O}_i$ denotes the i-th obstacle in $\mathcal{C}$. In $\mathcal{C}$-space, the set of configurations where the vehicle might collide with an obstacle is defined as obstacle configuration space, also referred to as forbidden space, and can be expressed as

$$\mathcal{C}_{forb} = \{\eta \in \mathcal{C} | \mathcal{A}(\eta) \cap \mathcal{O} \neq \emptyset\} \tag{2.2.1}$$

Hence, the remaining set of $\mathcal{C}$ is collision-free configuration set, called free configuration space, denoted as

$$\mathcal{C}_{free} = \mathcal{C} \backslash \mathcal{C}_{forb} = \{\eta \in \mathcal{C} | \mathcal{A}(\eta) \cap \mathcal{O} = \emptyset\} \tag{2.2.2}$$

For a 3D ASV, three-dimensional configuration space can be simplified as two-dimensional space by approximating the vehicle as a circle or disc with a radius representing the longest distance from the center point to any other point of the vehicle, shown in Figure 2.3. As a

consequence, the new two-dimensional configuration space is independent of heading ψ, and the configuration of the vehicle then becomes $p = (x, y)$.

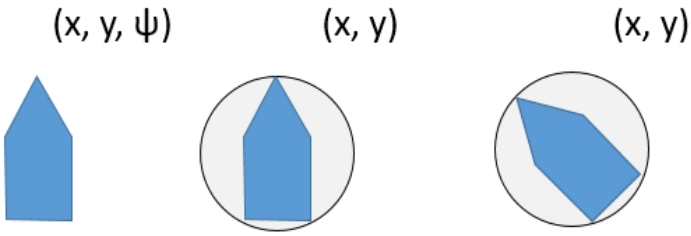

Figure 2.3: Three-dimensional simplified as two-dimensional space

2.2.2 Path Planning Algorithm

Over the past few decades, path planning or trajectory planning has been a hot issue in the field of automation. Path planning algorithm is intended to generate geometrical path from the initial point to goal, connecting a sequence of waypoints without considering temporal assignments. Considerable research has been done on this topic, and overview of existing path planning algorithm is presented in (LaValle, 2006). Path planning methods are usually divided into two categories, global and local path planning, also referred to as deliberate and reactive methods.

Deliberate Methods

Given complete information of the environment, deliberate method is designed to find a route from the start point to goal. Majority of deliberate path planning algorithms are complete, indicating that the algorithm is always able to find the path if it exists. Global methods including graph search algorithms such as A* (Hart et al., 1968) and D* (Stentz et al., 1995), rapidly-exploring random trees (RRT) (LaValle, 1998) and probabilistic roadmap (PRM) (Kavraki et al., 1994) are widely used. Nevertheless, the main drawback of deliberate method is the demand for long computing time, which makes it no longer applicable to rapidly changing dynamic environment. Compared to local methods, they are much slower since the global methods are required to calculate all the subsequent motions until the vehicle reaches the goal, meaning that problems may occur when the vehicle encounters unexpected situations.

Reactive Methods

Based on data and information of immediate environment from the sensors, reactive approaches are able to make real-time response that requires low computation capacities.

Due to the property that only a subset of configuration space is considered, it behaves in a ”greedy” way to reach the goal, which results in great possibility of being trapped in local minima, as shown in Figure 2.4. Popular reactive method including Vector Field Histogram (VFH) (Borenstein and Koren, 1991), Potential field method (Khatib, 1986), and velocity space based methods, like Velocity Obstacles (VO) (Fiorini and Shiller, 1998) and Dynamic Window (DW) (Fox et al., 1997) will be elaborated in later section.

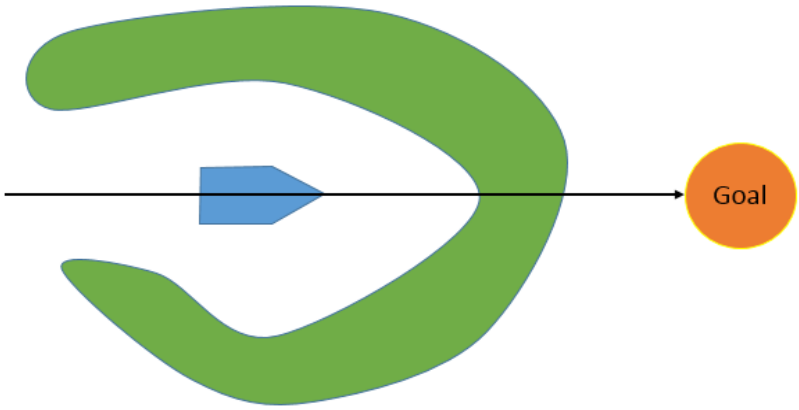

Figure 2.4: Vehicle trapped in local minima

2.2.3 Path Tracking Algorithm

Path tracking scheme is required to drive a vehicle to follow a pre-determined path generated by path planner. Popular path tracking algorithms, including pure pursuit path tracking and Line of Sight (LOS) method are both able to steer the vehicle to track a desired path. The main difference between these two methods is that the pure pursuit guidance law directly drives the vehicle towards the next waypoint W_{i+1}, while LOS steers the vehicle onto the path towards the waypoint, illustrated in Figure 2.7. To absorb the advantages of both methods, a combined pure pursuit LOS guidance law (Kothari et al., 2014) for ASV is adopted in this thesis.

Line of Sight

Line of sight (LOS) guidance is a widely-used method that generates heading commands steering a vehicle to the desired goal position, which consists of a reference point, an interceptor and a target. Both reference point and target can be viewed as wayponits, denoted as (x_k, y_k) and (x_{k+1}, y_{k+1}). LOS guidance law is usually exploited to address practical path-following issues of marine vehicles due to its simplicity and intuitiveness: it drives a vessel towards the given point, a constant distance ahead of the vessel along the desired

paths (Fossen, 2011).

Since only horizontal motion is considered for the surface vessel, the speed of the vessel can be represented as,

$$U(t) := ||v(t)|| = \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2} \geq 0 \tag{2.2.3}$$

The course angle, denoted as $\chi(t)$, can be defined as,

$$\chi(t) := atan2(\dot{y}(t), \dot{x}(t)) \in \mathbb{S} := [-\pi, \pi] \tag{2.2.4}$$

where the $atan(y, x)$ is the four-quadrant inverse tangent of y and x, $arctan(y, x) \in [-\pi/2, \pi/2]$. Considering a straight-line connecting two waypoints, $p_k = (x_k, y_k)$ and $p_{k+1} = (x_{k+1}, y_{k+1})$, the rotation angle of the straight line with respect to NED reference frame, denoted as α_k can be described as,

$$\alpha_k := atan2(y_{k+1} - y_k, x_{k+1} - x_k) \in \mathbb{S} := [-\pi, \pi] \tag{2.2.5}$$

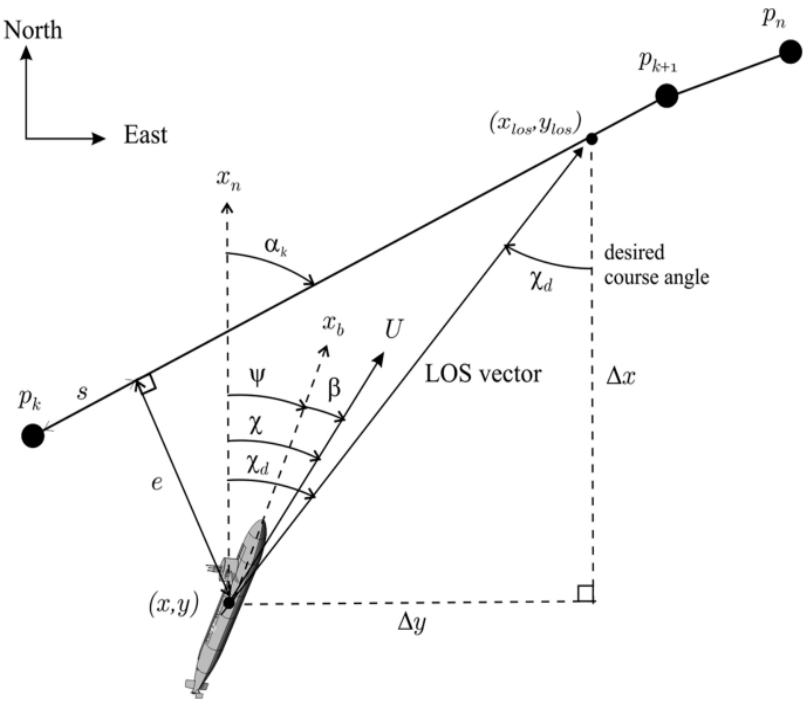

Figure 2.5: LOS guidance with desired course angle points toward the intersection point (Fossen, 2011)

For lookahead-based steering, the course angle consists of two parts and can be expressed as,

$$\chi_d(e) = \chi_p + \chi_r(e) \tag{2.2.6}$$

where e is the cross-track error, normal to the path line between two waypoints, illustrated in Fig. Besides, the path-tangential angle and velocity-path relative angle can be given as,

$$\chi_p = \alpha_k \tag{2.2.7}$$

and

$$\chi_r(e) := arctan(\frac{-e}{\Delta}) \tag{2.2.8}$$

The relative angle of the velocity-path ensures that the velocity is directed toward a point on the path located a look-ahead distance $\Delta(t) > 0$ ahead of the direct projection of $p^n(t)$ on to the path. The look-ahead distance can be calculated as follows, illustrated in Figure 2.6.

$$\Delta(t) = \sqrt{R^2 - e(t)^2} \tag{2.2.9}$$

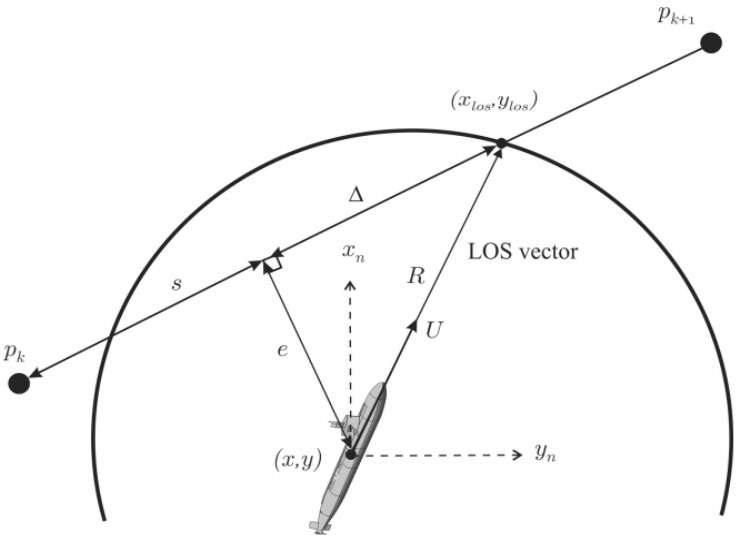

Figure 2.6: Circle of acceptance with constant radius R (Fossen, 2011)

For path-following controllers, the transformation from the desired course angle χ_d to the desired heading angle command ψ_d can be expressed as,

$$\psi_d = \chi_d - \beta \tag{2.2.10}$$

where β is the sideslip angle computed by,

$$\beta = arcsin(\frac{v}{U}) \tag{2.2.11}$$

The desired heading angle ψ_d then can be used as the reference in a heading controller.

Pure Pursuit Path Tracking

Pure pursuit path tracking algorithm (Coulter, 1992) has been widely used as a steering controller for autonomous vehicles. (Yamasaki et al., 2009) proposes a robust path-tracking method for UAV based on pure pursuit steering law. (Rankin et al., 1998) presents a review and evaluation of PID, pure pursuit, and weighted steering controller for an autonomous land vehicle. The major objective of this method is to calculate curvatures enabling the vehicle to chase a moving target point that is some distance ahead of it on the pre-planned path. The chord length of the arc represents the look-ahead distance joining current position and goal point, and it's used when search for the next target point.

Combined Pure Pursuit and LOS (PLOS)

A combined pure pursuit and line-of-sight (PLOS) path tracking algorithm is employed to achieve two objectives. That is, LOS accounts for the position error d regarding desired path where d also referred to as cross-track error, while pure pursuit is used to compensate for the heading error $(\theta_d - \psi)$ towards next waypoint.

Navigation model for ASV is defined as follows

$$\dot{x} = v \cdot cos\psi \tag{2.2.12a}$$
$$\dot{y} = v \cdot sin\psi \tag{2.2.12b}$$
$$\dot{\psi} = u \tag{2.2.12c}$$

where v is the surge speed of ASV, ψ represents the heading angle while u denotes the control input.

The initial position of vehicle and waypoints ca be represented as

$$p = (x, y), \quad W_i = (x_i, y_i), \quad W_{i+1} = (x_{i+1}, y_{i+1}) \tag{2.2.13}$$

The position error d then can be calculated as

$$d = R \cdot sin(\theta - \theta_u) \tag{2.2.14a}$$
$$R = W_i - p = \sqrt{(x_i - x)^2 + (y_i - y)^2} \tag{2.2.14b}$$
$$\theta = atan2(y_{i+1} - y_i, x_{i+1} - x_i) \tag{2.2.14c}$$
$$\theta_u = atan2(y - y_i, x - x_i) \tag{2.2.14d}$$

where R is the distance from the waypoint W_i to the vehicle, is the LOS angle formed by waypoints W_i and W_{i+1}, while θ_u is the angle formed by vehicle and waypoint W_i.

Apart from minimizing the heading error $(\theta_d - \psi)$, the PLOS algorithm is also aimed at shrinking the cross-track error d, enabling the vehicle to converge to the desired path, as depicted in Figure 2.7. As a consequence, a weighted sum of position and heading error is

derived as the guidance law,

$$\psi_d = k_1 \cdot (\theta_d - \psi) + k_2 \cdot d \tag{2.2.15a}$$

$$\theta_d = atan2(y_{i+1} - y, x_{i+1} - x) \tag{2.2.15b}$$

where $k_1 > 0$ and $k_2 > 0$ are the gains, θ_d is the desired heading angle towards the next waypoint W_{i+1}, ψ is the heading angle of the vehicle, and d is the cross-track error, the distance vehicle deviates from the desired path.

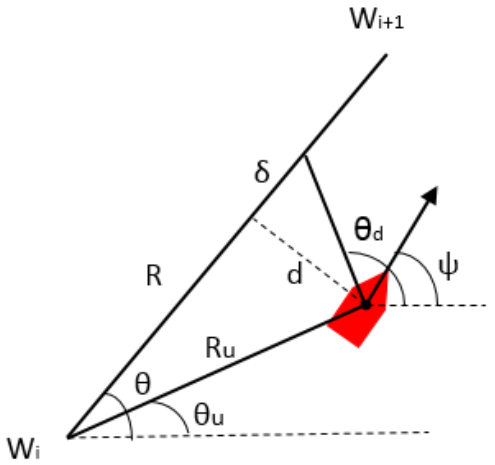

Figure 2.7: PLOS path tracking algorithm regarding straight line

Chapter 3

Global Path Planning

In this chapter, a global path planning algorithm based on previous work (Hassani and Lande, 2018), is introduced. Using Bézier curves as a basis to generate a set of paths joining the initial and final position while avoiding the collision with static obstacles, the problem is formulated as a optimization issue. Property of differential flatness is employed to incorporate the dynamics of the vehicle by assigning corresponding cost to each path. Furthermore, the optimal path is selected by minimizing the objective function while satisfying the constraints.

3.1 Bézier Curves

Bézier curve is a polynomial parametric curve widely used in the field of animation, computer graphics and path planning. The history of Bézier curve can be tracked back to 1912, a Russian mathematician Sergei Natanovich Bernstein proposed the concept of Bernstein polynomials while it was not applied to graphics at that moment. Using Bernstein polynomials as mathematical basis, a French engineer, Pierre Bézier finally made Bézier curves well known to the world in 1960s. The property of Bézier curve has motivated a variety of research on path planning exploiting Bézier spline to generate smooth path. (Choi et al., 2008) proposes two path planning methods based on Bézier curves for autonomous vehicles with waypoints and corridor constraints, in which cubic Bezier curve segments are exploited to generate smooth path. Besides, a Bézier curve-based cooperative collision-avoidance method for multiple nonholonomic robots is proposed in (Škrjanc and Klančar, 2010), where optimal path passing through start point and goal point, is obtained by minimizing the cost function, considering path lengths between the robots.

As an effective tool in graphics, Bézier curves are usually exploited to model smooth curves, and Bézier spline can be generated by lining up multiple Bézier curves. A Bézier curve is defined by a set of control points, from P_0 to P_n, where P_0 and P_n are the endpoints of the curve, and n represents the degree of curve. Thus, the number of control points for a curve of degree n is n+1, while the intermediate control points are not perfectly

lying on the curve. By definition, a Bézier curve of degree n can be represented as

$$p(t) = \sum_{i=0}^{n} B_i^n(t) p_i \qquad t \in [0,1] \tag{3.1.1}$$

where t represents a normalized time variable, and p_i denotes the control points together with the basis function $B_i^n(t)$ determine the shape of curve. The basis function, also referred to as Bernstein polynomial, can be explicitly expressed as

$$B_i^n = \binom{n}{i} (1-t)^{n-i} t^i, \qquad i = 0, 1, 2, \ldots, n \tag{3.1.2}$$

where $\binom{n}{i}$ is the binomial coefficient, given as

$$\binom{n}{i} = \frac{n!}{i!(n-i)!} \tag{3.1.3}$$

For n = 5, the corresponding Bézier curve can be expressed as

$$\begin{aligned} p(t) =& (1-t)^5 P_0 + 5t(1-t)^4 P_1 + 10t^2(1-t)^3 P_2 \\ &+ 10t^3(1-t)^2 P_3 + 5t^4(1-t) P_4 + t^5 P_5 \quad t \in [0,1] \end{aligned} \tag{3.1.4}$$

To simplify the computation of derivatives, it might be sensible to take advantage of matrix representation, since parameters, coefficients and control points can be expressed separately in that way.

$$p_p(t) = \begin{bmatrix} 1 & t & \ldots & t^n \end{bmatrix} \begin{bmatrix} b_{0,0} & 0 & \ldots & 0 \\ b_{1,0} & b_{1,1} & \ldots & 0 \\ \vdots & \vdots & \ddots & 0 \\ b_{n,0} & b_{n,1} & \ldots & b_{n,n} \end{bmatrix} \begin{bmatrix} p_0 \\ p1 \\ \vdots \\ p_n \end{bmatrix} \tag{3.1.5}$$

where coefficients can be denoted as

$$b_{i,j} = (-1)^{i-j} \binom{n}{i} \binom{i}{j} \tag{3.1.6}$$

The polygon is formed by sequentially connecting control points with straight lines, starting from P_0 and ending at P_n. This polygon is then called the Bézier polygon or control polygon. And the Bézier curve is contained in the convex hull of the Bézier polygon. Taking Bézier curve of degree 5 as an example, control points and Bernstein polynomials are given as

$$\begin{aligned} p_0 = [-1,-1]^T, \quad p_1 = [0,2]^T, \quad p_2 = [3,4]^T, \\ p_3 = [5,5]^T, \quad p_4 = [10,3]^T, \quad p_5 = [12,-1]^T \end{aligned} \tag{3.1.7}$$

and

$$\begin{aligned} B_0^5(t) = (1-t)^5, \quad B_1^5(t) = 5(1-t)^4 t, \quad B_2^5(t) = 10(1-t)^3 t^2 \\ B_3^5(t) = 10(1-t)^2 t^3, \quad B_4^5(t) = 5(1-t)t^4, \quad B_5^5(t) = t^5 \end{aligned} \tag{3.1.8}$$

As shown in Figure 3.1, the Bézier curve of degree 5 is completely contained in the control polygon, connected by control points.

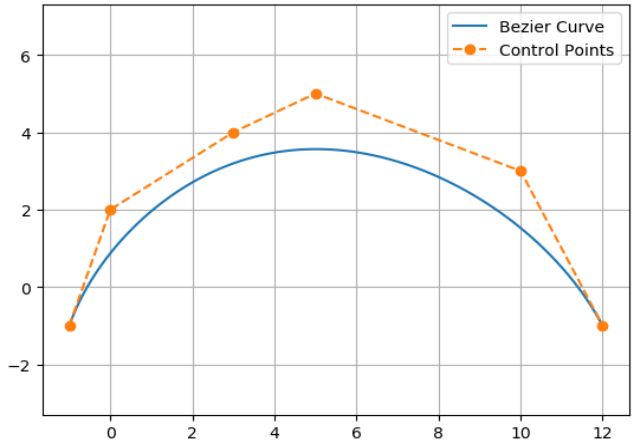

Figure 3.1: Bézier curve of degree 5 with control points and Bézier polygon

The derivative for a Bézier curve of degree n, is definitely the Bézier curve of degree n-1. As the derivative for a Bézier curve is independent of control points and parameter t, the derivative can be obtained based on the derivative of the Bernstein polynomial, which can be defined as

$$B_i^n(t)' = n(B_{i-1}^{n-1}(t) - B_i^{n-1}(t)), \qquad t \in [0,1] \tag{3.1.9}$$

Furthermore, with Eq. (3.1.9) as the basis, the derivative of Bézier curve can be expressed as

$$p'(t) = n\sum_{i=0}^{n-1} B_i^{n-1}(t)(p_{i+1} - p_i), \qquad t \in [0,1] \tag{3.1.10}$$

To determine the k-th derivative for a Bézier curve, we need to find the control points of all the derivatives before k, and a general expression can be derived as

$$q_i^k = q_{i+1}^{k-1} - q_i^{k+1}, \qquad i \in 0, 1, \ldots, n-k \tag{3.1.11}$$

where n represents the degree of curve, k denotes the derivative, and q_i^k is the k-th derivative of control points, e.g, $q_i = p_{i+1} - p_i$. To be specific, the first and second derivatives at the endpoints of Bézier curve of degree 5 can be expressed as

$$\begin{aligned} p'(0) &= 5(p_1 - p_0) & p'(1) &= 5(p_5 - p_4) \\ p''(0) &= 20(p_2 - 2p_1 + p_0) & p''(1) &= 20(p_5 - 2p_4 + p_3) \end{aligned} \tag{3.1.12}$$

where $p'(0)$ and $p'(1)$ represent the first derivative at start point and end point, respectively, while $p''(0)$ and $p''(1)$ denote the second derivatives at endpoints.

By mathematical definition, curvature is a measure for the rate of change of a curve's direction in terms of distance along the the curve. Given a parameterized curve with respect to arc length $p_p(s)$, the curvature of curve can be defined as

$$\kappa(s) = |\frac{\boldsymbol{T}s}{ds}| \tag{3.1.13}$$

where tangent T is referred to as T = $\frac{dp_p}{ds}$, and arc length $p_p(s)$ can be derived as

$$s = \int_{t_0}^{t_1} \mid p'_p(\tau)d\tau \mid = \int_{t_0}^{t_1} \sqrt{x'_p(\tau)^2 + y'_p(\tau)^2}d\tau \tag{3.1.14}$$

Given the position of control point p(t) = (x(t), y(t)), the curvature of Bézier curve can be calculated as

$$\kappa(t) = \frac{\mid p'_p(t) \times p''_p(t) \mid}{\mid p'_p(t) \mid^3} = \frac{\mid x'_p(t)y''_p(t) - x''_p(t)y'_p(t) \mid}{(x'_p(t)^2 + y'_p(t)^2)^{\frac{3}{2}}} \tag{3.1.15}$$

Furthermore, the radius of curvature is defined to be the reciprocal of the curvature that mostly approximate to the curve at the given point, denoted as

$$R(s) = \frac{1}{\kappa(s)} \tag{3.1.16}$$

3.2 Differential Flatness

Differential flat system is characteristic of the competence that it is able to generate effective control policies for nonlinear systems (Murray et al., 1995). A system is said to be differential flat if all involving states and inputs can be explicitly expressed by a fictitious flat output vector and the derivatives in terms of the flat outputs. Specifically, if the system has states $x \in R^n$ and inputs $u \in R^m$, the system is then differential flat if we can find the flat output y in the form

$$y = y(x, u, \dot{u}, \cdots, u^{(p)}) \tag{3.2.1}$$

such that

$$\begin{aligned} x &= x(y, \dot{y}, \cdots, y^{(q)}) \\ u &= u(y, \dot{y}, \cdots, y^{(q)}) \end{aligned} \tag{3.2.2}$$

Differential flat systems are essential in the case of that explicit trajectory generation is required. For the sake of simplicity, trajectories can be planned in output space and then be mapped to appropriate inputs, due to the fact that the behavior of flat outputs are the key factors to determine a flat system. The flatness property of the system implies that the trajectory planning problem can be simplified as algebra in theory, and computationally attractive algorithms in practice. (Murray et al., 1995) presents initial results on the characterization of differential flatness for mechanical systems and shows how symmetries and

inertial properties relate to differential flatness.

In this section, we show that vessel dynamics with three flat output variables [x, y, ψ], is differential flatness. That is, all relevant states $[x, y, \psi, u, v, r]$ and inputs τ_u and τ_r can be written as algebraic functions of the three flat outputs and their derivatives. In this report, the main purpose is to derive a path planning approach for an underactuated surface vessel, in which smooth and continuous trajectory in the flat output space can be followed by the underactuated surface vessel.

In order to build a flat mathematical model for the surface vessel, some simplifications and assumptions must be proposed to ensure the differential flatness of the model:
1) The ship is fore/aft symmetric, which means all the off-diagonal elements in inertia matrix M and damping matrix D would be eliminated.
2) Since the ship motion is assumed to be low-speed, nonlinear damping is neglected in this report, that is, only linear damping matrix D is considered in (Lande, 2018).

Based on the above assumptions, the simplified mathematical model can be derived as follows,

$$\begin{aligned} \dot{\eta} &= R(\psi)v \\ M\dot{v} + C(v)v + Dv &= \tau \end{aligned} \tag{3.2.3}$$

where $\tau = [\tau_1, 0, \tau_3]$, $M = diag\{m_{11}, m_{22}, m_{33}\}$, and $D = diag\{d_{11}, d_{22}, d_{33}\}$

$$C(v) = \begin{bmatrix} 0 & 0 & -m_{22}v \\ 0 & 0 & m_{11}u \\ m_{22}v & -m_{11}u & 0 \end{bmatrix}, \qquad R(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Inserting the matrix into the mathematical model, we will obtain a set of motion equations on a more specific form.

$$\dot{x} = ucos(\psi) - vsin(\psi) \tag{3.2.4a}$$
$$\dot{y} = usin(\psi) + vcos(\psi) \tag{3.2.4b}$$
$$\dot{\psi} = r \tag{3.2.4c}$$
$$\dot{u} = vr - \beta_1 u + \tau_u \tag{3.2.4d}$$
$$\dot{v} = -ur - \beta_2 v \tag{3.2.4e}$$
$$\dot{r} = -\beta_3 r + \tau_r \tag{3.2.4f}$$

where $m_{11} = m_{22}$, $\beta_1 = \frac{d_{11}}{m_{11}}$, $\beta_2 = \frac{d_{22}}{m_{22}}$, $\beta_3 = \frac{d_{33}}{m_{33}}$, $\tau_u = \frac{\tau_1}{m_{11}}$, $\tau_r = \frac{\tau_3}{m_{33}}$

Multiplying Eq.(3.2.4a) by $cos(\psi)$ and multiplying Eq.(3.2.4b) by $sin(\psi)$, we then get the following equations,

$$\dot{x}cos(\psi) = ucos^2(\psi) - vsin(\psi)cos(\psi) \tag{3.2.5a}$$
$$\dot{y}sin(\psi) = usin^2(\psi) + vsin(\psi)cos(\psi) \tag{3.2.5b}$$

To eliminate the term v, we add the two equations above and rearrange the terms, it then follows,

$$u = \dot{x}cos(\psi) + \dot{y}sin(\psi) \tag{3.2.6}$$

From the equation above, we can draw the conclusion that u can be written as an algebraic function of the three flat outputs and their derivatives $(\dot{x}, \dot{y}, \psi)$.

Similarly, the term v can be derived by multiplying Eq.(3.2.4a) by $sin(\psi)$ and multiplying Eq.(**??**) by $\cos\psi$,

$$v = -\dot{x}sin(\psi) + \dot{y}cos(\psi) \tag{3.2.7}$$

Eq.() then proves that v can be expressed as a function of the flat outputs and their derivatives $(\dot{x}, \dot{y}, \psi)$.

As Eq.(3.2.4c) shows, r can be explicitly expressed by the derivative of the heading, that is $\dot{psi}$.

$$r = \dot{\psi} \tag{3.2.8}$$

To prove that the inputs τ_u and τ_r are flat, we can rewrite Eq.(3.2.4d) and Eq.(3.2.4f) as,

$$\tau_u = \dot{u} + \beta_1 u - vr \tag{3.2.9a}$$
$$\tau_r = \dot{r} + \beta_3 r \tag{3.2.9b}$$

Taking the derivative of u and r with respect to time, we then get

$$\dot{u} = \ddot{x}cos(\psi) - \dot{x}sin(\psi)\dot{\psi} + \ddot{y}sin(\psi) + \dot{y}cos(\psi)\dot{\psi} \tag{3.2.10a}$$
$$\dot{r} = \ddot{\psi} \tag{3.2.10b}$$

Inserting Eq.(3.2.9a) into Eq.(3.2.10a), then the following equation can be proved to hold,

$$\tau_u = \ddot{x}cos(\psi) + \ddot{y}sin(\psi) + \beta_1(\dot{x}cos(\psi) + \dot{y}sin(\psi)) \tag{3.2.11}$$

Eq.(3.2.11) proves that input τ_u is flat, which can be written as a function of flat outputs and their derivatives $(\dot{x}, \dot{y}, \ddot{x}, \ddot{y}, \psi)$.

To further prove τ_r is flat, insert Eq.(3.2.8) and Eq.(3.2.10b) into Eq.(3.2.9b),

$$\tau_r = \ddot{\psi} + \beta_3\dot{\psi} \tag{3.2.12}$$

Since states [u, v, r] and inputs $[\tau_u, \tau_r]$ have all been proved that they can be written as functions of flat outputs and their derivatives, thus the system is flat. Differential flatness of the mathematical model has been demonstrated as above, which can be used as a powerful property to allow for trajectory planning without the need to solve for the differential equations of motion at each iteration. Moreover, the trajectory and its derivatives are sufficient to obtain the states and control inputs at each point along the trajectory. In the next section, a polynomial-based approach is proposed to smooth the piecewise-linear path based on the differential flatness of the mathematical model.

3.3 Path Optimization

Based on the property of Bézier curve and differential flatness, the path planning problem is then formulated as an optimization problem. The main objective is to generate a path from initial point to goal consisting of piecewise Bézier curve segments, while dynamics of the vehicle and static obstacle avoidance are involved by imposing constraints on physics and workspace.

Optimization is an essential tool applied in decision making, which consists of objective function and certain constraints. To take advantage of this tool, we have to identify the objective at the first step, which could be time, profit, energy consumption or other quantitative measure of the performance. The objective depends on some variables, called decision variables, reflecting the characteristics of the system. The main goal of this optimization problem is to determine the value for decision variables that optimize the objective function. Besides, the variables are restricted by some constraints, including equality constraints and inequality constrains. The process of identifying objective, variables, and constraints for a given optimization problem is referred to as modeling (Nocedal and Wright, 2006). Since the mathematical model has been constructed in the previous section, the next step is to choose an appropriate optimization algorithm and apply the algorithm to the model. Generally speaking, optimization is to find the minimal or maximal value of the objective function subject to constraints on the variables, and a general formulation is given as

$$\min_{x \in R^n} f(x) \qquad subject \quad to \begin{array}{ll} c_i(x) = 0, & i \in \mathcal{E}, \\ c_i(x) \geq 0, & i \in \mathcal{I}, \end{array} \tag{3.3.1}$$

where f(x) represents the objective function, x is the vector of variables. c_i denoted the constraint functions, $\mathcal{E}$ and $\mathcal{I}$ are sets of indices for quality and inequality constraints, respectively.

3.3.1 Decision Variables

To start with the optimization problem, the decision variable ought to be determined. As the Bézier spline depends on the positions of control points, it is reasonable to select control points as the decision variable. Assuming that Bézier spline is connected by m Bézier curve segments, the decision variables can be expressed using a vector as

$$x_1 = \{p_{0,1}, \ldots, p_{n,1}, p_{0,2}, \ldots, p_{n,2}, p_{0,m}, \ldots, p_{n,m}\} \tag{3.3.2}$$

where n is the degree of Bézier curve.

3.3.2 Objective Function

Considering path planning, the optimization problem is solved by minimizing the objective function, that is also known as cost function. Generally, objective function is constructed considering several aspects, including time cost, energy and fuel consumption with respect to path planning optimization. In this thesis, cost function f(x) is developed by exploiting the property of differential flatness and assigning a cost to each path generated by Bézier

curves. As illustrated in the previous section, it is natural to choose the x and y coordinates of Bézier curves as the flat outputs, whose derivatives are also easy to derive.

In addition to differential flatness, the energy consumption is also a key factor that should be incorporated into the objective function. Assuming that there is no sideslip along the path, the Newton's second law of motion and relationship between force and work can be expressed as follows

$$\sum F = m\dot{u}, \quad W_{ab} = \int_a^b F(s)ds \tag{3.3.3}$$

where the first expression is related to Newton's second law, $\dot{u}$ is the acceleration. W represents the work traveling along the curve from point a to b, while s denotes the length. Since the work is related to energy consumption and is dependent on acceleration, the new cost function can be formulated by using $\dot{u}$ as the variable.

$$J = \int_a^b g(\dot{u})ds, \quad g(\dot{u}) = \sqrt{|\dot{u}|} \tag{3.3.4}$$

Another vital assumption to derive the cost function is that the time dependent flat output can be directly replaced by the parameter of Bézier curve t. Hence, by substituting $\dot{u}$ with the first derivative $u'(t)$, the cost function then can be transformed into

$$J = \sum_{i=1}^{m} \int_0^1 \sqrt{|u_i'(t)|}dt \tag{3.3.5}$$

where i is the curve segment number, and $u_i'(t)$ can be derived by differentiating Eq. (3.2.6).

3.3.3 Constraints

To find the solution with respect to path planning, we formulate a constrained optimization problem, where constraints play a fundamental role, where constraints including equality and inequality constraints are scalar functions of decision variables x, and constraints are imposed that variable x should satisfy. All constrained involved in the path planning optimization are presented in this section, including various constraints regarding smoothness, boundary conditions, turning radius and static obstacles.

Parametric Continuity

Parametric continuity is a concept used to describe the smoothness of a parametric curve, denoted as C^n, where n represents the degree of continuity. By definition, a parametric curve is defined to be C^k continuous if $\frac{d^k p}{dt^k}$ exists and is continuous on [0, 1]. And different orders of parametric continuity can be expressed as

- C^0: Curve is continuous.
- C^1: First derivative of the curve is continuous.

- C^n: From first to n-th derivatives are all continuous.

As the generated curve is connected by several Bézier curve segments, the continuity at the joint should be taken into consideration. To sure C^0 continuity, the last control point of a curve segment must be the same as the first control point of next segment. For a fifth order Bézier spline, this constraint can be given as

$$p_{5,i} = p_{0,i+1}, \qquad i \in \mathcal{I}^{m-1} \tag{3.3.6}$$

where m represents the number of curve segments, while i denotes the i-th segment. Besides, C^1 continuity can also be guaranteed by imposing constraints on the first derivatives,

$$p_{5,i} - p_{4,i} = p_{1,i+1} - p_{0,i+1}, \qquad i \in \mathcal{I}^{m-1} \tag{3.3.7}$$

By combining Eq. (3.3.6) and Eq. (3.3.7), the constraint can be further simplified as

$$p_{1,i+1} + p_{4,i} = 2p_{5,i} = 2p_{0,i+1}, \qquad i \in \mathcal{I}^{m-1} \tag{3.3.8}$$

In order to have a continuous curvature, C^2 continuity can be obtained by constraining the second derivative at the endpoints,

$$p_{2,i+1} - 2p_{1,i+1} + p_{0,i+1} = p_{5,i} - 2p_{4,i} + p_{3,i}, \qquad i \in \mathcal{I}^{m-1} \tag{3.3.9}$$

This constraint can also be simplified by inserting Eq. (3.3.6) as

$$p_{2,i+1} - 2p_{1,i+1} = -2p_{4,i} + p_{3,i}, \qquad i \in \mathcal{I}^{m-1} \tag{3.3.10}$$

The given parametric constraints can all be characterised by linear equality constraints. The path is able to have C^0, C^1 and C^2 by meeting these constraints, and only the first and last three control points of each segments are required to be considered.

Geometric Continuity

Since the parametric continuity mainly focus on the smoothness of parameterization, and does not necessarily reflect the smoothness of the curve, geometric continuity is thus introduced to ensure that path also has continuity on course and curvature. Geometric continuity is a concept to describe smoothness of curve through geometry rather than algebra, and various orders of geometric continuity can be given as

- G^0: Curve is connected at joint points.
- G^1: Tangent vector is continuous at joint points.
- G^2: Center of curvature is shared at joint points.

To enable the fifth order Bézier spline has G^2 continuity, following constraints are imposed,

$$\begin{aligned} p_i'(1) &= a_i p_{i+1}'(0), && i \in \mathcal{I}^{m-1} \\ p_i''(1) &= a_i^2 p_{i+1}''(0) + b_i p_{i+1}'(0), && i \in \mathcal{I}^{m-1} \end{aligned} \tag{3.3.11}$$

where a_i and b_i are newly introduced decision variables, denoted as a vector set,

$$x_2 = \{a_1, a_2, \ldots, a_{m-1}, b_1.b_2, \ldots, b_{m-1}\} \tag{3.3.12}$$

where m denotes the number of curve segments, a_i are a set of strictly positive constants, while b_i represents a set of arbitrary constants.

Boundary Conditions

For path planning algorithm, it is essential to satisfy the boundary conditions of the vehicle. Béizer spline is generated path through the initial and final control points according to the boundary conditions, while it is not required to necessarily pass through the intermediate control points. Assuming the initial and final pose of the vehicle is defined as $q_0 = [x_0, y_0, \psi_0]$ and $q_f = [x_f, y_f, \psi_f]$, respectively, and boundary conditions are required to be compatible with the initial and final pose of vehicle. Figure 3.2 shows that the Bézier curve is able to be constrained by boundary conditions with $q_0 = [0, -1, 60°]$ and $q_f = [8, 0, -45°]$.

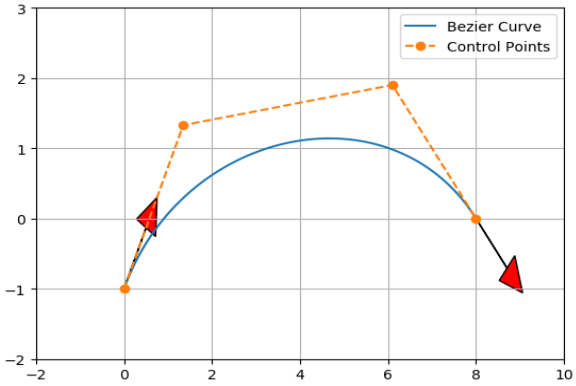

Figure 3.2: Bézier curve satisfying the boundary conditions at endpoints

Since fifth order Bézier spline is adopted in this thesis and it has been proved to have G^2 continuity, the curvature of the curve is capable of smoothly transitioning from the start point to the end point while meeting the heading constraints regarding boundary conditions. Hence, initial and final constraints for positions and heading are formulated by a set of linear equality constraints, given as

$$p_{0,1} = W_0, \quad p_{5,m} = W_f \tag{3.3.13}$$

where W_0 and W_f represent the positions of initial and final control points. As for the orientation of the vehicle, the heading at the endpoints is dependent on the tangent of the curve, which can be restricted by a set of constraints,

$$l_0 \begin{bmatrix} sin(\psi_0) \\ con(\psi_0) \end{bmatrix} = 5(p_{1,1} - p_{0,1}), \quad l_f \begin{bmatrix} sin(\psi_f) \\ con(\psi_f) \end{bmatrix} = 5(p_{5,m} - p_{4,m}), \tag{3.3.14}$$

where ψ_0 and ψ_f represent the heading angle at the start and goal control points, respectively, while l_0 and l_f are strictly positive decision variables, representing the length of tangents at the endpoints, denoted as a new vector

$$x_3 = l_0, l_f, \quad where \quad l_0, l_f > 0 \tag{3.3.15}$$

After imposing the constraints above, the decision variables for the path optimization problem can eventually be determined and written as a vector,

$$x = \{x_1, x_2, x_3\}, \tag{3.3.16}$$

where x_1 is dependent on the positions of control points, x_2 is included to ensure path is geometric continuous, while x_3 is introduced to satisfy boundary conditions.

Turing Radius

Due to the physical and dynamic constraints of ASV itself, minimum turning radius should be incorporated as another constraint, reflecting the capability of the vehicle's maneuverability. The relationship between curvature and turning radius has been explained in the previous section, that the turning radius is the reciprocal of the curvature. Therefore, the constraints on turning radius lead to the consequence that the curvature along the path should be less than the maximum curvature, shown as

$$\kappa(t) = \frac{\mid p_p'(t) \times p_p''(t) \mid}{\mid p_p'(t) \mid^3} = \frac{\mid x_p'(t)y_p''(t) - x_p''(t)y_p'(t) \mid}{(x_p'(t)^2 + y_p'(t)^2)^{\frac{3}{2}}} \tag{3.3.17}$$

$$\kappa_{max} = \frac{1}{R_{min}} \tag{3.3.18}$$

$$\kappa_i(t) < \kappa_{max}, \quad i \in \mathcal{I}^m \tag{3.3.19}$$

where κ_{max} denotes the maximum curvature, while R_{min} represents the corresponding minimum turning radius. And the constraint is formulated by restricting the curvature along the path smaller than maximum curvature. Besides, this constrained is characterised as nonlinear inequality constraint.

Static Obstacles

Besides, as collision avoidance (COLAV) is an inevitable issue in this case, a collision avoidance scheme should be developed and implemented taking static obstacles into account. The main objective is to determine the safety margin that the vehicle is allowed to travel through without suffering the risk of colliding with ant obstacle. To simplify the case, we assume that each obstacle can be represented by a circle with radius r_j and center point is located at $c_j = (x_j, y_j)$, yielding the forbidden zones. The corresponding constraint is imposed on the distance from each waypoint $p_i(t)$ on the Bézier curve to any center of static obstacles c_j. Assuming there exists n static obstacles, constraint can be expressed as

$$r_j \leq |p_i(t) - c_j|, \quad i, j \in \mathcal{I}^m \times \mathcal{I}^n \tag{3.3.20}$$

The formulated constraint above should be satisfied for any pair(i, j) to guarantee a feasible and collision-free path. However, if the constraint with respect to obstacle is required to hold for every pair of i and j, a large number of constraints will be created, resulting in high computational loads. To shrink the amount of obstacle constraints, pair of i and j should be pruned and merely consider the minimum distance between the path curve segments

and obstacles. In other words, only the most risky pairs of i and j are taken into account. Thus, the corresponding constraint can be reformulated as

$$r_j \leq min\{|p_i(t) - c_j|\}, \quad i, j \in \mathcal{I}^m \times \mathcal{I}^n \tag{3.3.21}$$

3.3.4 Implementation

Based on previous work (Lande, 2018), this optimization algorithm is implemented in $MATLAB^{®}$ where "fmincon" is used as the optimization solver providing SQP (Sequential Quadratic Programming) algorithm. Since this path optimization is defined as a nonlinear constrained optimization problem due to the existence of nonlinear objective function and constraints, the SQP approach is employed as one of the most effective methods solving for nonlinear constrained optimization. Consider a general nonlinear programming problem in the form as

$$\begin{aligned} & minf(x) && \text{(3.3.22a)} \\ subject \quad to \quad & c_i(x) = 0, i \in \mathcal{E}, && \text{(3.3.22b)} \\ & c_i(x) \geq 0, i \in \mathcal{I}, && \text{(3.3.22c)} \end{aligned}$$

The model is then reformulated by linearizing both equality and inequality constraints as

$$\begin{aligned} \min_p \quad & f_k + f_k^T p + \frac{1}{2} p_{xx}^T{}^2 \mathcal{L}_k p && \text{(3.3.23a)} \\ subject \quad to \quad & c_i(x_k)^T p + c_i(x_k) = 0, i \in \mathcal{E}, && \text{(3.3.23b)} \\ & c_i(x_k)^T p + c_i(x_k) \geq 0, i \in \mathcal{I}. && \text{(3.3.23c)} \end{aligned}$$

As (Lande, 2018) has proved that the quintic Bézier spline has the overwhelming superiority over cubic and quartic Bézier spline by possessing the property of freely set first and second derivatives at endpoints besides locality and C^1, C^2 continuity, quintic Bézier spline is exploited to generate feasible paths. To initialize the algorithm, relevant parameters should be specified including initial and final pose of the vehicle $[x, y, \psi]$, number of obstacles n, minimum turning radius R_{min} and radius of obstacles r. A example of path planning algorithm based on Bézier curve and optimization is displayed in Figure 3.3, and the initialization parameters are given in Table 3.1.

Initial pose	(0, 0, 0°)	Final pose	(1000, 1000, 30°)
Degree of Bézier curve	5	No. of obstacles	10
Minimum turning radius	100	Radius of obstacles	60

Table 3.1: Initialization parameters

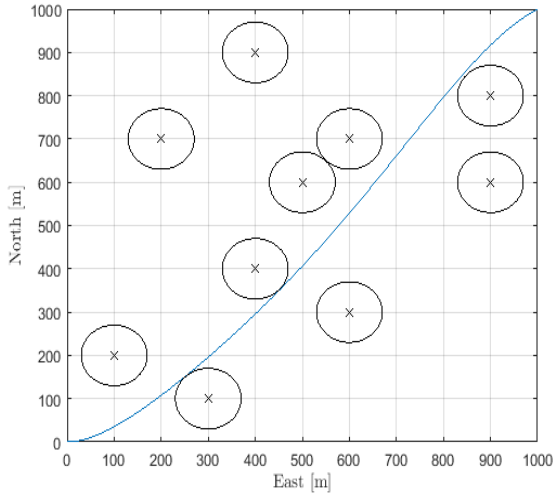

Figure 3.3: Feasible path from start to goal based on Bézier curve

The result above manifests that the path optimization algorithm based on Bézier curve is capable of generating a feasible path passing through start point and goal while avoiding collision with static obstacles at the same time. As a consequence, this path planning method is employed as a deliberate approach that will be incorporated into hybrid COLAV architecture in chapter 5.

Chapter 4

Path Planning and Collision Avoidance

Path planning is a widespread problem, not only associated with robotics, but also applied in cybernetics, artificial intelligence and other relevant engineering fields. When it comes to path planning, a number of factors should be taken into account. In general, it is necessary to consider the initial and final states of the vehicle, internal and external constraints, specifically, obstacle avoidance and vehicle dynamical constraints should be satisfied at the same time. Moreover, cost criterion is non-trivial such that the generated trajectory will minimize the cost while meeting all constraints. In some cases, motion planning module can operate off-line, by obtaining a prior knowledge of the vehicle and given environment. While suitable sensors are essential to be employed to monitor the vehicle motion such that the control system can adjust to movements and operate on-line.

Path planning algorithm generates geometric paths, from the initial point to the final point, while passing through a couple of waypoints based on the collision-free goal and no specific time law is involved. Taking the given geometric path as a basis, trajectory planning integrate time factor into planning algorithm. Time information at the waypoints has important effect on both kinematic and dynamic properties of the motion. In addition, trajectory planning is of great importance in multiple marine vehicles scenarios, especially where collision avoidance with dynamic obstacles or other vehicles should be taken into consideration, as the vehicle must satisfy spatial and temporal schedules simultaneously. And the trajectory tracking depends on absolute timing, which does not allow for on-line modification of the plan in case of disturbances during execution (Häusler et al., 2010)

A large number of motion planning methods are considered to address the issue. Among those existing motion planning approaches, the main objective is to find and optimize the trajectory through a complex environment while avoiding the collision with obstacles in the vicinity of a vehicle. In this section, we will give detailed explanation about path planning and collision avoidance, presenting an overview of a couple of motion planning

algorithms, compare the advantages and disadvantages of each technique, as well as the application fields. In particular, dynamic window (DW) algorithm exploited as the reactive COLAV method in this thesis, is emphatically introduced.

4.1 Deliberate Path Planning

Deliberate methods that rely on the information of the complete environment, is normally referred to as path planning, aiming to find a route from start to goal point. Majority of deliberate path planning algorithms are complete, indicating that the algorithm is always able to find the path if it exists. Therefore, it is less likely to fail when encountering local minima compared to a reactive COLAV method. However, the main downside of deliberate method is the demand for long computing time, which makes it no longer applicable to rapidly changing dynamic environment.

4.1.1 Grid-based approaches (A* and D*)

Grid-based approaches have been widely used in recent years, in which the environment is mapped to a set of cells representing individual obstacles at the corresponding positions. Typically, the graph search method aims to find the optimal path with a grid superimposed over an area. Informed search algorithms, like A* (Hart et al., 1968) and D* (Stentz et al., 1995) are adopted to generate optimal paths connecting the initial position to the target position while avoiding obstacles. A* is the classic and wildly used artificial intelligence search algorithm, which uses best-first search algorithm with heuristic functions. In general, the cost function considered in A* algorithm can be written as follows,

$$f(s) = g(s) + h(s) \tag{4.1.1}$$

where g(s) denotes the path cost or distance of getting from the root of the search tree to current node s, while h(s) represents the heuristic function: an estimate of the distance from node s to the goal state. Therefore, f(s) denotes the total expected cost of a optimal solution path from the root, through s, to the goal state. (Peng et al., 2015) presents the optimal path using A* search algorithm, where the path with the yellow grid in it is the optimal path from the start point to the end point as shown in Figure 4.1

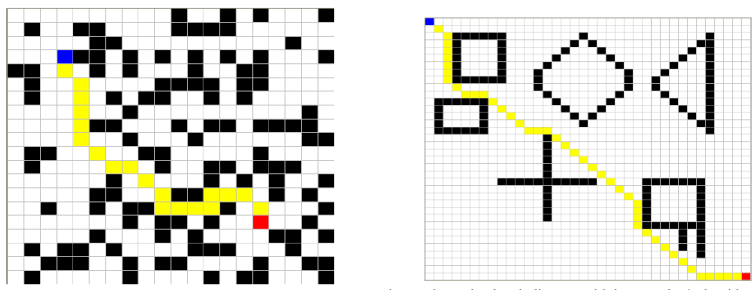

Figure 4.1: Optimal path with diagram with A* algorithm search (Peng et al., 2015)

(Stentz et al., 1995) proposed a popular graph search algorithm, called D* algorithm, which is applied in rapidly-changing and dynamic environments for its property of fast re-planning. It has been distinguished from the A* algorithm, as it can be performed without the heuristic function. D* algorithm is becoming important for path planning when navigate through unknown environment. This technique seeks the sequences of similar search problems by utilizing previous search patterns. The D* algorithm can be divided into two stages, initial planning and re-planning phases. When the vehicle stays still at the start point, the first phase initial planning is executed and re-planning will occur when the robot detects nodes with changed occupancy values during motion process. For every searched node n, the D* algorithm computes the path cost g(n) from the node n to the goal state, and the value of the key function k(n) for the re-planning stage, which stores the old values g(n) before weights change. (Dakulovic et al., 2011) proposes a novel algorithm called complete coverage D* (CCD*) algorithm, based on the D* search of the two-dimensional environment occupied grid graph, which is capable of producing new complete coverage path as the environment changes. The performance of this algorithm has been evaluated using a Pioneer 3DX mobile robot equipped with a laser range finder.

Applications based on grid-based approaches, such as parking and navigating the road in unstructured environments are successfully implemented. Nevertheless, this method is somehow limited with respect to complicated environments due to the exponential growth of computation complexity.

4.1.2 Rapidly-Exploring Random Tree (RRT)

In order to obtain a set of feasible trajectories, sampling based approaches directly sample the state space of the vehicle, and choose the best path to implement by evaluating the corresponding cost function. AS for nonholonomic path planning, Rapid-Exploring Random Tree (RRT) technique with RRT variants greatly contributes, which allows for searching in non-convex and high-dimensional spaces by randomly generating space-filling trees. Unlike A* and D*, RRT approach directly take the dynamics of the vehicle into account. In (Devaurs et al., 2016), based on the combination of two RRT variants, an efficient sampling based approach was formulated to address a complicated path planning problem. Based on RRT algorithm, (Naderi et al., 2015) presents an improved real-time path planning approach, named dubbed Real-Time RRT, which enables the tree root to move with the agent while still conserving the previously sampled paths. This online tree planning strategy does not have to wait for the tree to be fully constructed, since trees are being expanded concurrent with taking actions. Figure 4.2 shows that this algorithm allows the user to rapidly switch the target by efficiently taking advantage of the expanded tree. However, in sampling based path planning method, efficient heuristic functions are required to achieve real-time planning which is hard to implement in some complex environments.

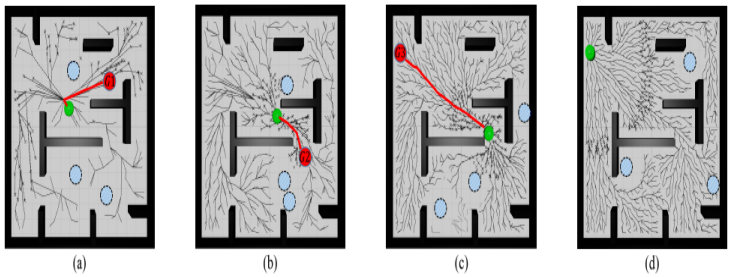

Figure 4.2: Path from initial point to different goals and dynamic obstacles (Naderi et al., 2015)

4.2 Reactive Collision Avoidance

Based on data and information of the immediate environment from the sensors, reactive approach also referred to as reactive COLAV method, is able to make real-time response that requires low computation capacities, which makes it superior when performing in real-time and dynamic environments. Due to the property that only a subset of configuration space is considered, reactive COLAV are mostly incomplete methods, yielding the consequence of being stuck in local minima. Some classic reactive COLAV methods, including Potential Field method (Khatib, 1986), Velocity Obstacle (Fiorini and Shiller, 1998) and Dynamic Window algorithm (Fox et al., 1997) are elaborated as follows.

4.2.1 Potential Field Method

Potential Field Algorithm originally proposed in (Khatib, 1986), is widely used in path-planning applications, which assigns a value calculated using an artificial potential function to each point and simulates the reaction of the vehicle to the potential field as it navigates towards the minimum potential (Radmanesh et al., 2018). The basic idea of the algorithm is that the motion in the configuration space is regarded as a moving point subject to the potential field, that is generated by the target configuration and the obstacles in C space. To be specific, the target configuration generates attractive potential, while the obstacles impose a repulsive force or potential on the moving vehicle, such that the vehicle is attracted towards the goal point and keep away from the obstacles.

In the case of the artificial potential algorithm, two types of functions are usually considered, one of which is based on harmonic functions, solving a partial differential equation with a Laplace term, while the other is defined to minimize the distance-to-go function (Hirsch, 2012). Nevertheless, there exist some challenges for this algorithm, that is the local minima trap issue, which appears when all attractive and repelling artificial potential cancel out each other. This is often the case when in the situation where the obstacle is located between the vehicle and goal or obstacles are closely spaced. Several approaches have been proposed to address this issue in (Ahuja and Chuang, 1997), for instance, adding an imaginary obstacle in the local minima region to repel the vehicle.

4.2.2 Velocity Obstacle

Velocity obstacle (VO) is a velocity space based method designed for robot motion planning in (Fiorini and Shiller, 1998). Velocity obstacle represents a set of vehicle's velocities that might lead to a collision with either static or dynamic obstacle. This VO method is able to consider moving obstacle as static one by mapping the dynamic obstacle into the $\mathcal{C}$-space of the vehicle and calculating the relative velocity. Given the current position and velocity of the vehicle and obstacles, velocities are selected outside of the VO to ensure safe operations and collision-free trajectory. In (Fiorini and Shiller, 1998), VO method has been demonstrated with good performance for point and disk robots with the presence of static and moving obstacles, and extensively for an automated vehicle in highway scenario. And one thing to note is that, VO method is guaranteed to be dynamically feasible by incorporating admissible velocities with the constraints of vehicle's accelerations.

Considerable extensive work has been done based on VO method, (Van den Berg et al., 2008) proposes a new concept, named "Reciprocal Velocity Obstacle", for real-time and multi-agent navigation including both static and moving obstacles. Based on Velocity Obstacle approach, (Kuwata et al., 2014) introduces a motion planning algorithm for USV to navigate in dynamic and cluttered environments involving International Regulations for Preventing Collisions at Sea (COLREGS) rules, enabling the USV to be safely deployed in environments with other traffic boats. Full-scale experiment is carried out and demonstrates that modified VO method is capable of navigating USV and obeying the COLREGS rules.

4.2.3 Dynamic Window Approach

Dynamic window method is a local reactive avoidance technique, originally proposed by (Fox et al., 1997). The advantage of this approach is that it directly incorporates the dynamics of the vehicle, that is, the search for commands is implemented in the space of velocities, consists of translational velocity and rotational rate.

Search Space

Circular trajectories

By adopting velocity space, the pruning of the search space enormously simplify the computational effort. The trajectory of the ASV can be approximated by a sequence of circular arcs, and each arc is uniquely determined by the velocity vector (v, w) with the radius $r = \frac{v}{w}$. For the ASV, the velocity pair is referred as surge speed u and yaw rate r, thus the velocity vector is transformed into (u, r). For each velocity pair within the velocity space, the dynamic window approach is designed to predict the trajectory that velocity pair (u, r) might generate for the next n time intervals. In order to simplify the optimization, we only consider the first time interval and assume that within the remaining n-1 time intervals, the velocity vector (u, r) remains unchanged. This assumption is derived based on the observation that search is automatically repeated after each time interval, while velocity will remain constant if there are no new commands.

Admissible velocities

The existence of obstacles in the nearby environment imposes restrictions on the translational and rotational velocities. The distance to the closest obstacle on the trajectory determines the maximal velocity of the vehicle. That is, the velocity is considered admissible if the vehicle is able to stop before it hits the next obstacle. As a consequence, the search space is reduced to a set of velocities that allow the vehicle to stop without colliding with any obstacle. The set of admissible velocities then can be defined as,

$$V_a = \{(u,r)|u \leq \sqrt{2 \cdot dist(u,r) \cdot \dot{u_b}} \wedge r \leq \sqrt{2 \cdot dist(u,r) \cdot \dot{r_b}}\} \tag{4.2.1}$$

where $dist(u,r)$ represents the distance to the closest obstacle on the corresponding trajectory, while $\dot{u_b}$ and $\dot{r_b}$ refer to the deceleration in surge and yaw, respectively.

Dynamic window

Due to the presence of kinematic and dynamic constraints, the space is reduced to a certain span around the current velocity, which only consists of admissible velocities that the vehicle can reach within the next time interval. Thus, the dynamic window can be described as,

$$V_d = \{(u,r)|u \in [u_a - \dot{u} \cdot \Delta t, u_a + \dot{u} \cdot \Delta t] \wedge \omega \in [r_a - \dot{r} \cdot \Delta t, r_a + \dot{r} \cdot \Delta t]\} \tag{4.2.2}$$

where the $\dot{u}$ and $\dot{r}$ represent the accelerations in surge and yaw direction, and (u_a, r_a) is the current velocity vector.

Resulting search space

With the restriction imposed on the velocity space, the resulting search space is the intersection of three restricted velocity set, namely, the set of possible velocities V_s, admissible velocities V_a and dynamic window V_d. Consequently, the resulting search space V_r now can be represented as,

$$V_r = V_s \cap V_a \cap V_d \tag{4.2.3}$$

where the possible velocities V_s is limited by the extreme value of the surge speed u and yaw rate r.

$$V_s = \{(u,r)|u \in [0, u_{max}] \wedge r \in [-r_{max}, r_{max}]\} \tag{4.2.4}$$

Objective Function

Among those velocity pairs within the resulting search space V_r, velocity vector (u,r) is chosen to maximize a certain objective function, which consists of some criteria, like target heading, clearance and velocity.

$$\begin{aligned} &G(u,r) = \alpha \cdot goal(u,r) + \beta \cdot dist(u,r) + \gamma \cdot vel(u,r) \\ &s.t.(u,r) \in V_r, \end{aligned} \tag{4.2.5}$$

where the terms goal(u, r), dist(u, r) and vel(u, r) is weighted by the factors α, β and γ. The term goal(u, r) denotes the desired heading angle towards the goal, used to measure the progress towards the target, which will reach the maximum value if the vehicle moves directly towards the goal position. This term can be mathematically denoted as the angle between the vector pointing to the goal and vector connecting start point and current position,

$$goal(u, r) = arccos(\frac{\overrightarrow{OA} \cdot \overrightarrow{OB}}{|\overrightarrow{OA}| \cdot |\overrightarrow{OB}|}), \tag{4.2.6}$$

where O and A represent the start and goal point, respectively, while B denotes the current point of vehicle.

And term $dist(u, r)$ represents the distance to the closet obstacle, and it will be set to a large constant value if there is no obstacle on the trajectory.

$$dist(u, r) = \frac{1}{r_{min}}, \tag{4.2.7}$$

r_{min} is referred to the distance from current position of vehicle to the nearest obstacle, that reveals the most threatening obstacle. And term dist(u, r) is the reciprocal of r_{min}, which will reach a maximum value when an obstacle occurs in the vicinity.

The last term regarding velocity vel(u, r) can be expressed as

$$vel(u, r) = u_{max} - u_c. \tag{4.2.8}$$

where u_{max} and u_c represent the maximum surge speed and the current velocity of the vehicle, respectively. The velocity term vel(u, r) is simply the difference between these two velocities, which means vel(u, r) is exclusively dependent on surge speed u. And velocity of the vehicle will approach the maximum speed as soon as possible within the restricted range of acceleration.

Besides, the dynamic window algorithm is originally designed for robots with first-order nonholonomic constraints and constant acceleration limits, where the first-order nonholonomic constraint is also said to be partially integrable. Since the heading angle of robot $\theta(t)$ depends upon the translational velocity v, which yields a first-order nonholonomic constraint. The general motion equations of the robot describe the kinematic characteristics of the robot, coordinate at time t_n, $x(t_n)$ and $y_(t_n)$ can be expressed as a function of $x(t_0)$, $y(t_0)$, v(t) and $\theta(t)$ as

$$x(t_n) = x(t_0) + \int_{t_0}^{t_n} v(t) \cdot cos\theta(t)dt \tag{4.2.9a}$$

$$y(t_n) = y(t_0) + \int_{t_0}^{t_n} v(t) \cdot sin\theta(t)dt \tag{4.2.9b}$$

As introduced in section 2, the mathematical model of the ASV can be simplified by a 3D representation,

$$\dot{\eta} = R(\psi)v \tag{4.2.10}$$

$$M\dot{v} + C(v)v + D(v)v = \tau \tag{4.2.11}$$

In contrast to the robot in the original DW approach, ASV is underactuated with second-order constraints. Underactuated vessel is defined to have less control inputs than the dimension of configuration space ($r < n$) (Fossen, 2011), therefore the ASV is considered to be underactuated with only two control points: propeller force and rudder angle, while the dimension of configuration space for surface vessel is 3 ($2 < 3$). Furthermore, constraint is supposed to be second-order nonholonomic if it is not partially integrable in (Oriolo and Nakamura, 1991), where conditions for partial integrability are proposed. Consider a (n-m)-dimensional constraint of first-order,

$$g(q, \dot{q}, t) = 0, \quad g \in R^{n-m}. \tag{4.2.12}$$

As constraint is defined to be partially integrable if it can be integrated to Eq. (4.2.12), we can differentiate Eq. (4.2.12) with respect to the time variable t,

$$\frac{\partial g}{\partial q}\dot{q} + \frac{\partial g}{\partial \dot{q}}\ddot{q} + \frac{\partial g}{\partial t} = 0 \tag{4.2.13}$$

Constraint is partially integrable if it is completely identical to Eq. (4.2.13), indicating that the term $\frac{\partial g}{\partial t}$ must be constant. In the case of ASV, considering the mathematical model, constraints are partially integrable if Eq. (4.2.11) is able to be integrated into the form as

$$g(v, \dot{v}, t) = 0 \tag{4.2.14}$$

The term $C(v)$ + $D(V)$ must be constant to guarantee the constraint is partially integrable, which conflicts with practical situations. The Coriolis and centripetal term $C(v)$, damping matrix $D(v)$ do not hold for the condition of being constant, meaning the constraint is not partially integrable, and thus second-order nonholonomic constraint.

Above all, the original dynamic window approach intended for robot with first-order nonholonomic constraint and constant acceleration limits is not applicable for ASV with second-order nonholonomic constraint and time-varying limits. (Eriksen et al., 2016) proposed a modified DW algorithm to adapt to AUV model without degraded performance, based on the integration of a new trajectory prediction method and a modified search space. The sideways motion the vehicle has been taken into account, and search space is also modified to account for the actuator limitations to further ensure feasible steering commands. The limit of yaw rate acceleration is assumed to be symmetric in the original DW algorithm, described as $\dot{r}_{min} = -\dot{r}_{max}$, resulting in the set of possible velocity as

$$V_s = \{(u, r) | u \in [0, u_{max}] \wedge r \in [-r_{max}, r_{max}]\} \tag{4.2.15}$$

The dynamic window is then modified due to the asymmetry of the acceleration limits,

$$V_d = \{(u, r) | u \in [u_a + \dot{u}_{min} \cdot \Delta t, u_a + \dot{u}_{max} \cdot \Delta t] \wedge \omega \in [r_a + \dot{r}_{min} \cdot \Delta t, r_a + \dot{r}_{max} \cdot \Delta t]\} \tag{4.2.16}$$

Besides, the set of possible velocities V_s is reformulated regarding the actuator saturation limits,

$$V_s = \{(u, r) | g(u, r) \geq 0\} \tag{4.2.17}$$

where the positive semi-definite function g(u, r) is derived by computing the boundaries of steady state solution of the dynamics, expressed as

$$M\dot{v}_r = \tau - C(v)v - D(v)v = 0 \tag{4.2.18}$$

The set of possible velocities V_s is then restricted to feasible velocity pairs of rudder and surge actuation.

The dynamic window (DW) algorithm is adopted as a reactive COLAV method due to its low computational complexity and the property of flexibility and responsiveness to the rapidly-changing environment, especially when moving obstacles are considered in this thesis.

Chapter 5

Hybrid COLAV Method

To sum up, previous sections regarding popular COLAV methods have demonstrated the upsides and downsides of both deliberate and reactive COLAV methods. Specifically, large amount of computational time is required while performing global path planning algorithm, which is prone to cause trouble when encountering unexpected situations due to the lack of real-time rapid response to dynamic and cluttered environments. Furthermore, to cope with the collision avoidance with dynamic obstacles, dynamic window approach functioning as a reactive COLAV method is adopted, even though this algorithm suffers from the risk of being trapped in local minima and not able to find the route. In order to generate the feasible and collision-free trajectory towards the goal in the presence of both static and moving obstacles, a hybrid COLAV architecture is motivated to take advantage of these two methods, resulting in low computational complexity and less probability of being stuck in local minima.

5.1 Hybrid COLAV Architecture

Base on the integration of global and local COLAV approaches, hybrid COLAV methods are widely used due to the restriction and drawbacks of either method. To utilize advantages of both methods, (Seder and Petrovic, 2007) proposes an improved dynamic window algorithm incorporated with a focused D* search algorithm, such that the vehicle is able to avoid collision with moving obstacles and less likely to be trapped in local minima. Besides, (Serigstad et al., 2018) introduces a hybrid dynamic window approach, functions as an interface to any deliberate COLAV method which generates time parameterized trajectories, enabling vehicles to avoid local minima. Moreover, (Lopes et al., 2016) proposes a hybrid motion planner intended for robot with first-order nonholonomic constraints to navigate in constrained indoor environment, combining A* algorithm and dynamic window approach.

In general, the hybrid architecture can be formulated as a hierarchy, decomposed into three layers: global path planner, interface including path tracking algorithm and local COLAV

method. In this thesis, the global path is generated within the framework of optimization, based on Bézier curves, and dynamic window approach is employed as a reactive COLAV method taking the moving obstacles into account. While path tracking algorithm (PLOS) is utilized to convert the global path into the desired trajectory, functioning as a guidance for DWA, such that the interface between the global and local layer is developed, aiming at diminishing the distance from current position to the desired trajectory at each time step. Hybrid navigation architecture can be intuitively illustrated by the flow diagram, shown as Figure 5.1.

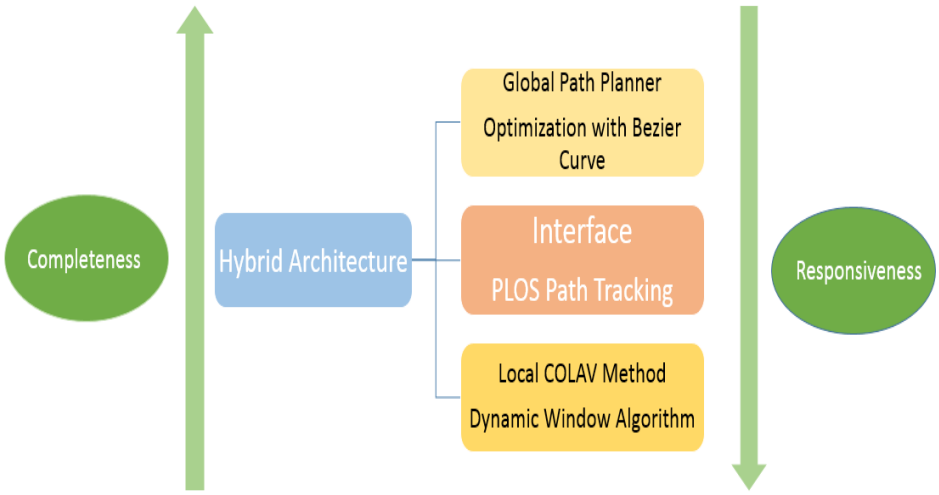

Figure 5.1: Diagram of hybrid COLAV architecture illustrating the framework

In addition to explaining the framework of the hybrid architecture, the diagram above also indicates the tendency of completeness together with responsiveness, between the global and local layer. In specific, the reactive CLOAV method is more responsive to the unforeseen and rapidly-changing environment based on immediate data from the sensor, and thus plays an important role in collision avoidance with moving obstacles. By contrary, deliberate path planning algorithm that relies on information of the complete environment is less responsive, but is capable of generating a feasible path towards the target, in other words, global method is more complete.

5.2 Modified Dynamic Window

The main objective of DW algorithm is to avoid collision with moving obstacles. In this thesis, we assume that prior knowledge of moving obstacles is provided, such that the trajectory and positions of moving obstacles at each time step are known. Existing methods

solving the issues of moving obstacles can be generally divided into two groups, characterised by extensive state-time methods and potential field approaches. (Fraichard, 1993) introduces state-time space as a tool, converting the constraints imposed by moving obstacles to static forbidden regions, since the prior knowledge of moving obstacle is give. While potential field method (Khatib, 1986) is capable of avoiding moving obstacles by simply generating repulsive force that repels the vehicle from dynamic obstacles. In contrast to the methods above, the basic concept of dynamic window with respect to moving obstacles can be described as calculating the trajectory of moving obstacle with prior knowledge, and predicted trajectory of vehicle based on the velocity pairs, thus, the distance between each moving obstacle and vehicle is utilized as an evaluating term in the objective function. The trajectory of moving obstacles with constant heading and speed, together with predicted trajectory of vehicle can be calculated by following expressions, respectively.

$$\begin{aligned} x_o(t_{k+1}) &= x_o(t_k) + v_o \times \psi_o \times \Delta t \\ y_o(t_{k+1}) &= y_o(t_k) + v_o \times \psi_o \times \Delta t \end{aligned} \tag{5.2.1}$$

where x_o, y_o ψ_o denote the pose of moving obstacles including position and heading angle, v_o and δt represent the velocity and time interval, respectively. Note that the expressions above are solely applicable to obstacles moving along straight lines, trajectories of other types will be introduced in later section.

$$\begin{aligned} x(t_{k+1}) &= x(t_k) + u \times \psi \times \Delta t \\ y(t_{k+1}) &= y(t_k) + u \times \psi \times \Delta t \\ \psi(t_{k+1}) &= \psi(t_k) + r \times \Delta t \end{aligned} \tag{5.2.2}$$

where the pose of the vehicle consists of position and heading, denoted as $[x, y, \psi]$, while velocity pair $[u, r]$ is referred to as surge speed and yaw rate.

Based on the original dynamic window approach discussed in Chapter 4, the original objective function including evaluation criteria, like progress towards the goal, distance to obstacle and maximum velocity, can be described as

$$\begin{aligned} &G(u, r) = \alpha \cdot goal(u, r) + \beta \cdot dist(u, r) + \gamma \cdot vel(u, r) \\ &s.t.(u, r) \in V_r, \end{aligned} \tag{5.2.3}$$

In order to integrate with the deliberate COLAV method, necessary modifications are further supplemented in DW algorithm to adapt to hybrid COLAV method. To fully exploit the global predefined path generated by deliberate method, an additional path alignment term denoted as $align(p_d, p_p)$ is incorporated into the objective function, which therefore becomes a key factor to evaluate the performance of the predicted trajectory from DW algorithm, and further determine the optimal velocity pair (u, r). Therefore, the modified objective function is given as

$$\begin{aligned} &G(u, r) = \alpha \cdot goal(u, r) + \beta \cdot dist(u, r) + \gamma \cdot vel(u, r) - \delta \cdot align(p_d, p_p) \\ &s.t.(u, r) \in V_r, \end{aligned} \tag{5.2.4}$$

where term $align(p_d, p_p)$ represents the distance between point on pre-defined trajectory and current position determined by velocity pair (u, r) at each time step. Besides, these

weight factors α, β, γ and δ determine how the hybrid COLAV favors trajectory concerning collision avoidance or aligning with the global preplanned path.

5.3 Interface Between Deliberate and Reactive Methods

To develop the interface between deliberate and reactive COLAV methods, temporal assignment should be considered with respect to global path, in other words; the predefined path generated by global path planner is obliged to be transformed into the desired trajectory as a guidance for local COLAV method. A combined pure pursuit and LOS algorithm (PLOS) introduced in Chapter 2 is utilized as a path tracking algorithm, where LOS accounts for the position error or cross-track error d regarding desired path, while pure pursuit is used to compensate for the heading error $(\theta_d - \psi)$ towards next waypoint. PLOS path following algorithm enables the vehicle to track the desired path, yielding pose $[x_d(t), y_d(t), \psi_d(t)]$ and velocity pair $[u_d(t), r_d(t)]$ at each time step, and the generated trajectory is shown in Figure 5.2.

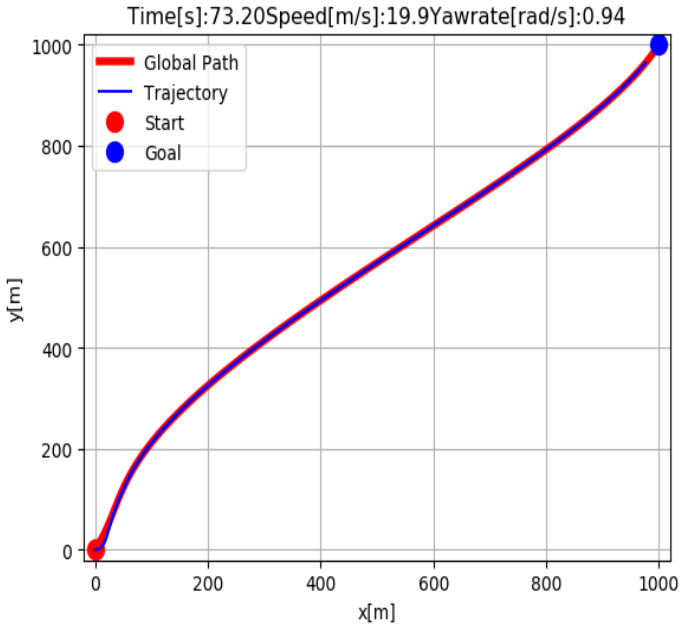

Figure 5.2: PLOS path tracking algorithm generates trajectory following the desired global path

The desired trajectory consists of a set of complete state of vehicle, and state at time t $p_d(t)$ can be presented as

$$p_d(t) = [x_d(t), y_d(t), \psi_d(t), u_d(t), r_d(t)] \tag{5.3.1}$$

As introduced above, a path alignment term is incorporated into the objective function, resulting in modified dynamic window. The predicted trajectory based on each velocity pair also generates a complete state at time t, defined as

$$p_p(t) = [x_p(t), y_p(t), \psi_p(t), u_p(t), r_p(t)] \tag{5.3.2}$$

The trajectory alignment is intended to minimize the Euclidean distance between desired and predicted trajectory at time step t_i , denoted as

$$align(p_d(t_i), p_p(t_i)) = \| \, p_d(t_i) - p_p(t_i) \, \|_2 \tag{5.3.3}$$

As depicted in Figure 5.3, the interface between global and local layer is formulated. For each velocity pair (u, r) within the scope of resulting search space, DW algorithm generates a corresponding predicted trajectory. Path alignment term $align(p_d(t_i), p_p(t_i))$ is feedback to the modified DW algorithm, further determine the optimal velocity pair that minimizes objective function, considered as steering command.

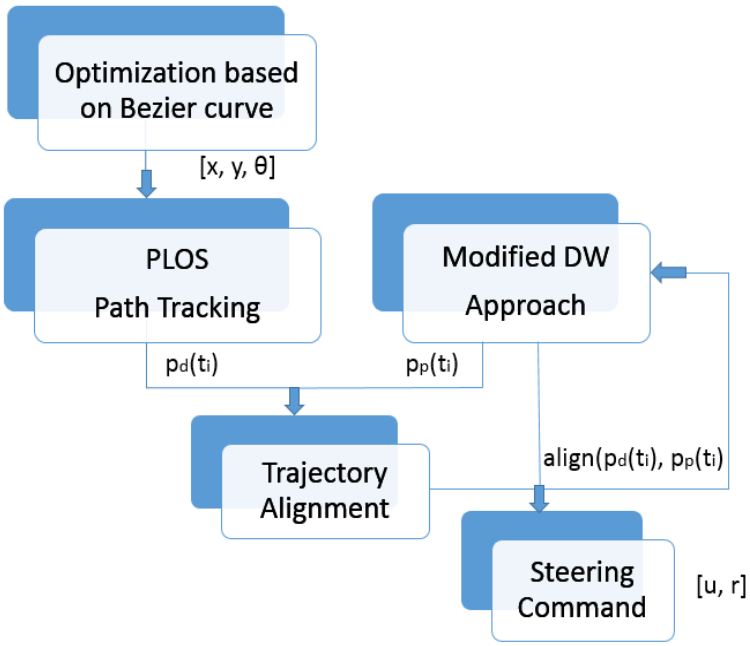

Figure 5.3: Interface between deliberate and reactive method with PLOS algorithm

The principle of selecting optimal velocity pair can be expressed as

$$(u, r) = \underset{(u,r) \in V_r}{\arg\min} \, G(u, r) \tag{5.3.4}$$

where V_r denotes the resulting search space consisting of admissible velocities with constraints. As the resulting search space V_r is an intersection of three restricted velocity sets,

which is a continuous space, Based on the assumption that obstacles move with constant speed or small variation such that collision won't occur during short time interval, we therefore approximate it by discretizing the space V_r into n velocity pairs with a small velocity interval, Further, trajectory in terms of each discrete velocity pair is predicted, and path alignment is added to evaluate the objective function. Eventually, optimal velocity pair (u, r) considered as the output of the algorithm, is selected by utilizing the "argmin" function.

The weight factors α, β, γ and δ determine how the algorithm favors aligning with the desired trajectory as much as possible or keeping a fair distance to any obstacles in the vicinity to guarantee a collision-free trajectory. As constraints of static obstacles have also been considered in deliberate path planning, the reactive trajectory is able to almost perfectly follow the desired path if moving obstacles are not involved, as depicted in Figure 5.4.

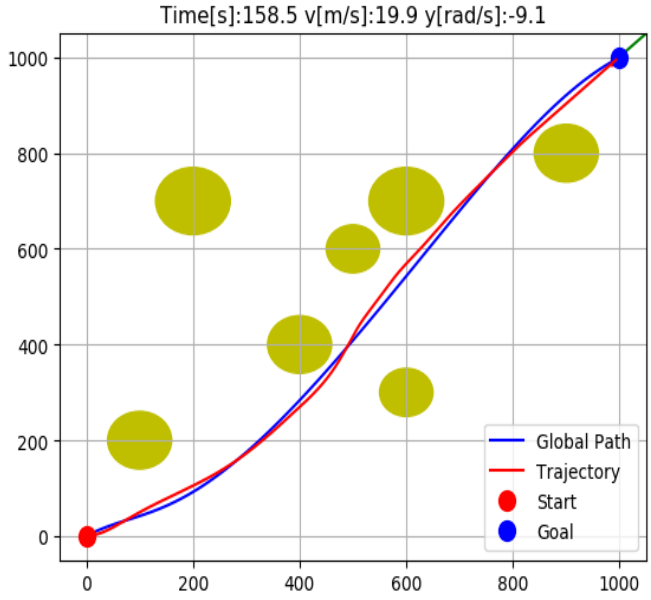

Figure 5.4: Reactive trajectory aligns well with global path merely considering static obstacles

In the case of avoiding collision with dynamic obstacles, the gain in terms of obstacle clearance function β should be tuned bigger such that the vehicle is able to avoid when a moving obstacle emerges in the vicinity. As a consequence, the hybrid algorithm tends to make compromise, giving up perfectly tracking the desired global path, and the practical trajectory may deviate from the global path to a certain extent, as shown in Figure 5.5.

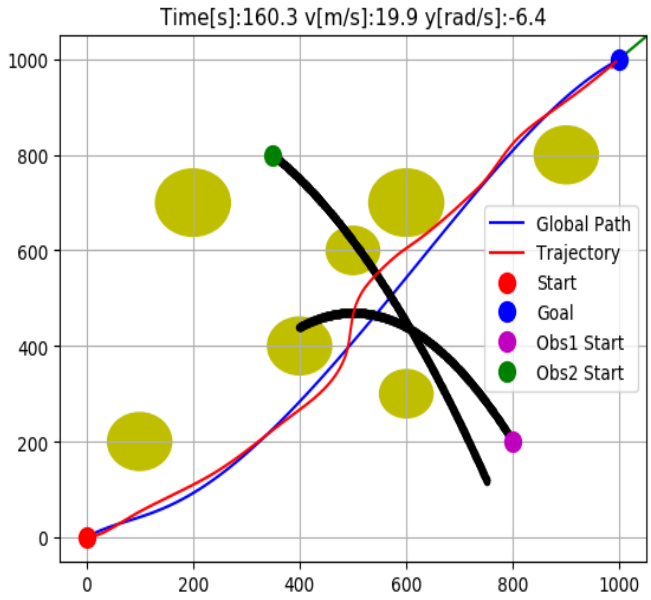

Figure 5.5: Reactive trajectory aligns well with global path while avoiding moving obstacles

Chapter 6

Simulation and Results

In this chapter, several numerical simulations are implemented in Python to evaluate the performance and efficacy of proposed algorithms, including deliberate path planning algorithm based on Bézier spline, dynamic window (DW) and hybrid COLAV algorithm. Besides, comparisons among these algorithms are presented to highlight the pros and cons of each method in different scenarios.

6.1 Simulation Implementation

Viknes 1030 is designed with good maneuverability, as shown in Figure 6.1, that is used as the vessel model in this thesis, and specifications of this ASV are presented in Table 6.1.

Figure 6.1: Viknes 1030 (Viknes, 2019)

Parameter	Value
Length	10.96 m
Width	3.42 m
Draught	1.15 m
Weight	5975 kg

Table 6.1: Basic specifications of Viknes 1030

To implement the simulation, relevant configuration parameters for each algorithm should be listed, specifically, parameters used in dynamic window algorithm simulations are given in Table 6.2, and initialization parameters defined for Bézier curve algorithm are listed in Table 6.3. In particular, the performance of the algorithm greatly depends on the tuning, thus it is essential to test the simulation through multiple debugging and eventually assign appropriate values to tuning parameters.

Parameter	Value	Comment
u_{max}	10 m/s	Maximum surge speed of ASV
u_{min}	0 m/s	Minimum surge speed of ASV
r_{max}	$15°/s$	Maximum yaw rate of ASV
r_{min}	$-15°/s$	Minimum yaw rate of ASV
acc_{max}	$2\ m/s^2$	Maximum acceleration of ASV
Δu	0.1 m/s	Interval of surge speed
Δr	$1.0°/s$	Interval of yaw rate
Δt	0.1 s	Sampling time interval
α	1.0	Weight of yaw rate term in cost function
β	15.0	Weight of distance term in cost function
δ	0.8	Weight of velocity term in cost function
γ	10.0	Weight of alignment term in cost function

Table 6.2: Parameters used in simulation for dynamic window algorithm

Parameter	Value	Parameter	Value
Initial pose	[0, 0, 30°]	Final pose	[500, 500, 30°]
Degree of Bézier curve	5	Number of obstacles	7
Min turning radius	100 m	Min, Max radii of obstacles	30, 50

Table 6.3: Parameters used in simulation for global path planning based on Bézier curves

Hybrid COLAV algorithm is implemented based on the integration of global path planning and dynamic window algorithm. At the beginning of every simulation, the global method together with the PLOS path tracking algorithm are devoted to generate global trajectory as a guidance for reactive COLAV algorithm. With the knowledge of the desired pose at each time step, predicted trajectory of discrete velocity pair $[u, r]$ in the search space

is generated, after that, the optimal velocity pair is finally determined by minimizing the objective function, in order to drive the vehicle towards the goal while avoiding collision with both static and moving obstacles. It's remarkable that the task of collision avoidance takes precedence over reaching the goal to guarantee safety of the vehicle.

6.2 Scenarios and Results

In this section, several simulation scenarios in terms of different situations and algorithms are presented. A couple of assessment criteria used to evaluate the performance and robustness of algorithms based on final results, are listed as follows.

- Capability of leading the vehicle to goal;
- Response to unexpected moving obstacles;
- Behaviour when encountering local minima;
- Minimum obstacle clearance.

Scenario 1

In the first scenario, the global path planning based on Bézier curve is tested, given initial and final pose of the vehicle, together with the constraints of static obstacles, approximated as several circles with different radii, ranging from 30m to 50m.

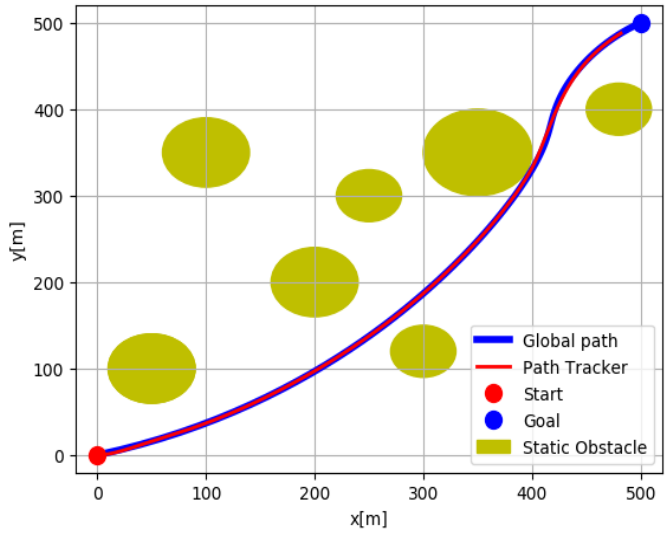

Figure 6.2: Trajectory generated by PLOS following the global path

As depicted in Figure 6.2, the deliberate algorithm is capable of generating a feasible and collision-free global path towards goal under the constraints of static obstacles. PLOS path tracker is adopted and implemented to convert the desired path into the trajectory, that is further applied to local DW algorithm. The corresponding result reveals that PLOS algorithm is able to generate a trajectory perfectly following the global path with insignificant errors, particularly, constraints are imposed on the maximum velocity and yaw rate of the vehicle. As shown in Figure 6.3, the speed keeps rising with a proportion gain until it reaches the max velocity $v_{max} = 10m/s$, while the yaw rate varies over time within the restricted region.

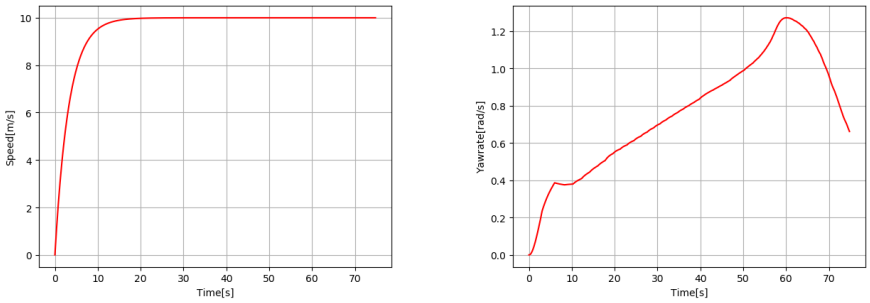

Figure 6.3: Speed and yaw rate of the trajectory generated by path tracker

The main objective of hybrid COLAV method is to generate a trajectory relies on velocity pair sequence, aligning well with the global path and simultaneously avoiding static and moving obstacles. To note that the hybrid algorithm always gives priority to collision avoidance rather than follows the global path to ensure a collision-free trajectory, which may deviate from the global path in order to avoid obstacles.

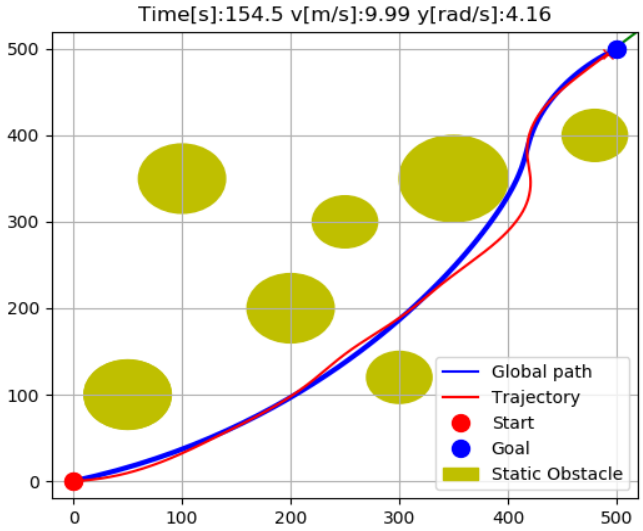

Figure 6.4: Trajectory generated by DW algorithm based on the global path

As depicted in Figure 6.4, the vehicle gives up tracking the path to keep a fair distance from the obstacle while turning around the obstacle in the vicinity of position x = 400m. As the constraints of obstacle avoidance imposed on the deliberate method is less strict, yielding the path fairly close to the obstacle, which is considered as an unacceptable risky behaviour

for reactive DW algorithm. Therefore, the simulation result in scenario 1 implies that hybrid method is able to generate a complete and collision-free trajectory in the presence of static obstacles. Besides, how the surge speed and yaw rate of ASV change over time is illustrated in Figure 6.5.

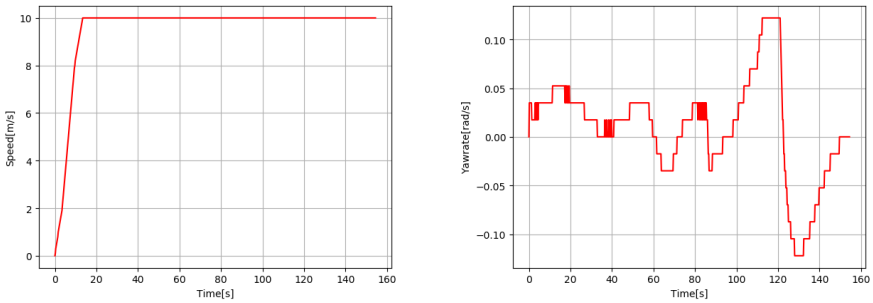

Figure 6.5: Speed and yaw rate of the trajectory generated by DW algorithm

Scenario 2

The major objective of the proposed COLAV algorithm is to avoid collision with the moving obstacles in the dynamic environment. Nevertheless, the global path generated based on Bézier curve in conjunction with optimization formulation is inadequate to handle moving obstacles due to the less responsiveness to unexpected situation. In this scenario, two moving obstacles with constant speeds and headings are considered, with parameters shown in Table 6.4. After applying the PLOS steering law to global path tracking, ASV ends up with colliding with the straight-line moving obstacle, as shown in Figure 6.6. The minimum distance between the vehicle and moving obstacles is set to be 30m, and therefore, collision will occur if the ASV moves into the collision range. The failed situation indicates that it's necessary to employ reactive COLAV method to handle moving obstacles.

Table 6.4: Parameters of the moving obstacles

Parameter	Moving Obs 1	Moving Obs 2
Initial position	[200, 400]	[400, 100]
Heading angle	$-45°$	$120°$
Moving speed	3 m/s	3 m/s

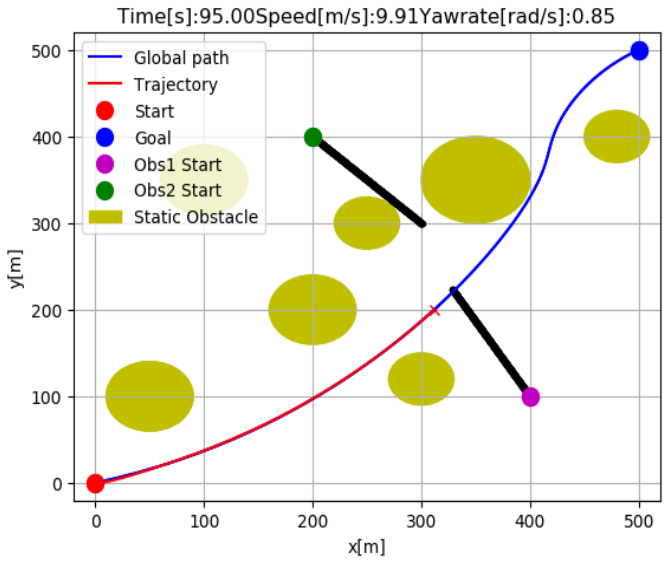

Figure 6.6: ASV collides with straight-line moving obstacles if only tracking global path

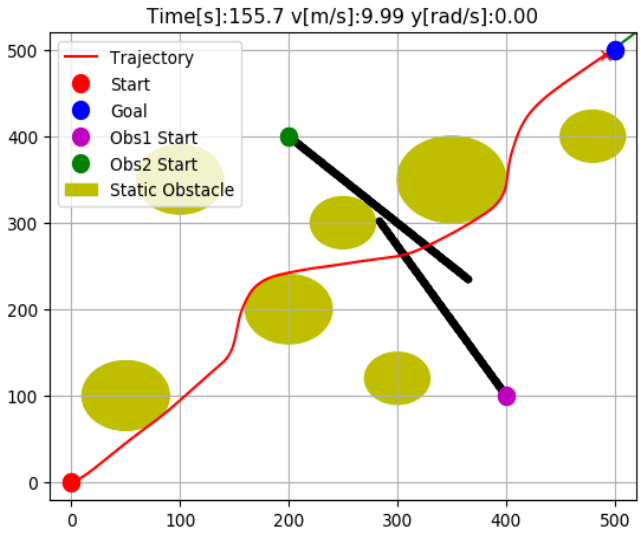

Figure 6.7: Trajectory generated by DW algorithm with straight-line moving obstacles

By contrast to global path, Figure 6.7 validates that reactive DW algorithm is more powerful in the case of avoiding collision with moving obstacles. Snapshots of the moving process show how the vehicle is able to handle moving obstacles by constantly calculating the objective function and adjusting the heading to yield a feasible and collision-free trajectory.

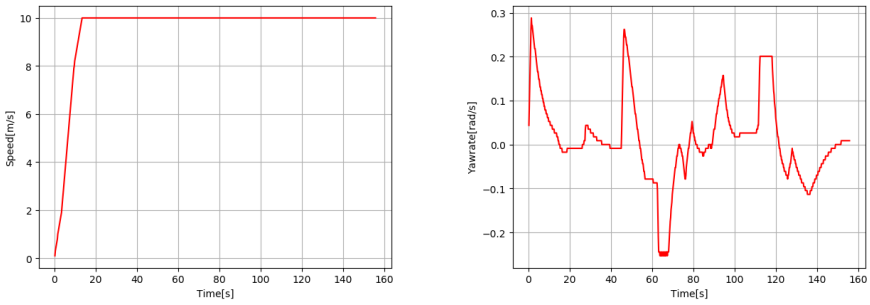

Figure 6.8: Speed and yaw rate of the trajectory in scenario 2

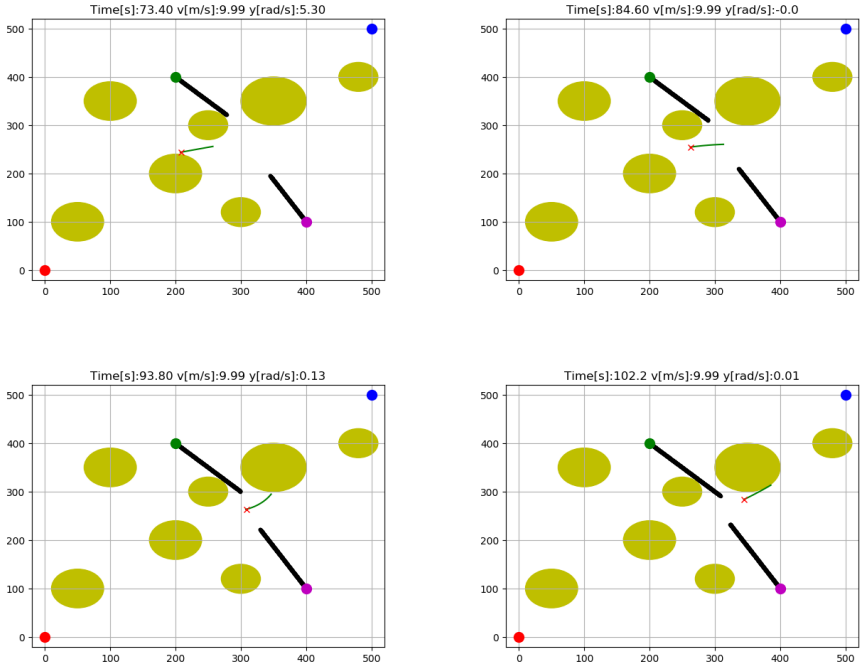

Figure 6.9: Snapshots of trajectory in scenario 2

Scenario 3

As shown in the above scenario, DW algorithm may manage to generate a collision-free trajectory leading to the given goal in some cases, but it is undeniable that it also suffers some drawbacks. Due to the downsides of local DW algorithm, high sensitivity to local minima is prone to cause failure that the ASV gets stuck and is not able to find the route towards goal. To address this issue, hybrid COLAV method is introduced. By the comparison of results shown in Figure 6.10 and 6.11, the advantage and necessity of the hybrid algorithm is demonstrated. When solely apply DW algorithm, the behaviour tends to be greedy and reach the goal as soon as possible, while ignoring the possibility of being trapped in local minima. Instead, with the guidance of the complete path from start to goal, the ASV is able to generate a trajectory towards the target avoiding local minima.

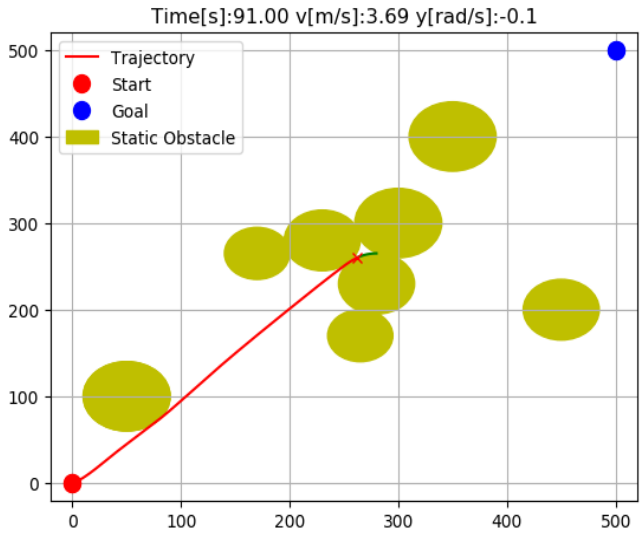

Figure 6.10: ASV trapped in local minima when only employ DW algorithm

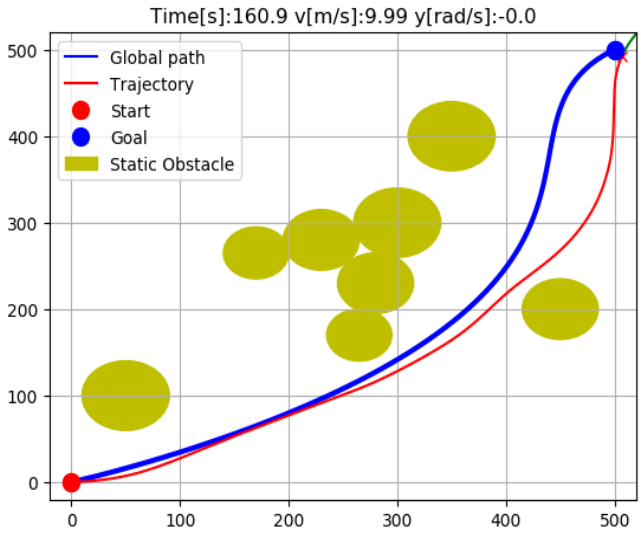

Figure 6.11: ASV avoid local minima when apply hybrid COLAV algorithm

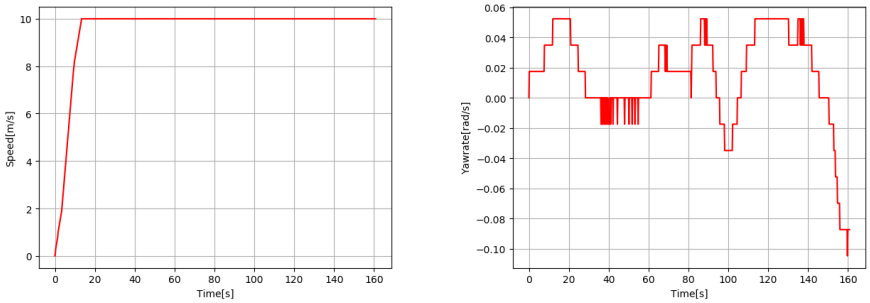

Figure 6.12: Speed and yaw rate of the trajectory in scenario 3

Scenario 4

The major intention of introducing reactive COLAV method is to take into account the unexpected situations, for instance, the occurrence of moving obstacles, such that the reactive algorithm is capable of making real-time response to cope with the rapidly-changing environment. In scenario 4, two moving obstacles with constant heading angle and velocity are considered, In other words, the trajectory of either moving obstacle is a straight line, besides, the specific information of obstacles are listed in Table 6.5.

Table 6.5: Parameters of the moving obstacles

Parameter	Moving Obs 1	Moving Obs 2
Initial position	[200, 400]	[400, 100]
Heading angle	$-45°$	$120°$
Moving speed	3 m/s	3 m/s

As shown in Figure 6.13, the final trajectory of ASV reveals that vehicle is able to follow the global path when there is no threat, while significantly deviating from the global path in the middle section to stay clear of the moving obstacle. The accepted minimum distance from the vehicle to each moving obstacle is defined to be 30 m, that is, the ASV gets into the collision when it's at a distance less than 30m from the obstacle.

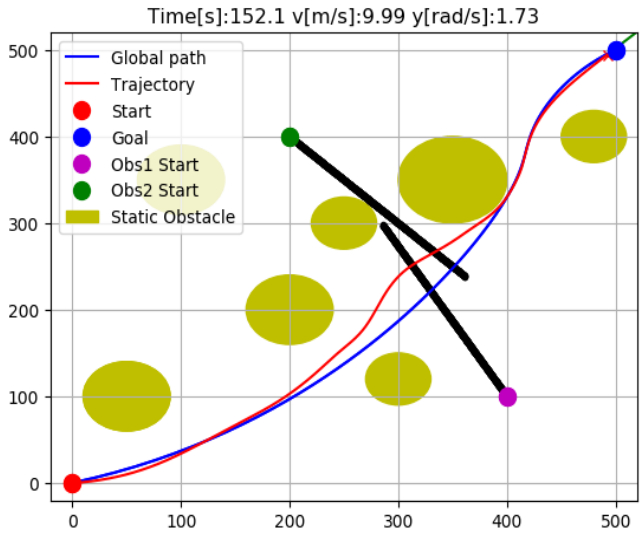

Figure 6.13: Trajectory generated by hybrid COLAV algorithm with straight-line moving obstacles

To further illustrate the process of collision avoidance, several snapshots are shown in Figure 6.14. The first snapshot implies the trend of moving away from the global path, since the ASV has predicted that two moving obstacles are approaching in opposite directions. To stay clear of two obstacles at the same time, the vehicle has to find a route traveling through a narrow passage, such that it could keep a relative safe distance to both moving obstacles. After getting rid of the moving obstacles, the path alignment takes precedence as soon as the ASV enters the safe region. As a consequence, the ASV re-follows the global path leading towards the goal as shown in the fourth snapshot.

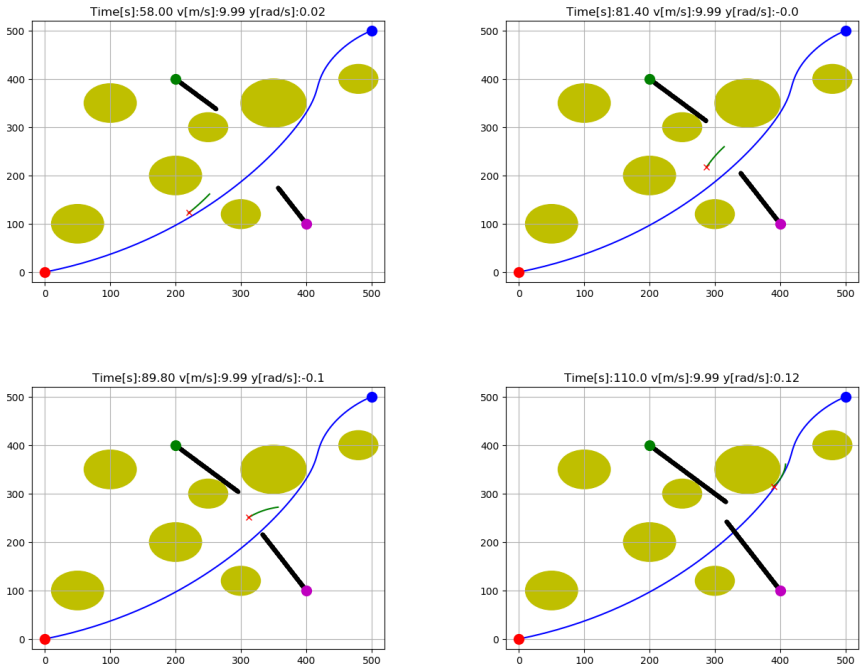

Figure 6.14: Snapshots of trajectory in scenario 4

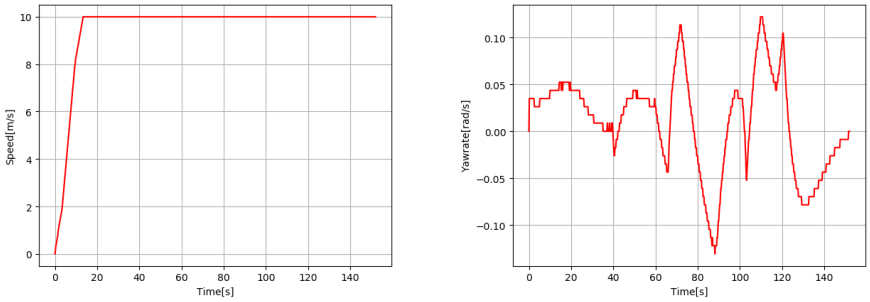

Figure 6.15: Speed and yaw rate of the trajectory in scenario 4

Scenario 5

In this scenario, moving obstacles with varying heading and velocity leading to circular-arc trajectories, are taken into consideration.

Table 6.6: Parameters of the moving obstacles

Parameter	Moving Obs 1	Moving Obs 2
Initial position	[150, 300]	[450, 100]
Equation of motion	y = -0.006x^2 + 1.8x +165	-0.01x^2 + 6.5x - 800
Moving speed	3 m/s	-1.5 m/s

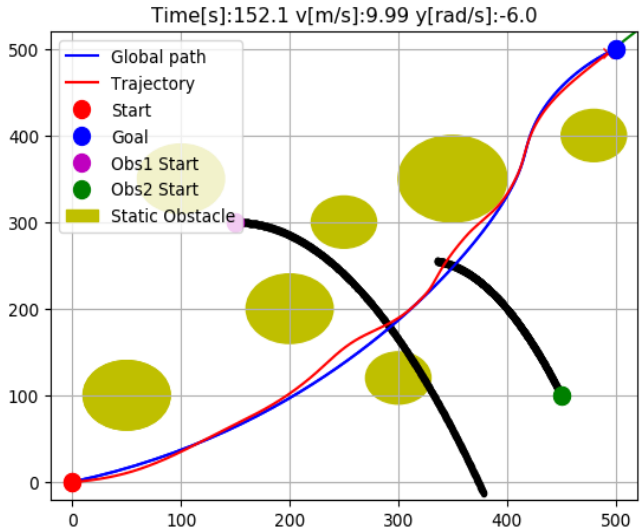

Figure 6.16: Trajectory generated by hybrid COLAV algorithm with circular-arc moving obstacles

A set of snapshots are presented as Figure 6.17, making a clear explanation of the moving process. The vehicle deviates from the global path to avoid the first moving obstacle emerging in the vicinity by changing the yaw rate, and it starts to catches up with the global path after entering the safe region. After tracking the path for a short distance, the occurrence of the second obstacle steers the vehicle off the track again. Further, the vehicle changes its heading to re-follow the path as soon as it gets rid of the obstacle.

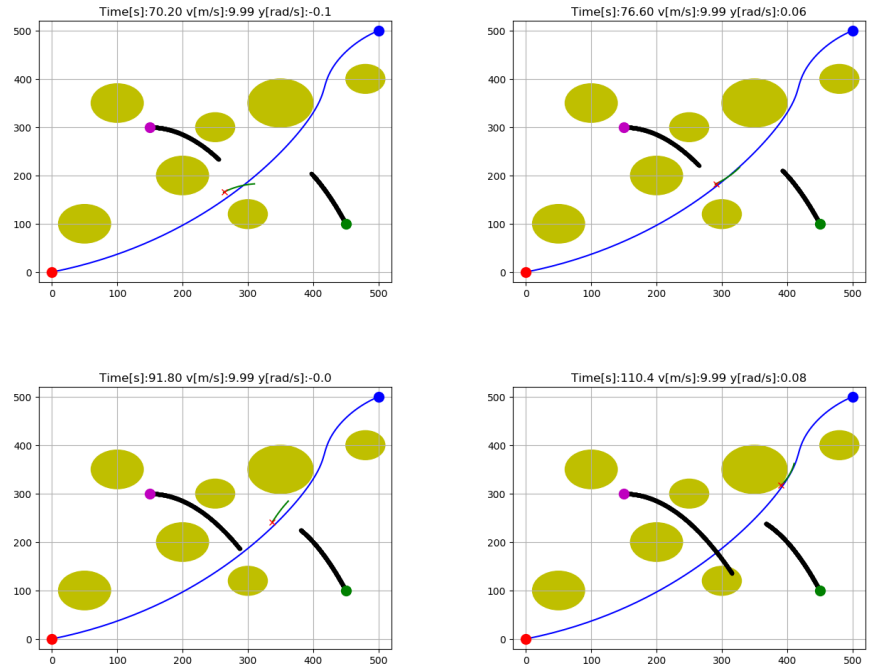

Figure 6.17: Snapshots of trajectory in scenario 5

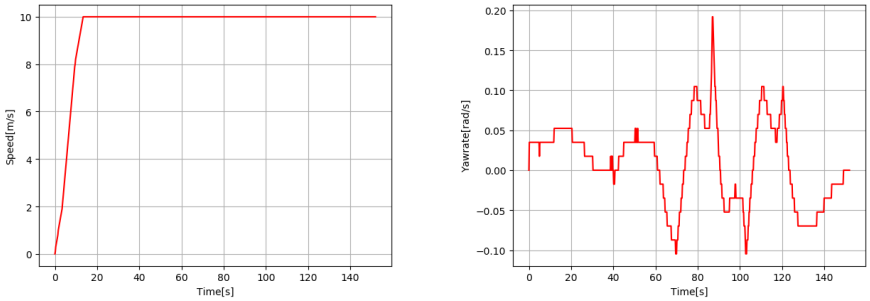

Figure 6.18: Speed and yaw rate of the trajectory in scenario 5

Scenario 6

In scenario 6, unknown dynamic obstacles with random trajectory are employed to evaluate the robustness of the hybrid COLAV algorithm. The unpredictable and rapidly-varying motion trends have made the collision avoidance task more challenging, demanding for more responsive performance. As depicted in Figure 6.19, the hybrid algorithm is still able to generate a collision-free trajectory almost coincided with the desired global path.

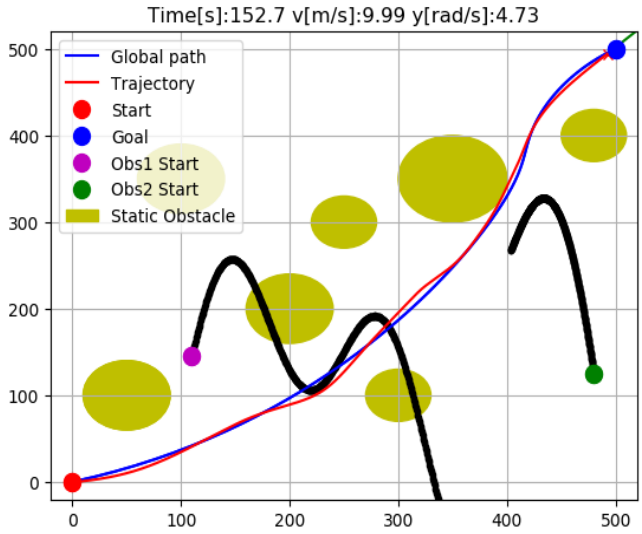

Figure 6.19: Trajectory generated by hybrid COLAV algorithm with random moving obstacles

The snapshots at different time shown in Figure 6.20, have revealed more details about how the vehicle stays clear of the moving obstacles by constantly adjusting its heading angle. When the vehicle is approaching the second moving obstacle, it is surrounded by static and moving obstacles on both sides. Compromise has been made to guarantee the safety by giving up keeping a fair distance to the circle of the static obstacle. As a consequence, the ASV takes the potential risk of running into the static obstacle to achieve collision avoidance with moving obstacle.

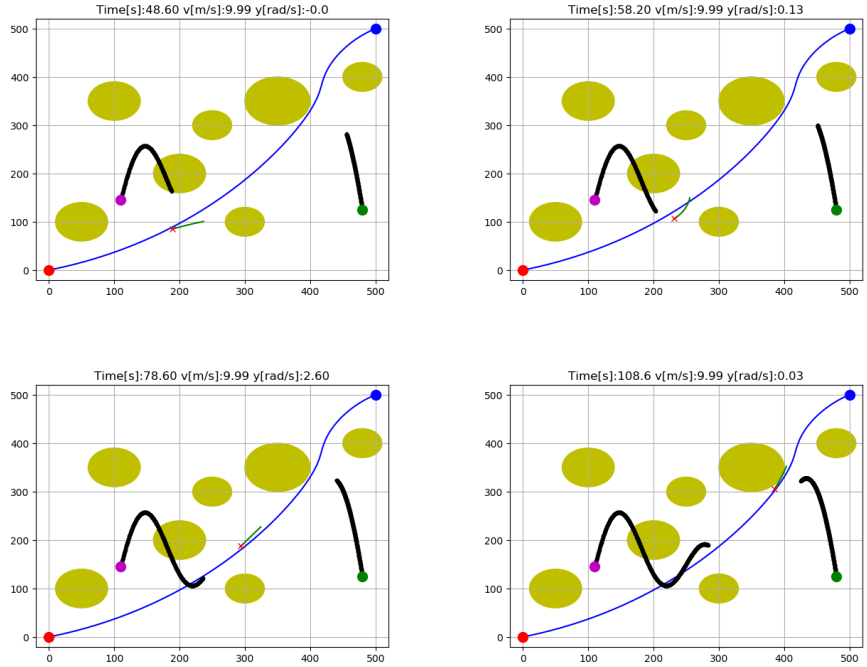

Figure 6.20: Snapshots of trajectory in scenario 6

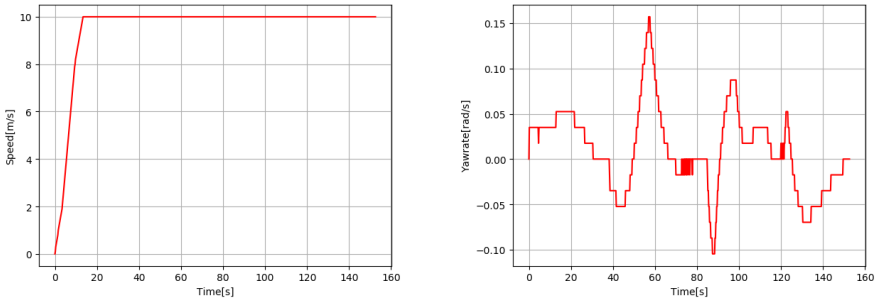

Figure 6.21: Speed and yaw rate of the trajectory in scenario 6

Scenario 7

To evaluate the robustness of this COLAV algorithm, Gaussian noise is added to the measured position and velocity of the moving obstacles. In this scenario, the COLAV algorithm only has access to noisy measurements with different standard deviation. Due to the robustness of the hybrid COLAV algorithm, the vehicle is still capable of generating a feasible and collision-free trajectory until the noise is increased to an unaccepted value. And the tolerance limit to noisy measurement is tested by increasing the value of standard deviation σ.

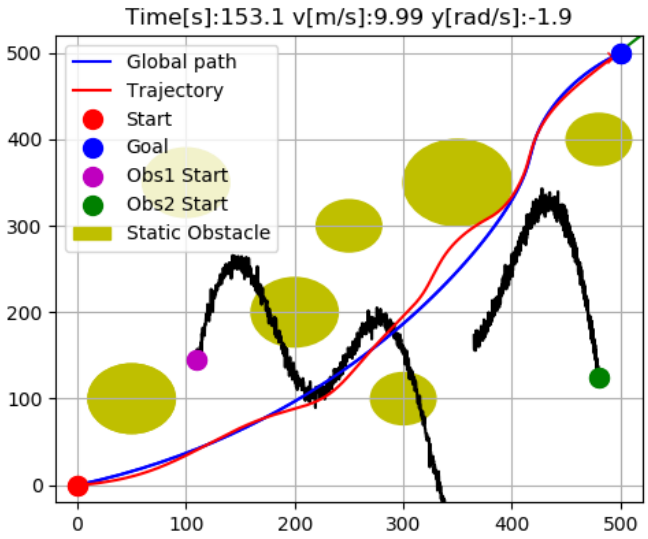

Figure 6.22: Trajectory generated by hybrid COLAV algorithm with Gaussian noise $\sigma = 6$

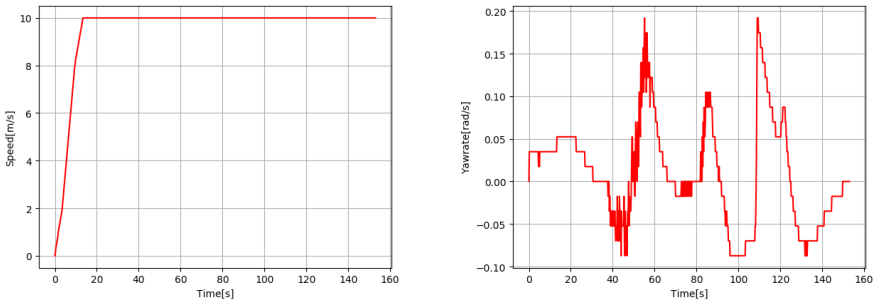

Figure 6.23: Speed and yaw rate of the trajectory in scenario 5

As shown in results including Gaussian noise with standard deviation σ of 6, 10, and 15, the vehicle is able to follow the global path and avoid random obstacles involving Gaussian noise with σ = 6 and 10, while it fails to proceed when the σ is set to 15.

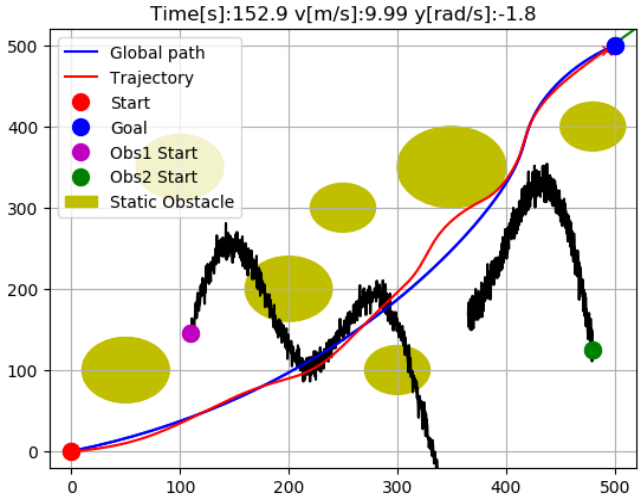

Figure 6.24: Trajectory generated by hybrid COLAV algorithm with Gaussian noise σ = 10

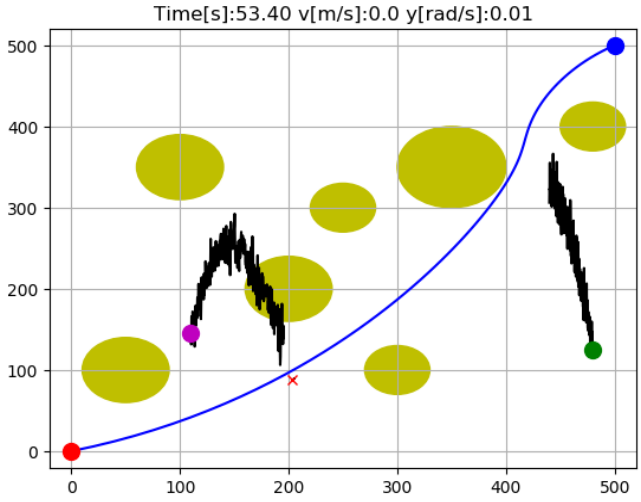

Figure 6.25: Trajectory generated by hybrid COLAV algorithm with Gaussian noise σ = 15

Snapshots shown in Figure 6.26, implies that the presence of Gaussian noise on measurement will disturb motion and eventually affect the trajectory of ASV.

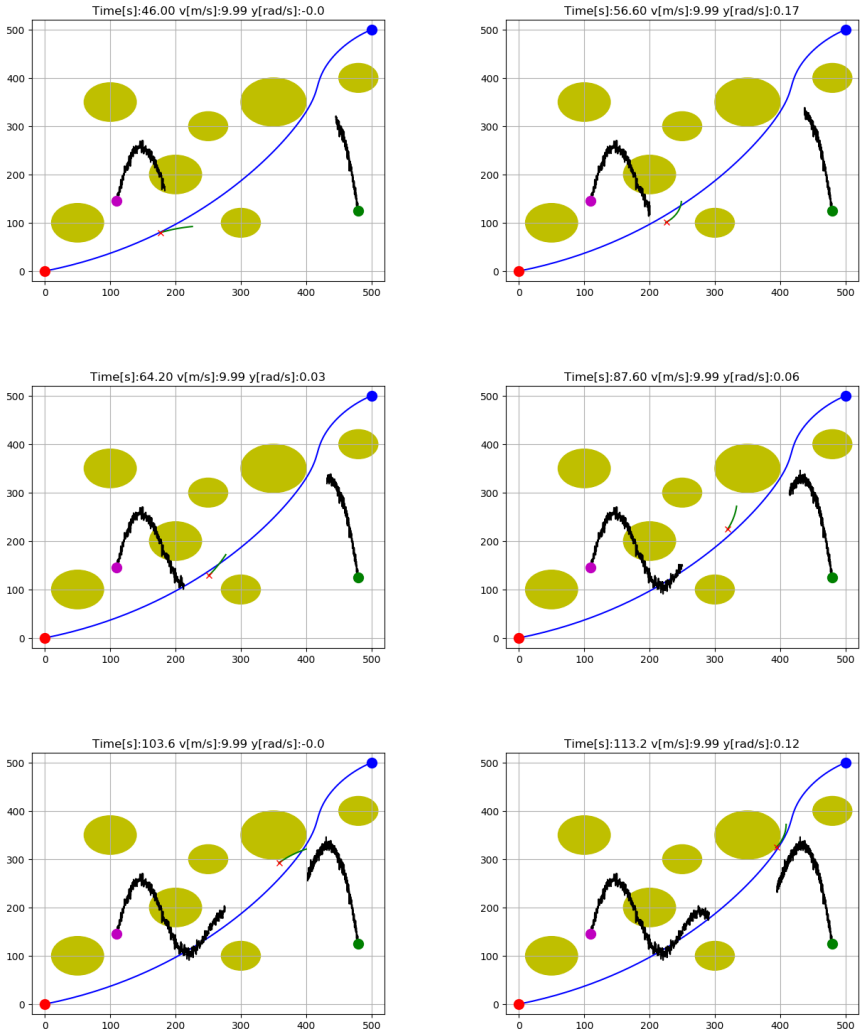

Figure 6.26: Snapshots of trajectory in scenario 7

Scenario	Type	Performance
1	Static obstacles	Both reactive and deliberate algorithms are capable of generating collision-free trajectory by keeping a fair distance larger than the corresponding radius to the center of every static obstacle.
2	Straight-line moving obstacles	The global path planning algorithm fails to handle moving obstacles, while the reactive DW approach is capable of generating a collision-free trajectory.
3	Local minima	Adopting only reactive DW algorithm causes the consequence of being trapped in the local minima, while this issue is easily addressed by introducing the hybrid method.
4	Straight-line moving obstacles	Hybrid COLAV method based on global path is adopted when including obstacles moving with constant heading and velocity. ASV is able to avoid collision with both static and moving obstacles.
5	Circular-arc moving obstacles	When considering moving obstacles with varying heading angle, ASV deviates the global path to stay clear of the moving obstacles and re-follows the path after it gets rid of moving obstacles.
6	Random moving obstacles	Moving obstacles with rapidly-changing motion trends make collision avoidance more challenging. However, ASV still manages to generate a collision-free trajectory aligning well with the global path.
7	Random moving obstacles with Gaussian noise	Gaussian noise with different standard deviations are incorporated to test the robustness and tolerance limits of the hybrid algorithm.

Table 6.7: Overview of different simulation scenarios

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, a hybrid COLAV method based on Bézier curves and dynamic window algorithm is introduced. PLOS path tracking algorithm is exploited to track the global path and extensively contribute to developing the interface between deliberate and reactive COLAV method. Furthermore, the feasibility and robustness of the algorithm are analysed regarding different scenarios through numerical simulations.

Global path planning method based on Bézier curves within the formulation of optimization is introduced, in which Bézier curves is used to generate a set of path, while static obstacles are regarded as constraints. Property of differential flatness is employed to incorporate the dynamics of the vehicle by assigning cost to each path, and the optimal path is selected by minimizing the objective function while satisfying the constraints. Further, global path is transformed into planned trajectory used as guidance for reactive COLAV method through PLOS path tracking algorithm.

As a reactive collision avoidance approach, dynamic window searches for commands in the space of velocity, consists of surge speed and yaw rate for ASV. Original dynamic window algorithm (Fox et al., 1997) has been modified to adapt to the newly proposed hybrid method by incorporating a path alignment function, which is simply the difference value of planned and predicted trajectory generated by global path planning and reactive DW algorithm, at each time step. Besides, the proposed hybrid algorithm has been extended to account for the dynamics of the vehicle.

From the results presented in Chapter 6, the hybrid COLAV method has been proven to work well handling both static and dynamic obstacles. By taking advantage of deliberate and reactive methods, the proposed hybrid algorithm manages to generate a collision-free trajectory towards the goal in all scenarios considered in this thesis, without being trapped in the local minima or running into unexpected moving obstacles. To test the robustness

of the hybrid algorithm, Gaussian noise is included to make only noisy measurements accessible to the ASV. However, it should be further investigated to promote compatibility to the International Regulations for Preventing Collisions At Sea (COLREG).

7.2 Future Work

Future work should aim to investigate the extensions and modifications that can be carried out to improve the developed hybrid COLAV method, mainly focused on the following topics:

- Adapt the algorithm to be COLREG compliant.
- Adopt a more predictive approach (MPC) to facilitate collision avoidance in guidance system.
- To further investigate the robustness of the system regarding the model uncertainties.
- Adjust the algorithm to meet demands for full-scale experiments.

Appendix

Appendix A: Conference Paper

The conference paper submitted to Joint CAMS and WROCO 2019 can be viewed on the following pages.

Hybrid Collision Avoidance with Moving Obstacles

Yi Chai * **Vahid Hassani** *,**

* *Centre for autonomous marine operations and systems (AMOS), Dept. of Marine Technology, Norwegian Univ. of Science and Technology, Trondheim, Norway.*
** *Department of Ships and Ocean Structures, SINTEF Ocean, Trondheim, Norway.*

Abstract: This paper proposes a hybrid collision avoidance (COLAV) approach based on the integration of a global path planning algorithm and a reactive collision avoidance approach to address the COLAV issue with the presence of the moving obstacles. Bézier curves are exploited as the basis for global path planning, while dynamic window (DW) algorithm is employed to search for optimal velocity pairs which ensure collision-free trajectory. In particular, the interface between the deliberate and reactive method is developed, enabling the vehicle to simultaneously track the generated global path towards the goal and avoid local collision. The performance and robustness of this hybrid COLAV method have been evaluated through numerical simulations.

Keywords: Bézier curve, path planning, dynamic window, collision avoidance

1. INTRODUCTION

A considerable amount of work has been done in the field of autonomous vehicles and collision avoidance (COLAV) over the past few decades. Autonomous path planning and collision avoidance are essential for Autonomous Surface Vehicles (ASV), navigating in unknown or partially known environment with static and moving obstacles in the vicinity of the vehicle. The hybrid COLAV architecture proposed in this article, decomposes the task into global path planning and local collision avoidance.

Reactive COLAV methods are widely used due to the low demand for computing capabilities. Velocity Obstacles method is one of those reactive COLAV approaches, intended for motion planning to avoid static and moving obstacles in the velocity space (Fiorini and Shiller, 1998). (Ge and Cui, 2002) proposed a new potential field method for motion planning of mobile robots in a dynamic environment with moving target and obstacles. Additionally, dynamic window algorithm is one of the existing reactive COLAV approach, originally designed for robot with first order nonholonomic constraints (Fox et al., 1997). A modified DW algorithm presented in (Eriksen et al., 2016), is adapted and tested for autonomous underwater vehicles (AUV) with second-order nonholonomic constraints.

Nevertheless, dynamic window algorithm suffers from many drawbacks, and the most significant one is high sensitivity to the local minima. (Seder and Petrovic, 2007) proposes an improved dynamic window algorithm incorporated with a focused D* search algorithm, such that the vehicle is less likely to be trapped in a local minima. Furthermore, (Serigstad et al., 2018) introduces a hybrid dynamic window approach, functions as an interface to any deliberate COLAV method which generates time parameterized trajectories, enabling vehicles to avoid local minima.

Motivated by the above considerations, in this paper, a hybrid COLAV architecture is presented, based on the combination of global pre-defined path generated by Bézier curves and dynamic window algorithm. Furthermore, interface between these two methods is developed, steering the vehicle to track the global path while avoiding both static and moving obstacles.

The global path planning is carried out using a new generation of path planning that incorporates in its formulation the dynamics of the vehicles and extra data made available by on board sensors about obstacles and other vehicles in vicinity (Hassani and Lande, 2018). Bézier Curves are used as the basis for generating a rich set of paths that determines spatial and temporal profile of the vehicles. Using differential flatness property of the vehicle, we are able to reconstruct all the states of the vehicles during the maneuver. The calculated states are then used to assign a cost function to each path that reflects the dynamic capabilities of the vehicle on that path. Hence, the global path generated by Bézier curves takes the dynamics of vehicle into account in their formulations; see (Hassani and Lande, 2018).

The rest of the article is organized as follows. Section 2 summarizes the results in (Hassani and Lande, 2018) and presents a brief introduction to the global path generator used in this article. In section 3, a short description of the Dynamic Window algorithm is presented. Section 4 describes the key idea behind the proposed hybrid COLAV technique. The performance and robustness of the proposed hybrid COLAV algorithm is evaluated through several simulation scenarios in Section 5. Conclusions and

suggestions for future research are summarized in Section 6.

2. GLOBAL PATH GENERATOR FOR FIXED OBSTACLES

This section summarized the results of (Hassani and Lande, 2018) in which, a class of Bézier curves is used to provide a rich class of potential paths. Using the flatness property of ASV, all the states and inputs of the ship along the path is computed from which a cost value can be assigned to each candidate path. Finally, an optimization problem is formulated that would give birth to a global path generator that would generate a path from point A to point B in presence of fixed obstacles. the calculated path satisfies dynamic limitations of the ASV such as required curvature, continuity, smoothness.

2.1 Bézier curve

The mathematical basis for the Bézier curve are the Bernstein polynomials, named after the Russian mathematician Sergei Natanovich Bernstein (Farin, 2014). In 1912 the Bernstein polynomials were first introduced and published as a means to constructively prove the Weierstrass theorem. In other words, as the ability of polynomials to approximate any continuous function, to any desired accuracy over a given interval. The slow convergence rate and the technological challenges in the construction of the polynomials at the time of publication, led to the Bernstein polynomial basis being seldom used for several decades to come. Around the 1960s, independently, two French automobile engineers of different companies, started searching for ways of representing complex shapes, such as automobile bodies using digital computers. The motivation for finding a new way to represent free-form shapes at the time, was due to the expensive process of sculpting such shapes, which was done using clay. The first engineer concerned with this matter was Paul de Faget de Casteljau working for Citroën, who did his research in 1959. His findings lead to what is known as de Casteljau algorithm, a numerically stable method to evaluate Bézier curves. De Casteljau work were only recorded in Citroën internal documents, and remained unknown to the rest of the world for a long time. His findings are however today, a great tool for handling Bézier curves (Farin, 2014). The person who lends his name to the Bézier curves, and is principally responsible for making the curves so well known, is the engineer Pierre Étienne Bézier. Bézier worked at Renault, and published his ideas extensively during the 1960s and 1970s. Both Bézier and de Casteljau original formulations did not explicitly invoke the Bernstein basis, however the key features are unmistakably linked to it and today the Bernstein basis is a key part in the formulation (Farouki, 2012).

A Bézier curve is defined by a set of control points $\boldsymbol{P}_i$ $(i = 0 \dots n)$ for which n denotes the degree of the curve. The number of control points for a curve of degree n is $n + 1$, and the first and last control points will always be the end points of the curve. The intermediate points does not necessarily lay on the curve itself. The Bézier curve can be express on a general form as

$$\boldsymbol{P}(t) = \sum_{i=0}^{n} B_i^n(t)\boldsymbol{P}_i \quad t \in [0, 1], \tag{1}$$

where t defines a normalized time variable and $B_i^n(t)$ denotes the blending functions of the Bézier curve, which are Bernstein polynomials defined as

$$B_i^n = \binom{n}{i}(1-t)^{n-i}t^i, \quad i = 0, 1, 2..., n. \tag{2}$$

2.2 Differential flatness

A dynamic model of ASV is presented in (Hassani and Lande, 2018); furthermore, it is shown that the proposed model exhibits a differential flatness property; see (Van Nieuwstadt and Murray, 1998). A system is said to be differentially flat if one can find a set of outputs, equal in number to the number of inputs, such that one can express all states and inputs as functions of these outputs and their derivatives. This can be formulated mathematically for a nonlinear system, as follows. Consider a nonlinear system

$$\dot{x} = f(x, u) \quad x \in \mathbb{R}^n,\ u \in \mathbb{R}^m \tag{3}$$
$$y = h(x) \qquad y \in \mathbb{R}^m, \tag{4}$$

where x denotes the state vector, u denotes the control input vector and y denotes the tracking output vector.

Such a system is said to be differentially flat if there exist a vector $z \in \mathbb{R}^m$, known as the flat output, of the form

$$z = \zeta(x, u, \dot{u}, ..., u^{(r)}), \tag{5}$$

such that

$$x = \phi(y, \dot{y}, ..., y^{(q)}) \tag{6}$$
$$u = \alpha(y, \dot{y}, ..., y^{(q)}), \tag{7}$$

where ζ, ϕ and α are smooth functions.

3. DYNAMIC WINDOW ALGORITHM

Dynamic window method is a local reactive avoidance technique, searching for inputs implemented in the space of velocities. The main advantage of this approach is that it directly incorporates the dynamics of the vehicle, since the velocity space consists of translational velocity and rotational rate, which turn into surge speed u and yaw rate r for ASV, specifically. By adopting velocity space, the pruning of the search space enormously simplify the computational effort. Furthermore, the trajectory of the ASV can be approximated by a sequence of straight lines and circular arcs, and each arc is uniquely determined by the velocity tuple (u, r) with the radius $R = u/r$. For each velocity pair within the velocity space, the dynamic window algorithm is designed to predict the trajectory that velocity pair (u, r) might generate for the next n time intervals. Then, we only consider the first time interval and assume that velocity vector remains unchanged within the remaining n-1 time intervals. This assumption is based on the observation that search is automatically repeated after each time interval, while velocity will remain constant if there are no new commands.

3.1 Search Space

With the constraints imposed on the velocity space, the resulting search space is the intersection of three restricted velocity sets, namely, the set of possible velocities V_s, admissible velocities V_a and dynamic window V_d. The set of possible velocities is limited by the extreme value of the surge speed u and yaw rate r, which is defined as

$$V_s = \{(u,r)|u \in [0, u_{max}] \wedge r \in [-r_{max}, r_{max}]\}. \quad (8)$$

Due to the kinematic and dynamic constraints, the search space is reduced to a certain span around the current velocity, which only consists of reachable velocities within the next time interval. Thus, the dynamic window can be described as

$$\begin{aligned} V_d = \{(u,r)|u_c \in [u - \dot{u_b} \cdot \Delta t, u_c + \dot{u_a} \cdot \Delta t] \\ \wedge \quad \omega \in [r_c - \dot{r_b} \cdot \Delta t, r_c + \dot{r_a} \cdot \Delta t]\}, \end{aligned} \quad (9)$$

where accelerations u_a and r_a are maximal translational and rotational accelerations, while u_b and r_b are maximal breakage decelerations. Terms u_c, r_c are current surge speed and yawrate.

The existence of obstacles in the vicinity imposes restrictions on the velocity pairs. The velocity is considered admissible if the vehicle is able to move to the next point before it hits the next obstacle on the predicted trajectory. As a consequence, the search space is reduced to a set of velocities that allow the vehicle to move without colliding with any obstacle, which can be defined as

$$\begin{aligned} V_a = \{(u,r)|u \leq \sqrt{2 \cdot dist(u,r) \cdot \dot{u_b}} \\ \wedge \quad r \leq \sqrt{2 \cdot dist(u,r) \cdot \dot{r_b}}\}, \end{aligned} \quad (10)$$

where $dist(u,r)$ represents the distance to the closest obstacle on the corresponding trajectory.

3.2 Objective Function

Among those velocity pairs within the resulting search space V_r, velocity vector (u,r) is chosen to maximize a certain objective function, which consists of some criteria, like target heading, clearance and velocity.

$$\begin{aligned} &G(u,r) = \alpha \cdot goal(u,r) + \beta \cdot dist(u,r) + \gamma \cdot vel(u,r) \\ &s.t.(u,r) \in V_r, \end{aligned} \quad (11)$$

where the terms goal(u, r), dist(u, r) and vel(u, r) are weighted by the factors α, β and γ. The terms involved in the objective function can be denoted as,

$$goal(u,r) = arccos(\frac{\overrightarrow{OA} \cdot \overrightarrow{OB}}{|\overrightarrow{OA}| \cdot |\overrightarrow{OB}|}), \quad (12)$$

$$dist(u,r) = \frac{1}{r_{min}}, \quad (13)$$

$$vel(u,r) = u_{max} - u_c. \quad (14)$$

Trajectory of the vehicle can be calculated with the velocity pairs (u, r), which implies the position is given at each time step. The term goal(u, r) is used to measure the progress towards the target, mathematically denoted as the angle between the vector pointing to goal and vector connecting start point and current position. r_{min} is referred to the distance from current position to the nearest obstacle, and the distance function dist(u, r) will reach a maximum value when obstacle occurs in the vicinity. The velocity term vel(u, r) is the difference value between maximal surge speed and the current one, which means vel(u, r) is exclusively dependent on surge speed u.

4. ADAPTIONS FOR HYBRID COLAV

As a reactive COLAV approach, dynamic window algorithm is restricted in many ways. The main drawback is that the vehicle may suffer from the risk of getting stuck in local minima and being unable to reach the goal, even though an exact path leading to the goal exists. Hence, it becomes necessary to employ a global path generated by Bézier curves, as a guidance for dynamic window algorithm. Based on the proposed deliberate and reactive COLAV methods, it's essential to develop the interface between global path planning and local collision avoidance algorithm.

4.1 Pure Pursuit Path Tracking Algorithm

To incorporate global pre-defined path generated by Bé zier curves with dynamic window algorithm, a path tracking algorithm is obliged to be adopted. Pure pursuit path tracking algorithm (Coulter, 1992) has been widely used as a steering controller for autonomous vehicles. (Yamasaki et al., 2009) proposes a robust path-following for UAV using pure pursuit guidance algorithm. (Rankin et al., 1998) presents a review and evaluation of PID, pure pursuit, and weighted steering controller for an autonomous land vehicle.

The major objective of this method is to calculates curvatures enabling the vehicle to chase a moving target point that is some distance ahead of it on the pre-planned path. The chord length of the arc represents the look-ahead distance joining current position and goal point, and it's used when search for the next target point. The state of vehicle, including position and heading need to be updated after each search, and can be presented as

$$x_{i+1} = x_i + vcos\theta_i \Delta t, \quad (15)$$

$$y_{i+1} = y_i + vsin\theta_i \Delta t, \quad (16)$$

$$\theta_{i+1} = \theta_i + \omega \Delta t. \quad (17)$$

4.2 Interface between deliberate and reactive COLAV

Desired trajectory has been derived by employing pure pursuit path tracking algorithm, which can be used as a guidance for dynamic window. Hence, the interface between the deliberate and reactive method needs to be developed, enabling the vehicle to simultaneously track the generated global path towards the goal and avoid local collision. Based on the objective function presented in section 3, a new term corresponding to path alignment should be incorporated, denoted as $align(p_p, p_t)$, distance between point on pre-defined trajectory and current position determined by velocity pair (u, r) at each time step.

$$\begin{aligned} G(u,r) = &\alpha \cdot goal(u,r) + \beta \cdot dist(u,r) + \\ &\gamma \cdot vel(u,r) - \delta \cdot align(p_p, p_t) \end{aligned} \quad (18)$$

These weight factors α, β, γ and δ determine how the hybrid COLAV favors trajectory keeping, collision avoidance or aligning with the global path.

In addition, deliberate COLAV based on Bézier curves only ensures collision-free path with the presence of static obstacles, the gain in terms of obstacle clearance function β should be tuned bigger such that the vehicle is able to avoid when a moving obstacle emerges in the vicinity. As a consequence, the practical trajectory may deviate from the global path to a certain extent, presented in the following simulation section.

5. SIMULATION RESULTS

In this section, some simulation scenarios are presented to show the performance of hybrid COLAV method, including the ability of following global pre-planed path and collision avoidance. In the following scenarios, the hybrid algorithm manages to generate a trajectory from start point (0, 0) to goal point (1000, 1000) under different conditions of obstacles.

First Scenario:
As shown in Fig. 1, trajectory of vehicle aligns well with the global pre-defined path when merely considering static obstacles. The trajectory solely differs slightly when approaching a static obstacle, that indicates the prominent ability of tracking planned path.

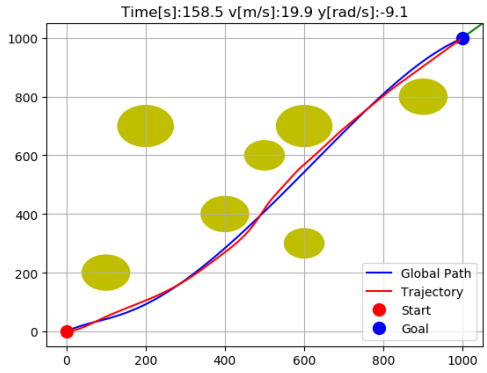

Fig. 1. DW trajectory with static obstacles

Second Scenario:
In the second scenario, moving obstacles with constant speed and heading are involved. Fig. 2 gives us a clear explication that the vehicle deviates from the global path by changing the yaw rate while a moving obstacle emerges in the vicinity and catches up with the path after entering safe region.

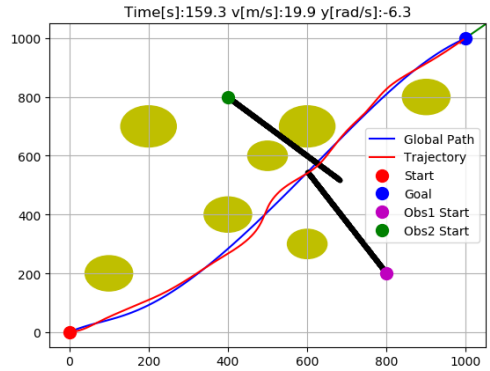

Fig. 2. DW trajectory with straight-line moving obstacles

Third Scenario:
In order to avoid moving obstacles with varying speed and heading which yield circular arc trajectory, significant offset from global path occurs for a period of time to generate a collision-free trajectory, shown in Fig. 3.

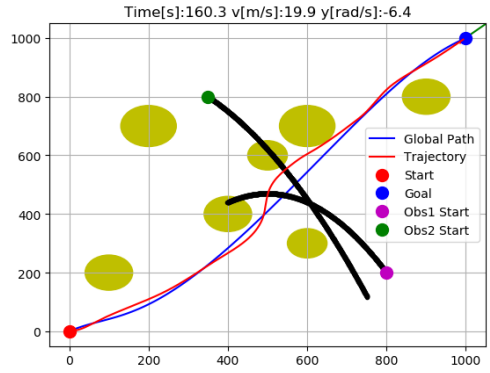

Fig. 3. DW trajectory with circular-arc moving obstacles

Fourth Scenario:
In this scenario, an unknown dynamic obstacle with random trajectory is employed to evaluate the robustness of this hybrid method. As depicted in Fig. 4, the vehicle deviates from the global path within a relatively long distance, to ensure that it stays clear of the collision region. And as soon as the vehicle keeps a fair distance to the dynamic obstacle, it re-follows the global path moving towards the target.

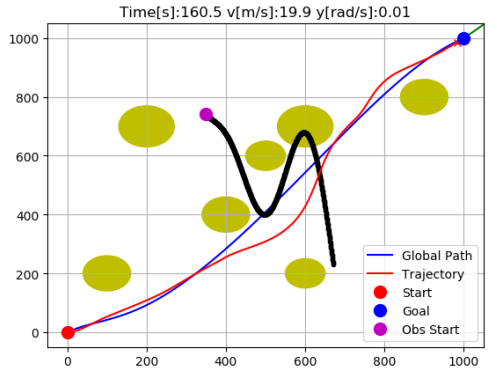

Fig. 4. DW trajectory with random shape moving obstacles

Fifth Scenario:
To evaluate the robustness of this COLAV algorithm, a Gaussian noise is added to the measured position and velocity of the obstacle. In other words, in this Scenario the COLAV algorithm only has access to noisy measurements of position and speed of the moving obstacle. Fig.5 shows the comparison of results with different standard deviation of Gaussian noise. The vehicle is still able to follow the global path and avoid random obstacle involving Gaussian noise with zero mean value and standard deviation $\sigma = 5, 10$, while it fails to proceed when the standard deviation is set to 20.

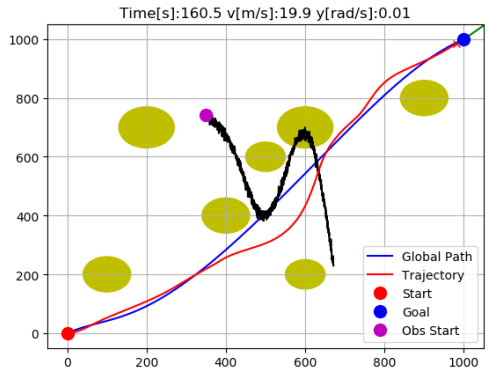

Fig. 5. DW trajectory with random moving obstacles including Gaussian noise

6. CONCLUSION

A hybrid COLAV method based on Bézier curves and dynamic window algorithm is introduced. Pure pursuit guidance is exploited to track the global path and extensively contribute to developing the interface between deliberate and reactive COLAV method. Furthermore, the feasibility and robustness of the algorithm is analysed regarding different scenarios through numerical simulations. The future work will include conforming with the International Regulations for Preventing Collisions At Sea (ColReg).

REFERENCES

Coulter, R.C. (1992). Implementation of the pure pursuit path tracking algorithm. Technical report, Carnegie-Mellon UNIV Pittsburgh PA Robotics INST.

Eriksen, B.O.H., Breivik, M., Pettersen, K.Y., and Wiig, M.S. (2016). A modified dynamic window algorithm for horizontal collision avoidance for auvs. In *2016 IEEE Conference on Control Applications (CCA)*, 499–506. IEEE.

Farin, G. (2014). *Curves and surfaces for computer-aided geometric design: a practical guide*. Elsevier.

Farouki, R.T. (2012). The bernstein polynomial basis: A centennial retrospective. *Computer Aided Geometric Design*, 29(6), 379–419.

Fiorini, P. and Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7), 760–772.

Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1), 23–33.

Ge, S.S. and Cui, Y.J. (2002). Dynamic motion planning for mobile robots using potential field method. *Autonomous robots*, 13(3), 207–222.

Hassani, V. and Lande, S.V. (2018). Path planning for marine vehicles using bezier curves. In *Proceedings of the 11th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles*, volume 51, 305–310. Elsevier.

Rankin, A.L., Crane, C.D., and Armstrong, D.G. (1998). Evaluating a pid, pure pursuit, and weighted steering controller for an autonomous land vehicle. In *Mobile Robots XII*, volume 3210, 1–13. International Society for Optics and Photonics.

Seder, M. and Petrovic, I. (2007). Dynamic window based approach to mobile robot motion control in the presence of moving obstacles. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 1986–1991. IEEE.

Serigstad, E., Eriksen, B.O.H., and Breivik, M. (2018). Hybrid collision avoidance for autonomous surface vehicles. *IFAC-PapersOnLine*, 51(29), 1–7.

Van Nieuwstadt, M.J. and Murray, R.M. (1998). Real-time trajectory generation for differentially flat systems. *International Journal of Robust and Nonlinear Control*, 8(11), 995–1020. doi:10.1002/(SICI)1099-1239(199809)8:11¡995::AID-RNC373¿3.0.CO;2-W.

Yamasaki, T., Takano, H., and Baba, Y. (2009). Robust path-following for uav using pure pursuit guidance. In *Aerial Vehicles*. IntechOpen.

Bibliography

Ahuja, N., Chuang, J.-H., 1997. Shape representation using a generalized potential field model. IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (2), 169–176.

ASV, 2019. https://www.maritimeindustries.org/write/Uploads/News/2017/Q1/ASV_Global_to_Demonstrate_C-Worker_5_Autonomous_Surface_Vehicle_at_U.S_Hydro_2017.jpg, accessed June 1, 2019.

Borenstein, J., Koren, Y., 1991. The vector field histogram-fast obstacle avoidance for mobile robots. IEEE transactions on robotics and automation 7 (3), 278–288.

Choi, J.-w., Curry, R., Elkaim, G., 2008. Path planning based on bézier curve for autonomous ground vehicles. In: Advances in Electrical and Electronics Engineering-IAENG Special Edition of the World Congress on Engineering and Computer Science 2008. IEEE, pp. 158–166.

Coulter, R. C., 1992. Implementation of the pure pursuit path tracking algorithm. Tech. rep., Carnegie-Mellon UNIV Pittsburgh PA Robotics INST.

Dakulovic, M., Horvatic, S., Petrovic, I., 2011. Complete coverage d* algorithm for path planning of a floor-cleaning mobile robot. In: 18th international federation of automatic control (IFAC) world congress. pp. 5950–5955.

Devaurs, D., Siméon, T., Cortés, J., 2016. Optimal path planning in complex cost spaces with sampling-based algorithms. IEEE Transactions on Automation Science and Engineering 13 (2), 415–424.

Eriksen, B.-O. H., Breivik, M., Pettersen, K. Y., Wiig, M. S., 2016. A modified dynamic window algorithm for horizontal collision avoidance for auvs. In: 2016 IEEE Conference on Control Applications (CCA). IEEE, pp. 499–506.

Fiorini, P., Shiller, Z., 1998. Motion planning in dynamic environments using velocity obstacles. The International Journal of Robotics Research 17 (7), 760–772.

Fossen, T. I., 2011. Handbook of marine craft hydrodynamics and motion control. John Wiley & Sons.

Fox, D., Burgard, W., Thrun, S., 1997. The dynamic window approach to collision avoidance. IEEE Robotics & Automation Magazine 4 (1), 23–33.

Fraichard, T., 1993. Dynamic trajectory planning with dynamic constraints: A'state-time space'approach. In: Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93). Vol. 2. IEEE, pp. 1393–1400.

Ge, S. S., Cui, Y. J., 2002. Dynamic motion planning for mobile robots using potential field method. Autonomous robots 13 (3), 207–222.

Goerzen, C., Kong, Z., Mettler, B., 2010. A survey of motion planning algorithms from the perspective of autonomous uav guidance. Journal of Intelligent and Robotic Systems 57 (1-4), 65.

Hart, P. E., Nilsson, N. J., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics 4 (2), 100–107.

Hassani, V., Lande, S. V., 2018. Path planning for marine vehicles using bezier curves. IFAC-PapersOnLine 51 (29), 305–310.

Häusler, A. J., Ghabcheloo, R., Pascoal, A. M., Aguiar, A. P., 2010. Multiple marine vehicle deconflicted path planning with currents and communication constraints. IFAC Proceedings Volumes 43 (16), 491–496.

Hirsch, M. W., 2012. Differential topology. Vol. 33. Springer Science & Business Media.

Kavraki, L., Svestka, P., Overmars, M. H., 1994. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. Vol. 1994. Unknown Publisher.

Khatib, O., 1986. Real-time obstacle avoidance for manipulators and mobile robots. In: Autonomous robot vehicles. Springer, pp. 396–404.

Kothari, M., Postlethwaite, I., Gu, D.-W., 2014. Uav path following in windy urban environments. Journal of Intelligent & Robotic Systems 74 (3-4), 1013–1028.

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), 110–119.

Lande, S. V., 2018. Path planning for marine vehicles using bézier curves. Master's thesis, NTNU.

LaValle, S. M., 1998. Rapidly-exploring random trees: A new tool for path planning.

LaValle, S. M., 2006. Planning algorithms. Cambridge university press.

Lopes, A., Rodrigues, J., Perdigao, J., Pires, G., Nunes, U., 2016. A new hybrid motion planner: applied in a brain-actuated robotic wheelchair. IEEE Robotics & Automation Magazine 23 (4), 82–93.

Manley, J. E., 2008. Unmanned surface vehicles, 15 years of development. In: OCEANS 2008. Ieee, pp. 1–4.

Murray, R., Rathinam, M., Sluis, W., 1995. Differential flatness of mechanical control systems. In: Proceedings of the 1995 ASME International Congress and Exposition.

Naderi, K., Rajamäki, J., Hämäläinen, P., 2015. Rt-rrt*: a real-time path planning algorithm based on rrt. In: Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games. ACM, pp. 113–118.

Nocedal, J., Wright, S., 2006. Numerical optimization. Springer Science & Business Media.

Oriolo, G., Nakamura, Y., 1991. Control of mechanical systems with second-order nonholonomic constraints: Underactuated manipulators. In: [1991] Proceedings of the 30th IEEE Conference on Decision and Control. IEEE, pp. 2398–2403.

Peng, J., Huang, Y., Luo, G., 2015. Robot path planning based on improved a* algorithm. Cybernetics and Information Technologies 15 (2), 171–180.

Radmanesh, M., Kumar, M., Guentert, P. H., Sarim, M., 2018. Overview of path-planning and obstacle avoidance algorithms for uavs: A comparative study. Unmanned Systems, 1–24.

Rankin, A. L., Crane, C. D., Armstrong, D. G., 1998. Evaluating a pid, pure pursuit, and weighted steering controller for an autonomous land vehicle. In: Mobile Robots XII. Vol. 3210. International Society for Optics and Photonics, pp. 1–13.

Seder, M., Petrovic, I., 2007. Dynamic window based approach to mobile robot motion control in the presence of moving obstacles. In: Proceedings 2007 IEEE International Conference on Robotics and Automation. IEEE, pp. 1986–1991.

Serigstad, E., Eriksen, B.-O. H., Breivik, M., 2018. Hybrid collision avoidance for autonomous surface vehicles. IFAC-PapersOnLine 51 (29), 1–7.

Škrjanc, I., Klančar, G., 2010. Optimal cooperative collision avoidance between multiple robots based on bernstein–bézier curves. Robotics and Autonomous systems 58 (1), 1–9.

Sørensen, A. J., 2012. Marine control systems propulsion and motion control of ships and ocean structures lecture notes.

Spong, M. W., Hutchinson, S., Vidyasagar, M., et al., 2006. Robot modeling and control. Vol. 3. wiley New York.

Stentz, A., et al., 1995. The focussed dˆ* algorithm for real-time replanning. In: IJCAI. Vol. 95. pp. 1652–1659.

Van den Berg, J., Lin, M., Manocha, D., 2008. Reciprocal velocity obstacles for real-time multi-agent navigation. In: 2008 IEEE International Conference on Robotics and Automation. IEEE, pp. 1928–1935.

Viknes, 2019. https://viknes.no/modell/viknes-1030/, accessed May 4, 2019.

Yamasaki, T., Takano, H., Baba, Y., 2009. Robust path-following for uav using pure pursuit guidance. In: Aerial Vehicles. IntechOpen.