ISTANBUL MEDENIYET UNIVERSITY
INSTITUE OF GRADUATE STUDIES
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

COLREGs-Compliant and Non-Prioritized Motion Planning for Autonomous Unmanned Surface Vehicles

Master Thesis

Mustafa Bayrak

August 2023

ISTANBUL MEDENIYET UNIVERSITY
INSTITUE OF GRADUATE STUDIES
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

COLREGs-Compliant and Non-Prioritized Motion Planning for Autonomous Unmanned Surface Vehicles

Master Thesis

Mustafa Bayrak

Supervisor

Asst. Prof. Haluk Bayram

August 2023

THESIS JURY APPROVAL

This Master thesis titled "COLREGs-Compliant and Non-Prioritized Motion Planning for Autonomous Unmanned Surface Vehicles" written by Mustafa Bayrak at the Department of Electrical and Electronics Engineering was accepted by our jury.

JURY MEMBERS	**SIGNATURE**
Supervisor:	
Asst. Prof. Haluk Bayram Istanbul Medeniyet University	.....
Other Jury Members:	
Asst. Prof. Filiz Gürkan Gölcük Istanbul Medeniyet University	.....
Prof. Cabir Vural Marmara University	.....

Date of Defense: 15/08/2023

STATEMENTS

Style and Reference Manual Statement

Having reviewed this thesis written under my supervision, I confirm that it has been written in accordance with APA Manual of Style and used its [footnote/in text] reference format consistently throughout the entire text.

Asst. Prof. Haluk Bayram

Declaration of Originality

I hereby declare that all information in this dissertation has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conducts, I have fully cited and referenced all material and results that are not original to this work.

Mustafa Bayrak

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor Asst. Prof. Haluk Bayram for his guidance on this thesis and throughout my Master's degree.

I would like to thank to my thesis committee members, Asst. Prof. Haluk Bayram, Asst. Prof. Filiz Gürkan Gölcük and Prof. Cabir Vural, for generously dedicating their precious time and attention to assess and evaluate my thesis.

I am also grateful to TÜBİTAK (Scientific and Technological Research Council of Türkiye) for granting me the "2210-C National MSc/MA Scholarship Program in the Priority Fields in Science and Technology" scholarship for this thesis.

I extend my gratitude to Asiye Demirtaş and Muhammed Koray Yılmaz for their efforts in developing a thesis format compatible with LaTeX. Their contributions have facilitated my academic work in a professional manner.

I would like to thank to my family for supporting and sponsoring me not only on this thesis but throughout my academic career as well.

GENİŞ ÖZET

Otonom İnsansız Deniz Araçları için COLREG-uyumlu ve Önceliksiz Hareket Planlaması

Mustafa Bayrak

Yüksek Lisans Tezi, Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Haluk BAYRAM

Ağustos, 2023. 48 sayfa.

Bu tezde insansız suüstü araçlarında kullanılmak üzere Denizde Çatışmayı Önleme Tüzüğü'ne (DÇÖT) uyumlu bir rota planlama algoritması geliştirilecektir. DÇÖT 1972'de Uluslararası Denizcilik Organizasyonu (IMO) tarafından yayınlanmıştır. Bu tüzüğün amacı, uluslararası sularda seyreden gemilerin aralarında yaşanabilecek çarpışmaların önüne geçmektir. Otonom suüstü araçları için DÇÖT uyumlu rota planlaması geliştirileceği zaman tüzüğün 13. ile 17. kuralları arası önemlidir. Otonom suüstü araçları sivil ve askeri alanda birçok çözümün üretilmesinde kullanılan robotik platformlarıdır. Son yıllarda otonom suüstü araçlarında hareket planlama algoritmaları üzerine yapılan akademik çalışmaların ve bu alandaki özel sektör girişimlerinin artışı söz konusudur. Otonom suüstü araçlarının kargo taşımacılığı ve toplu ulaşım gibi alanlarda kullanılabilmesi için yalnızca DÇÖT'ye uyumlu olması yeterli olmayabilir. 1972'de ortaya konan tüzük, insanlı komuta edilen gemiler göz önünde bulundurularak tasarlanmıştır. İnsanlı komuta edilen gemilerin çoğunlukta olduğu bir deniz trafiğinde bulunan bir otonom suüstü aracının güvenle seyredebilmesi için olabildiğince güvenli ve temkinli bir hareket planlama algoritmasına ihtiyacı vardır.

Suüstü araçlarının otonom kabiliyetlerinin artırılması için insanlı komuta edilen gemilerin çoğunlukta olduğu trafiklerde uzun saha deneylerinin yapılması gereklidir. Bunun yapılabilmesi için de olabildiğince güvenli ve muhafazakâr bir hareket planlama algoritmasına ihtiyacı vardır. Bu tezde ortaya konan önceliksiz hareket planlaması sayesinde otonom suüstü aracı başka bir gemi ile karşılaştığında tüzüğün 13. 14. ve 15. kuralları çerçevesinde yol veren gemi konumuna düşecek, bunun gerçekleşemeyeceği koşullarda karşılaşma durumuna düşmekten kaçınacaktır. Bu sayede otonom suüstü aracı güvenli bir şekilde seyredecek, kritik durumlarda inisiyatifi insanlı kumanda edilen deniz araçlarına bırakmış olacaktır. Literatürde bulunan birçok çalış-

mada otonom suüstü aracının yol verilen tekne konumunda olduğu ikili etkileşimlere dair çözümler üretilmiştir. Tamamen otonom suüstü araçlarının gerçek dünyaya entegre edilebilmesi, lojistik ve turizmde kullanılabilmesi için diğer insanlı kontrol edilen gemiler ile etkileşimlerden olabildiğince kaçınan, kaçınamadığı durumlarda yol veren tekne konumuna düşerek deniz trafiğini tehlikeye sokmayan bir rota planlaması gereklidir. Hedeflediğimiz bu önceliksiz davranış modeline sahip hareket planlama algoritmasını gerçelkeyebilmek için bazı varsayımlarda bulunduk. Bu varsayımların birincisi otonom suüstü aracının geniş bir görüş alanına sahip olmasıdır. Geniş bir görüş alanı gereksiniminin sebebi diğer gemileri onların etki alanına (ship domain) girmeden önce tespit edebilmek ve diğer gemilerin etki alanına girmeden hareket planlaması yapabilmektir. Yaptığımız diğer varsayım ise gemilerin etki alanı ile ilgilidir. Otonom suüstü aracının bulunduğu deniz trafiğindeki gemilerin her biri için çembersel birer etki alanı tanımladık. Bu çembersel etki alanlarının yarıçapları geminin ebatları ile orantılı bir şekilde hesaplanmaktadır. Gemiler için tanımladığımız etki alanlarına başka bir geminin girmesi durumunda, diğer geminin bulunduğu bağıl konuma göre uygulanması gereken DÇÖT kuralı değişmektedir. Önceliksiz davranış modeline sahip otonom suüstü aracının diğer geminin rotasını tekrardan planlamasına neden olmaması için diğer geminin etki alanının pruva kısmında veya sancak kısmında kalan bölümüne düşmemelidir. Önceliksiz davranış modeline sahip otonom suüstü aracının diğer gemilerin etki alanına girmesine dair kabullerimiz, DÇÖT'nin aşağıdaki maddelerine dayanmaktadır:

- Denizde Çatışmayı Önleme Tüzüğü'nde iki geminin KURAL 14: PRUVA PRUVAYA GELİŞ DURUMU başlıklı kuralında tarif edilen senaryoda iki geminin de rotasını sancak yönüne değiştirmesi gerekmektedir. Dolayısıyla otonom suüstü aracının başka bir gemi ile pruva pruvaya gelmemesi gereklidir.
- Denizde Çatışmayı Önleme Tüzüğü'nde iki geminin KURAL 15: AYKIRI GEÇİŞ başlıklı kuralında tarif edilen senaryoda diğer gemiyi sancak tarafından gören geminin diğer geminin pruvasından geçmemesi gerekir. Dolayısıyla otonom suüstü aracının başka bir geminin sancak tarafında konumlanmaması gereklidir.

Bu tezde öne sürdüğümüz hareket planlama algoritmasına ek olarak, insansız suüstü

araçları için hareket planlama algoritmalarını gerçek dünyada olduğu gibi birden fazla geminin bulunduğu gerçekçi deniz trafiği senaryoları altında test etme ve doğrulama problemini de ele aldık. İnsansız suüstü araçarı için bir planlama algoritması geliştirildiğinde, bu algoritmayı gerçekçi bir deniz trafiğini içeren ve gerçek dünya koşullarına benzeyen senaryolar altında test etmek çok önemlidir. Bu amaçla, araştırmacıların istenilen sayıda gemi eklemesine ve rotalarını manuel olarak tanımlamasına veya Otomatik Tanımlama Sistemi (AIS) verilerinden otomatik olarak oluşturmasına olanak tanıyan açık kaynaklı bir Virtual RobotX tabanlı simülasyon ortamı tasarladık. Bu simülasyon ortamında bulunacak olan her gemi, DÇÖT'ye uyumlu hareket planlamayı sağlayan algoritmalar ile donatılmıştır.

Bu algoritmalar şu şekilde özetlenebilir:

- Küresel hareket planlayıcı, geriye yönelik Otomatik Tanımlama Sistemi verilerini veya kullanıcının manuel olarak girdiği rotaları kaynak alarak simülasyon ortamında bulunan gemilerin küresel rotalarını belirler.
- Algılama modülü, simülasyondaki gemilerin üzerinde bulunan Küresel Konumlandırma Sistemi (GPS) sensörlerini kullanarak gemilerin birbirlerine uzaklıklarını esas alarak gemilerin birbirlerini görmesini makul işlem gücü gereksinimi ile simüle eder.
- Gemi etki alanı, algılama modülünden alınan veriye göre gemilerin karşılaşma durumunda olup olmadığını kontrol eder. Literatürde gemi etki alanı teorisine dair birçok modelleme yapılmıştır. Basit, anlaşılır ve düşük işlem gücü gereksinimine sahip olması adına çalışmamızda çembersel bir gemi etki alanı modelledik. Bu çemberin yarıçapı geminin boyutlarına göre belirlenmekle beraber kullanıcıların tercihine ve yaptıkları kabullere göre değişiklik gösterebilir.
- Anahtarlama mekanizması, bir geminin yerel rota planlama veya küresel rota planlama durumlarından hangisinde olduğunu kontrol eden bir mekanizmadır. Gemi etki alanından aldığı bilgiye göre geminin bir karşılaşma durumunda olup olmadığını kontrol eder. Karşılaşma durumunda yerel rota planlamayı çalıştırır. Yerel rotanın simülasyon ortamında bulunan diğer gemiler tarafından kesintiye

uğratıldığı durumda yeni bir yerel rota planlar. Yerel rota başarılı bir şekilde takip edildiği durumda küresel rotaya geri dönüşü sağlar.

- Yerel rota planlama, anahtarlama mekanizması tarafından uyarıldığı takdirde gemi için DÇÖT'ye uyumlu bir rota planlamasını sağlar. Bu çalışmada Hızla genişleyen Rastgele Ağaç (RRT) algoritmasının DÇÖT'ye uyumlu bir rota oluşturması için geminin bulunmaması gereken konumlara sanal engeller yerleştirilerek yerel rota oluşturulur.

- Rota takip mekanizması, geminin küresel rotasını ve o esnada varsa yerel rotasını analiz edip geminin o esnada hangi ara hedef noktaya ulaşması gerektiğini belirler.

- Kontrolcü mekanizması, geminin ara hedef noktaya ulaşabilmesi için gerekli eyleyici komutlarını üretir. Kontrol algoritması olarak bu çalışmada Saf Takip (Pure Pursuit) algoritmasını kullanmayı tercih ettik. Sabit hızla hareket eden gemilerde dümen açısının kontrolü ile rota takibinin başarılı bir şekilde yapılabilmesi, katsayılarının daha kolay ayarlanabilmesi gibi nedenler bu algoritmayı tercih etmemizde etkili oldu.

Bahsedilen simülasyon ortamının yazılım mimarisinde paralel gerçek zamanlı algoritmaların koşturulabildiği, açık kaynaklı yapısı sayesinde sensörler, işlem birimleri için bol ve çeşitli kaynakların bulunabildiği, geniş forum ve yardım ağı bulunan yapısal bir yazılım olan Robot İşletim Sistemi (ROS) kullanılmıştır. Simülasyon deneyleri için ROS ile gerçek zamanlı haberleşebilen 3 Boyutlu simülasyon motoru olan Gazebo tercih edilmiştir. Gazebo simülasyon ortamı, ROS'u üreten ve yürütücülüğünü üstlenen kurum olan Open Robotics'in ürettiği simülasyon ortamıdır. ROS'a uyumlu olması, uluslararası robotik camiasında yaygın kullanımı, gerçekçi fizik motoru ve düşük işlem gücü gereksiniminden dolayı tercih edilmiştir. Gerçek dünyada çalışması için üretilen robotlarda algoritmaların gerçek robot üzerinde denemeden önce bir simülasyon ortamında test edilmesi projenin maliyetini düşürür ve yaşanabilecek olası kırım hadiselerinin önüne geçer. ROS tabanlı yazılım mimarisinde, ROS düğümleri olarak kullanılan Python ve C/C++ dillerinde yazılan algoritmalar kullanılarak simülasyon ortamında bulunan gemilerin kontrolü sağlanmaktadır.

VRX simülasyon ortamı, Open Robotics'in düzenlediği RobotX yarışmasının bilgisa-

yar ortamında modellenmiş halidir. VRX simülasyon ortamında üzerinde IMU, lidar, kamera, GPS gibi robotlarda kullanılan temel sensörlerin bulunduğu pervane itkili katamaran bot modeli bulunmaktadır. Dalga, akıntı, rüzgâr, sis gibi gerçek hayatta açık denizlerde seyreden gemilere etki eden bozucu etmenler modellenmiştir. Durağan engellerin modellenmesi için dubalar, işaretçiler ve liman modelleri bulunmaktadır. Geliştirdiğimiz simülasyon ortamı, oldukça gerçekçi olan VRX simülasyon ortamını temel alarak tasarlanmıştır. Gerçek bir otonom insansız suüstü aracı için tasarlanacak yazılımların testinin öne sürdüğümüz simülasyon ortamı ile yapılmasının mümkün olduğunu göstermek adına bu tezde öne sürdüğümüz hareket planlama algoritmasını bu simülasyon ortamı ile test ettik.

Bu simülasyon ortamını geliştirmekteki amacımız insansız suüstü araçlarında hareket planlama algoritmaları üzerine çalışan araştırmacıların geliştirdikleri algoritmaları test etmekte kullanabileceği, deniz trafiğinin modellenmesi adına tasarlanan yazılımın parametrelerini değiştirebilecekleri veya diledikleri modülleri ekleyebilecekleri, açık kaynak kodlu ve geliştirmeye müsait bir simülasyon ortamının ortaya konmasıdır. Yaptığımız literatür taramasında araştırmacıların ortaya koydukları hareket planlama algoritmalarının test edilmesi için kendi simülasyon ortamlarını oluşturduklarını gördük. Bunların çoğunluğu iki boyutlu ve bahsettiğimiz çevresel etmenleri içermeyen nispeten daha basittir. Ortaya koyacağımız hareket planlama algoritmasının sanal ortamda test edilmesi için tür bir simülasyon ortamına ihtiyacımız vardır. Akademik literatürde gördüğümüz bu açık kaynak kodlu simülasyon ortamı eksikliğini giderebilmek için modüler, anlaşılabilir bir kaynak koduna sahip, geliştirmeye açık, iyi dokümante edilmiş ve diğer araştırmacıların da kullanabilmesini göz önünde bulundurarak bir simülasyon ortamı tasarladık.

Simülasyon ortamında bulunan diğer gemilerin gerçekçi bir deniz trafiği ortamını modelleyebilmesi adına geriye yönelik Otomatik Tanımlama Sistemi (AIS) verilerinden faydalanan MATLAB tabanlı bir çözüm geliştirdik. AIS, gerçek zamanlı gemi pozisyon takibinin yapılabildiği bir servistir. Gemilerde bulunan vericilerin yer istasyonlarına gönderdikleri konum, doğrultu ve gemiye dair detayların bulunduğu sinyaller sayesinde dünya genelinde gemi takibi yapılmaktadır. Bu veri tabanından elde edilen bilgilerde dakika cinsinden gecikmeler yaşanmaktadır. Gemilerin kıyılardan uzaklaş-

ması sonucunda gönderdikleri sinyallerde paket kaybı yaşanabileceği gibi karada bulunan alıcıların menzilinden tamamen çıkabilir. Okyanusta bulunan gemilerin takibinin yapılabilmesi için uydu görüntüsü tabanlı ve yapay zeka destekli sistemler kullanılmaktadır. Gerçek zamanlı AIS verileri ileri bir tarihte kullanmak adına kayıt edilebilir. Bu kayıt edilmiş verilere geriye yönelik AIS verisi denir. Geriye yönelik AIS verisini depolayıp talep eden müşterilere satan servislerin yanında belirli bir bölgenin verisini ücretsiz olarak kullanıma açan devlet kuruluşları mevcuttur. Simülasyonda AIS verisinin kullanılabilmesi için yalnızca verinin edinilmesi yeterli değildir. Edinilen geriye yönelik AIS verisi hem yüzey alanı hem süre olarak simüle edilebilir olmayan çok büyük ölçekte veriler olarak depolanmış olabilir. Dolayısıyla büyük bir AIS verisinden simüle edilebilir küçük veri parçalarının seçilebilmesi ve bu veri parçasının simülasyonda kullanıma uygun formatta kaydedilmesi gerekir. Bu problemi çözmek için MATLAB tabanlı bir analiz yazılımı geliştirdik. Bu yazılım büyük veriyi konum ve zaman dilimine göre ayrıştırır ve ayrıklaştırılmış bu veri parçalarını içinde bulunan gemi sayısı, kat ettikleri mesafe, gönderdikleri AIS verisi gibi parametrelerini hesaplar. Geliştirdiğimiz MATLAB yazılımı sayesinde kullanıcılar simülasyonlarında kullanmak istedikleri deniz trafiği durumuna göre analiz yapıp simülasyonda çalıştırmaya hazır ve gerçekçi deniz trafiği içeren verileri büyük AIS verisinden kolayca seçip kullanabilirler.

Önerdiğimiz önceliksiz davranış modeli tabanlı hareket planlama algoritmasını tasarlarken ve test ederken geliştirmiş olduğumuz açık kaynak kodlu simülasyon ortamını kullandık. Gerçekçi bir deniz trafiğinin modellenmesi için geriye yönelik AIS verisini analiz edip trafiğin söz konusu olduğu bir konum ve saat dilimini ayıklamak için MATLAB tabanlı yazılımımızı kullandık. Yapılan simülasyon testleri neticesinde geliştirdiğimiz önceliksiz hareket planlama algoritmasına sahip otonom suüstü aracının deniz trafiğinde mevcut olan diğer gemilerin rotasını değiştirmeye zorlamadan güvenli bir seyir yaptığını gözlemledik.

Anahtar Kelimeler: insansız suüstü araçları, test ortamı, hareket planlama algoritması, Denizde Çatışmayı Önleme Tüzüğü, Virtual RobotX

ABSTRACT

COLREGs-Compliant and Non-Prioritized Motion Planning for Autonomous Unmanned Surface Vehicles

Mustafa Bayrak

Master Thesis, Electrical and Electronics Engineering Department

Supervisor: Asst. Prof. Haluk Bayram

August, 2023. 48 pages.

In this thesis, we introduce a motion planning algorithm for unmanned surface vehicles that has a non-prioritized behavior. This means that it avoids encounters that might prompt other vessels to re-plan their routes in order to comply with International Regulations for Preventing Collisions at Sea (COLREG).

The development of a safe and conservative motion planning algorithm is important for the seamless integration of autonomous vehicles into existing maritime traffic. This algorithm allows autonomous unmanned surface vehicles (USVs) to navigate confidently and responsibly, enhancing overall traffic management and ensuring adherence to maritime regulations. Through this approach, we aim to contribute to the broader vision of achieving a harmonious coexistence between autonomous and manned vessels in shared waterways. To realize this vision, we present a non-prioritized motion planning algorithm. Its core objective is to ensure that when an autonomous surface vehicle encounters another ship, it prioritizes yielding the right of way while meticulously adhering to the 13th, 14th, and 15th rules of COLREGs. This algorithm is designed to avoid encounters that may disrupt the routes of other vessels, ultimately fostering a safer and more efficient maritime traffic environment.

For implementation of our non-prioritized motion planning algorithm, we propose a Rapidly-Exploring Random Trees (RRT) based motion planning algorithm ensuring that the generated path does not lead to encounters that trigger the need for other vessels in the environment to revise their own paths. Central to this approach is the utilization of virtual obstacles. We guide the RRT algorithm by defining virtual obstacles it should avoid passing through. By this approach, we prevent the algorithm from generating paths that would cause the re-planning of routes for other vessels.

In addition to our proposed motion planning algorithm, we consider the problem of testing and verifying motion planning algorithms for unmanned surface vehicles under realistic scenarios in which multiple vessels travel around as they do in the real world. Once a planning algorithm is developed for USVs, it is crucial to test a planning algorithm under scenarios that closely resemble real world conditions, which include environmental disturbances and real marine traffic. For this purpose, we present an open-source Virtual RobotX-based (VRX) simulation environment that enables researchers to add an arbitrary number of vessels and manually define their routes, or automatically generate them from AIS data. Each vessel is equipped with global and local motion controllers complying with COLREGs.

Keywords: unmanned surface vehicles, testbed environments, motion planning, COLREGs, Virtual RobotX

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

AIS	Automatic Identification System
COLREGS	The International Regulations for Preventing Collisions at Sea
GPS	Global Positioning System
IMU	Inertial Measurement Unit
JSON	JavaScript Object Notation
LIDAR	Light Detection and Ranging
ROS	Robot Operating System
RRT	Rapidly exploring Random Tree
USV	Unmanned Surface Vehicle
VRX	Virtual RobotX
WAMV	Wave Adaptive Marine Vehicle

CHAPTER 1: INTRODUCTION

Unmanned surface vehicles (USVs) are rapidly gaining traction across a spectrum of applications, ranging from transportation and logistics to sea surface cleaning. The seamless and secure navigation of these USVs within marine traffic environments relies on compliance to International Regulations for Preventing Collisions at Sea (COLREGs) through robust motion planning algorithms. Given that a substantial 81.1% of maritime incidents in 2022 were attributed to contributing factors associated with the human element, the paramount importance of ensuring safe navigation takes precedence in the development of autonomous USVs designed to operate in waterways [European Maritime Safety Agency, 2022].

To increase the autonomy of USVs, it is important to conduct extensive field tests in dense maritime traffic environments where manned vessels are in majority. These tests require a safe and conservative action planning algorithm. This thesis proposes a non-prioritized motion planning approach that ensures that when an autonomous surface vehicle encounters another ship, it will prioritize giving way while adhering to the 13th, 14th, and 15th rules of the COLREGs. This will allow the autonomous surface vehicle to navigate safely, and during critical situations, the responsibility for avoiding collision will be transferred to the manned sea vehicle.

For testing COLREGs-compliant motion planning algorithms, utilizing Automatic Identification System (AIS) can be an effective way. AIS is an automatic vessel tracking system that is widely used by ships in maritime traffic. In maritime traffic, AIS serves a critical role for vessel traffic services and ships in maritime traffic for a safe navigation. Real time AIS data is available through AIS receivers, while AIS data can be tapped into and stored for back tracking the maritime traffic. Historical AIS data can be a useful way for modeling realistic maritime traffic scenarios.

Motion planning is one of the main components of a USV. Many studies have developed COLREGs-compliant motion planning algorithms using various methods, such as genetic algorithms, model predictive control, generalized velocity obstacles, rolling horizon optimization, negotiation-based approaches, and collision risk computation-based

approaches [Akdağ et al., 2022]. However, we identify the need for a more conservative motion planning algorithm for USVs in marine traffic environments where manually controlled vessels are in the majority. This algorithm should avoid interfering with the routes of other vessels. Therefore, our proposed non-prioritized motion planning algorithm can serve to fulfill this need.

1.1 Problem Statement

In this thesis, we consider the problem of designing a COLREGs-compliant motion planning algorithm for autonomous vessels that ensures non-interference with other vessels' routes within the simulation environment. Thus, our objective is to develop a conservative motion planning approach for autonomous unmanned surface vehicles, operating in a maritime traffic where manned ships predominate. An illustration of the problem can be seen in Figure 1.1.

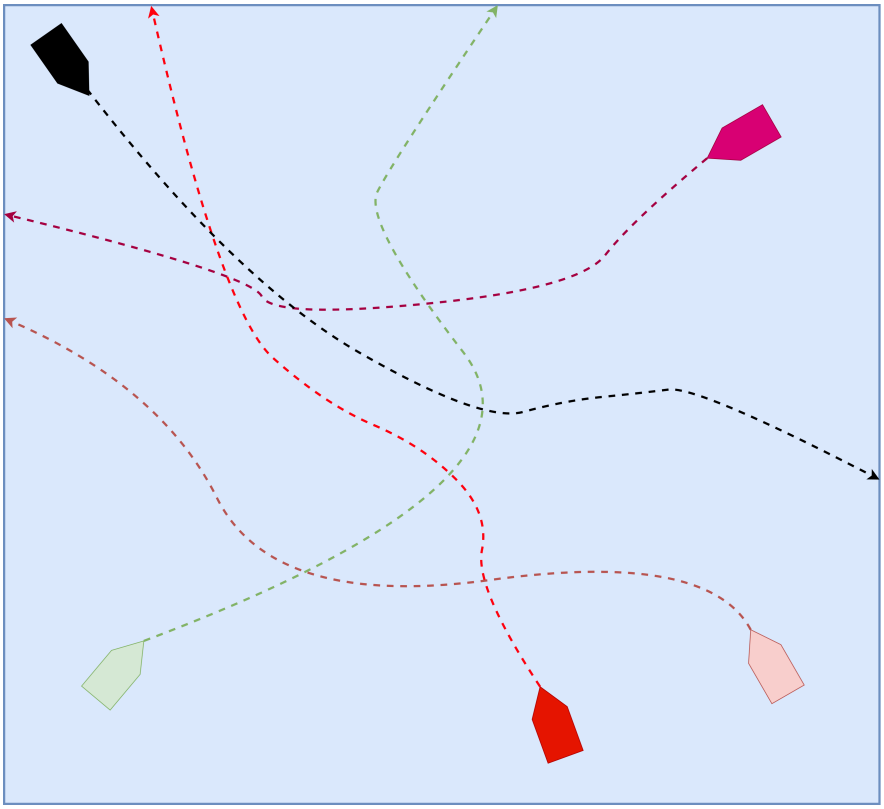


Figure 1.1: Illustration of maritime traffic environment with vessels in the traffic and routes of the vessels.

1.2 Related Literature

Numerous studies offer solutions to the route planning problem in autonomous surface vehicles, and one such study is conducted by [Singh et al., 2018]. In their research, they propose an A* based dynamic route planning algorithm, which is thoroughly tested using 2D simulations. The algorithm takes into account various factors, including stationary obstacles, moving objects, and sea currents in the environment. However, it is worth noting that the COLREGs are not considered in their study. The importance of adhering to this regulation is acknowledged in the conclusion of the article, and it is recommended as a topic for further investigation. In our thesis, we are presenting a COLREGs-compliant motion planning algorithm.

Regarding autonomous surface robots, there are studies considering COLREGs. These studies apply some or all of the scenarios described in Rules 13, 14, and 15 to enhance the collision avoidance capabilities of the vehicles.

In [Beser and Yildirim, 2018], two-dimensional simulation tests on the binary map of Prince Islands region are conducted, implementing the 14th and 15th rules of the regulation. These simulations focused on encounter scenarios involving only two surface vehicles in the environment. Notably, the simulations assumed that the LIDAR and RADAR sensors of autonomous surface vehicles were non-functional. Instead, the direction angle of the other surface vehicle was calculated using camera view and compass angle. The study did not include simulations or discussions regarding the 13th rule of the regulation. To address these limitations, comprehensive simulations that account for all three rules are conducted in this thesis.

In their study, [Zaccone et al., 2019] proposed a Rapidly-exploring Random Trees Star (RRT*) based motion planning algorithm. The research focused on two-dimensional simulations to explore interactions between two ships in accordance with COLREGs. The mathematical formulation of bilateral interaction scenarios, as specified by the regulation, was defined using simple inequality sets and integrated into the route planning structure. The developed algorithm underwent simulation tests across three different scenarios. For each scenario, two routes were drawn: one compliant with the regulation

and another disregarding the regulation. The study demonstrated the applicability of the RRT* algorithm for a route planning solution in compliance with COLREGs. However, it's worth noting that the experimental part only simulated bilateral interactions between two ships. Environments with more than two ships, especially where maritime traffic is intense, require the autonomous surface vehicle to accurately assess interactions with multiple ships and make decisions based on the relevant regulations when approaching each ship. To address this limitation, simulations involving multiple ships are conducted in this thesis. Our goal is to design a motion planning algorithm to effectively handle complex encounter scenarios in busy maritime traffic, thereby ensuring safer and more efficient navigation for autonomous surface vehicles.

In their study, [Wang et al., 2021] presented a collision avoidance algorithm designed to comply with the COLREGs. The research involved 2D MATLAB simulations, incorporating the characteristics of the unmanned surface vehicle named "Jiuhang490". Conducted simulations covered various scenarios, including encounters with stationary objects, a ship moving linearly at a constant speed, and stationary obstacles and ships in the environment. To avoid both stationary and moving obstacles, the researchers developed a risk level model. Circular regions were defined around each object, with the risk coefficient decreasing from the inner region to the outer region. For route planning, a navigation waypoint model was utilized to reach multiple waypoints one by one in order to reach the ultimate destination. Two motion models were employed in the simulations. Unmanned surface vehicle executed maneuvers using rudder commands while maintaining a constant speed. However, in emergency situations where the risk of collision was imminent and rudder movements alone were insufficient, the vehicle's speed was also adjusted. It is essential to note that the route planning proposed in this study was primarily designed for local contexts. While most simulations involved only one ship, the researchers acknowledged that real-life usage should consider dense maritime traffic scenarios. In our thesis, simulations are conducted to test the algorithm's performance in environments with dense traffic, aiming to develop a more robust and adaptable route planning solution for unmanned surface vehicles.

In [Rothmund et al., 2022], a dynamic Bayesian network is utilized to estimate the intentions of other ships. This modeling approach is crucial as it considers actions that

may not comply with the COLREGs and accounts for uncertain COLREG situations. In our approach, we provide an environment with vessels navigating in accordance with the COLREGs to thoroughly test the developed algorithms.

For collision avoidance, [Hagen et al., 2022]. utilize a Scenario-based Model Predictive Control approach . In their study, they conduct simulations comparing the original approach introduced in [Hagen et al., 2018] with a newly introduced Multi-step Scenario-based Model Predictive Control method. The motivation behind the design of the multi-step Scenario-based Model Predictive Control was to overcome the limitations of the single-step approach. By transitioning to a multi-step framework, the aim was to provide a more comprehensive and flexible solution for collision avoidance and motion planning. The simulations evaluate the performance of both approaches in various scenarios. The results of the simulations indicate that while the Multi-step approach shows slight improvement in only a few scenarios, the original method remains effective for motion planning.

Global and local path planning can be a useful approach for COLREGs-compliant motion planning. [Zhao et al., 2023] proposes a hierarchical framework involving global path optimization using an adaptive genetic algorithm with fuzzy interference and reactive local collision avoidance using a sensory vector structure. Simulations compare the approach with conventional genetic algorithm , improved artificial fish swarm algorithm and improvement adaptive ant colony optimization in terms of time consumption and solution quality. The proposed approach outperforms in cruising time, trajectory smoothness, and safety. The hierarchical path planning framework also maintains satisfactory trajectory computation time. Our thesis also involves a global and local path planning approach for achieving a reactive COLREGs-compliant motion planning while following a realistic maritime trajectory.

Autonomous collision avoidance in crowded ocean environments can be challenging to achieve using traditional methods. In [Peng et al., 2023], Random Network Distillation and Proximal Policy Optimization are combined into Proactive Obstacle Navigation and Detection algorithm. In order to achieve COLREGs-compliant motion planning, the reward module proposed in this work consists of voyage penalty, arriving reward,

heading error, cross error reward and to COLREGs reward. It is worth mentioning that this study employs an unmanned aerial vehicle serving perception and data collection purposes for autonomous ships. Simulations indicate that the POND decision-making model exhibits faster convergence, improved search efficiency, and higher reward values. This study utilized a Unity game engine based 3D simulation environment is run on Windows 10 operating system. In our thesis, we developed an open-source Gazebo simulation environment that is compatible with the Robot Operating System (ROS) [Quigley et al., 2009], a widely adopted platform among researchers.

The Virtual RobotX (VRX) simulation environment [Bingham et al., 2019] is an open-source project designed for testing marine vehicle motion planning algorithms. It includes a realistic ocean environment and incorporates environmental disturbances such as winds and waves. Although it is not exclusively developed for marine traffic simulation, it can be modified for testing COLREGs-compliant motion planning algorithms and utilized for marine traffic simulation.

In recent years, various testbed environments for USV motion planning algorithms have been developed. For instance, [Bolbot et al., 2022] generate numerous traffic scenarios using Sobol sequences and select risky scenarios from them. However, they do not utilize AIS data. In our thesis, we utilize AIS data for marine traffic generation.

There are also studies that utilize AIS data. For example, [Bakdi et al., 2021] process maritime big data for evaluating collision and grounding assessments with a 15-minute prediction horizon. Their approach identifies risky situations in marine traffic. In our thesis, we use AIS data for global path planning in the developed simulation environment.

1.3 Contributions

Our main contributions in this thesis are as follows:

- Developing a non-prioritized motion planning algorithm that guarantees non-interference with other vessels' routes. As a result, our algorithm facilitates a

safer and more conservative path planning for autonomous surface vessels navigating within marine traffic environments where manned ships are predominant. By prioritizing safety and adherence to the COLREGs, our motion planning approach contributes to the seamless integration of autonomous vessels into busy maritime settings, reducing the risk of collisions and promoting efficient and secure navigation.

- Designing an open-source 3D realistic simulation environment that includes real marine traffic and environmental disturbances such as water current and waves for researchers to test their motion planning algorithms.

1.4 Approach

Many studies in the literature have treated autonomous surface vehicles and manned command ships equally in encounter situations. However, for fully autonomous surface vehicles to successfully integrate into the real world, especially in domains like logistics and tourism, it is crucial to design a route planning strategy that minimizes interactions with manned-controlled vessels and avoids disrupting their routes, ensuring the safety of maritime traffic. This distinctive aspect of our non-prioritized motion planning approach is achieved by defining circular ship domains for other vessels in the simulation environment and introducing virtual obstacles with dimensions equal to or larger than the vessels' ship domains. These virtual obstacles are circular to align with our circular ship domain assumption. To ensure non-interference with other vessels' routes, our approach restricts the planned path from extending into the areas ahead or starboard of other vessels. Our non-prioritized motion planning approach prioritizes secure navigation, even if it results in a longer trajectory compared to conventional COLREGs-compliant motion planning.

For testing and verifying USV motion planning algorithms such as the one we developed in this thesis, researchers often design their own simulation environments, considering various parameters for their research. Therefore, we believe that there is a demand for an open-source 3D realistic simulation environment that includes realistic marine traffic and environmental disturbances, such as water currents and waves. By utilizing such a simulator, researchers can test and validate their own motion planning algorithms,

ensuring compliance with the COLREGs and performance under realistic and dynamic conditions before deploying them in real-world applications. For this purpose, we designed the multiple vessels simulation environment [Bayrak and Bayram, 2023a], an open-source simulation environment capable of modeling realistic marine traffic with environmental disturbances, particularly for multiple COLREGs-compliant vessels. We leverage AIS data for a realistic maritime traffic and implement a modified RRT algorithm for local path planning. All software components are realized using the ROS. We have implemented a modular design, allowing users to effortlessly incorporate new elements or modify existing algorithms according to their preferences. This modular design approach enables further development and customization within our simulation environment, for different research needs and requirements.

In order to generate a realistic marine traffic simulation environment, we utilize a Gazebo-based VRX simulation environment [Bingham et al., 2019]. Therefore, our approach is compatible with Gazebo and ROS. Because of the years of development on the VRX project by the Open Source Robotics Foundation, VRX simulation environment has many capabilities that benefits us for testing and verifying USV motion planning algorithms.

Figure 1.2 presents the steps in developing the simulation environment. In this process, the first step determines, from AIS data, initial poses and trajectories for the vessels in the simulation, and then the second step uses this information to generate the ROS launch files. In the last step, we initialize the simulation environment and execute the ROS nodes that are responsible for the motion planning of all the vessels.

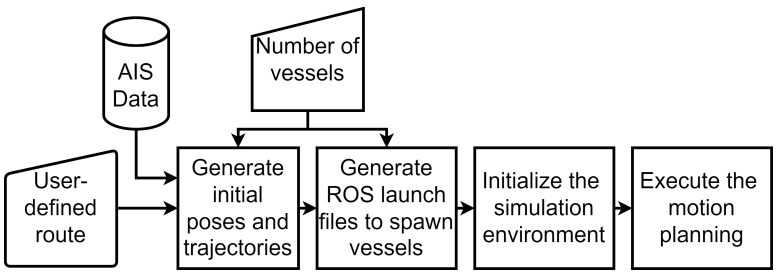


Figure 1.2: Overview of the proposed approach.

As can be seen in Figure 1.3, the generated multiple vessels can navigate in compliance with the COLREGs in the simulation environment. The developed simulation envi-

ronment generates the vessels, sets their global paths and controls them in real time. The red USV executes the motion planning algorithm or any algorithm the user would like to test.

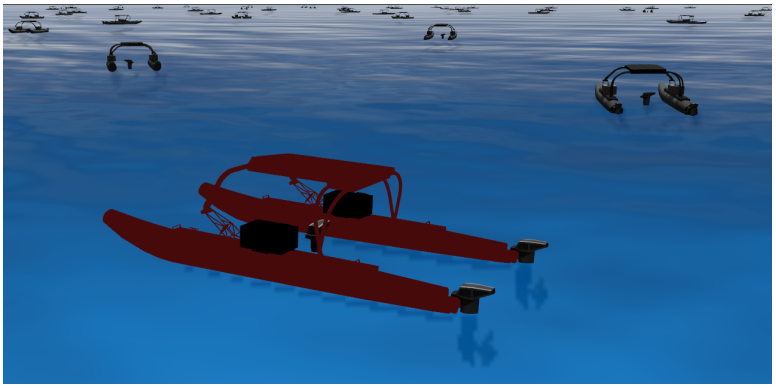


Figure 1.3: Snapshot from the simulation environment populated with 100 vessels. The red USV will be controlled by the motion planning algorithm developed by the user.

1.5 Outline

This thesis consists of six chapters. This first chapter presents an introduction to the thesis, including the problem statement, a review of related literature, the contribution of the thesis, the proposed approach, and an outline of the thesis. In Chapter 2, we provide a detailed description of the non-prioritized motion planning algorithm proposed in this thesis. We elaborate on the modules we designed for the algorithm. In Chapter 3, we introduce our open-source multi vessel simulation environment for verifying COLREGs-compliant motion planning algorithms. We provide detailed explanations and pseudo-codes for the designed ROS nodes responsible for handling various aspects of the simulation environment. The chapter concludes with the presentation of the overall ROS system depicted through block diagrams. In Chapter 4, we explain our method for utilizing AIS data to have a realistic maritime traffic environment within our simulation. We provide description of our method to filter and crop the large AIS data to extract a suitable clip for simulation runs. This process ensures that the selected AIS data accurately represents the desired scenario and reflects real-world maritime traffic dynamics. In Chapter 5, we present the simulations obtained from the implementation of our motion planning algorithm within the realistic simulation environment. These chapters are followed by Chapter 6, where we discuss the conclusion.

CHAPTER 2: NON-PRIORITIZED MOTION PLANNING

In this chapter, we aim to provide a comprehensive insight into the idea behind the development of our non-prioritized motion planning algorithm. To implement our motion planning algorithm, certain assumptions were made, primarily focusing on our non-prioritized vessel's perception capabilities. We assumed our vessel to have excellent perception, capable of detecting other vessels within a range of kilometers. With this level of perception, our non-prioritized vessel is alerted to initiate a path planning process that avoids interference with the routes of other vessels within its perception range.

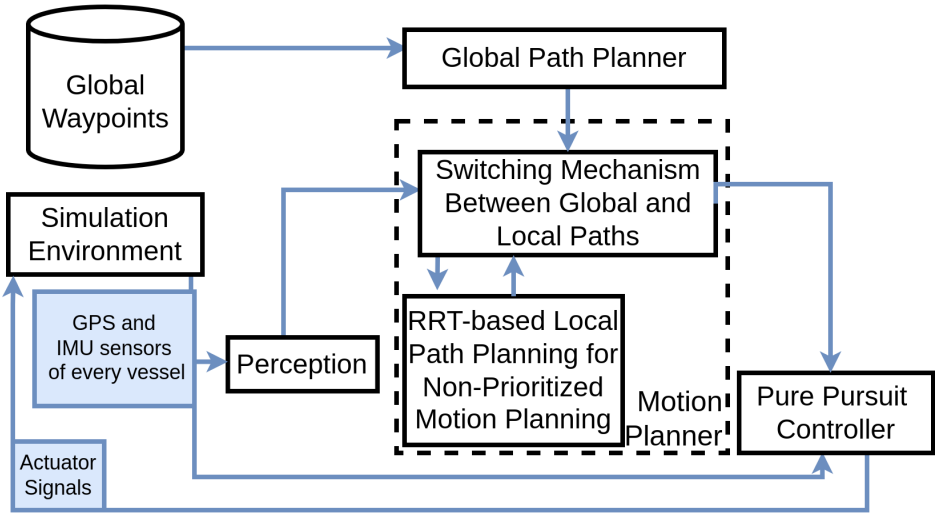


Figure 2.1: Simplified block diagram of non-prioritized motion planning.

Furthermore in this chapter, we will elaborate further on the implementation of our algorithms, which have been developed as ROS nodes. In Section 2.1, we describe the method to spawn USVs and set global paths for them. In Section 2.2, we explain how we emulated the perception for USVs that have non-prioritized motion planning. In Section 2.3 we discuss the switch mechanism to invoke local planner when necessary. In Section 2.5, we describe the local path generating process for the non-prioritized motion planning. Simplified block diagram of the proposed approach is given in Figure 2.1.

2.1 Global Path Planner

The `Global Path Planner` is designed to control the positions where USVs are spawned in the simulation environment and assign the pre-determined path that USVs must follow during the simulation run. Assigning data for this purpose is done through a JSON data file where users can manually determine a route for USVs.

2.2 Perception

To simulate perception for USVs implementing non-prioritized motion planning, we have devised a proximity-based approach. Our `Perception` module works by the information of GPS and IMU sensors of every other vessel present in the simulation environment. It then calculates the distance between the USV and each neighboring vessel, and deems that our USV can see another vessel if the distance falls within the predetermined perception range.

By implementing this proximity-based approach, our motion planning can effectively emulate perception capabilities, allowing them to accurately sense and respond to the presence of other vessels in their vicinity during the motion planning process. This feature is vital for ensuring safe and collision-free navigation in dynamic maritime environments.

2.3 Switching Mechanism Between Global and Local Paths

The `Switching Mechanism Between Global and Local Path` plays a crucial role in handling the transition between local and global paths for USVs. This module has the information from perception module of the USV and, based on the information received, invokes the `RRT-based Local Path Planning for Non-Prioritized Motion Planning` to generate a local path that avoids interfering with the current paths of other vessels.

2.4 Pure Pursuit Controller

The `Pure Pursuit Controller` determines the appropriate current waypoint that the USV should head towards, ensuring a successful navigation while following either the

local or global path. We employed a Pure Pursuit based trajectory tracking algorithm for our USV to successfully navigate through waypoints one by one.

2.5 RRT-based Local Path Planning for Non-Prioritized Motion Planning

In this section, we detail our approach for achieving a COLREGs-compliant path that avoids any interference with the routes of other vessels. The following parameters are used in the developed approach:

- **Ship Domain Radius** indicates the distance from a vessel at which an encounter situation occurs. This value is calculated for every vessel based on their size.
- **Perception Range** indicates the maximum distance from a vessel to be able to perceive other vessels.
- **Local Path Bounding Box** indicates the area for planning a local path.
- **Virtual Obstacle Radius** indicates the radius of virtual obstacles placed in local path bounding box to achieve COLREGs-compliant motion planning.

2.5.1 Ship Domain

To achieve path planning that is non-interfering with other vessels' routes, we adopted the circular ship domain approach for each vessel within the simulation environment. This circular ship domain assumption (as shown in Figure 2.2) simplifies implementation and allows us to define specific areas inside the ship domain corresponding to the relevant COLREG rules. Specifically, Rule 13, Rule 14, and Rule 15 describe encounter situations and the recommended navigation scenarios for the involved ships. Additionally, Rule 16 defines the necessary actions for the give-way vessel, while Rule 17 outlines the appropriate actions for the stand-on vessel.

Although the angle range corresponding to a vessel's fore is not explicitly described in the rules, there are various assumptions proposing different ranges. For our implementation, we defined a 10-degree range for a vessel's fore side, corresponding to Rule 14. As for the stern side, we considered the angle range values of 247.5 degrees and 112.5 degrees for our approach since these values are clearly stated in Rule 13.

Within the framework of these rules, to ensure a non-interfering motion planning approach for our non-prioritized vessel, we avoid it from entering another vessel's ship domain from the fore and starboard sides. By adhering to this principle, we can guarantee that our vessel does not cause other vessels to re-plan their routes due to potential interference from an encounter situation with our non-prioritized vessel.

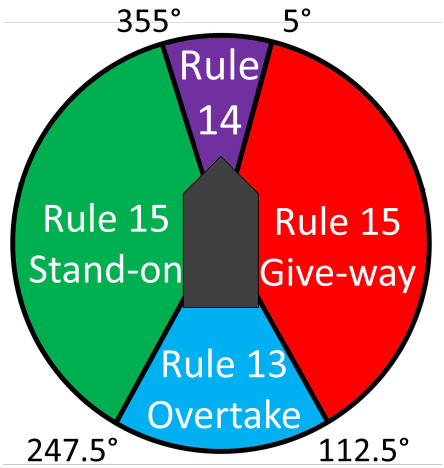


Figure 2.2: Ship domain and regions in accordance to which COLREG rule is valid.

2.5.2 Non-Prioritized Local Path Generation

Having clarified our ship domain assumption and its role in preventing interference with the routes of other vessels, we can now proceed to explain our approach for realizing this objective through non-prioritized motion planning. As seen in Section 2.5.1, it is important for our USV to avoid from approaching other vessels from their fore and starboard sides,to not invoke any alteration in their planned routes.

To prevent any interference with other vessels' routes, we introduce virtual obstacles with dimensions equal to or larger than the vessels' ship domains. These virtual obstacles take on a circular shape, in accordance with our circular ship domain assumption.

To ensure non-interference with other vessels' routes, our algorithm restricts the planned path from extending into the areas ahead or starboard of other vessels. Figure 2.2 illustrates the ship domain areas, with our algorithm avoiding the purple and red zones shown.

The reason we decided on this approach is that when our USV enters another vessel's

ship domain from ahead, it triggers COLREG Rule 14: Head-on Situation, requiring the other vessel to pass along our USV's starboard side. Similarly, if we enter another vessel's ship domain from starboard, we trigger COLREG Rule 15: Crossing Situation, making the other vessel the give-way vessel. As a result, the other vessel is expected to maneuver over our USV's stern and avoid crossing paths from ahead. This careful consideration of maritime regulations ensures safe and compliant interactions between our autonomous surface vehicles and other vessels, reducing the risk of potential collisions. With this virtual obstacles approach, we guarantee collision-free motion planning for our autonomous surface vehicles, maintaining a secure motion planning while navigating within complex maritime environments.

For ease of understanding and visualization, we have provided an example scenario to demonstrate our non-prioritized motion planning approach in Figure 2.3. In this illustration, our motion planning algorithm ensures that the non-prioritized vessel (depicted in bright green) avoids drawing a path that enters another vessel's ship domain from the fore or starboard side.

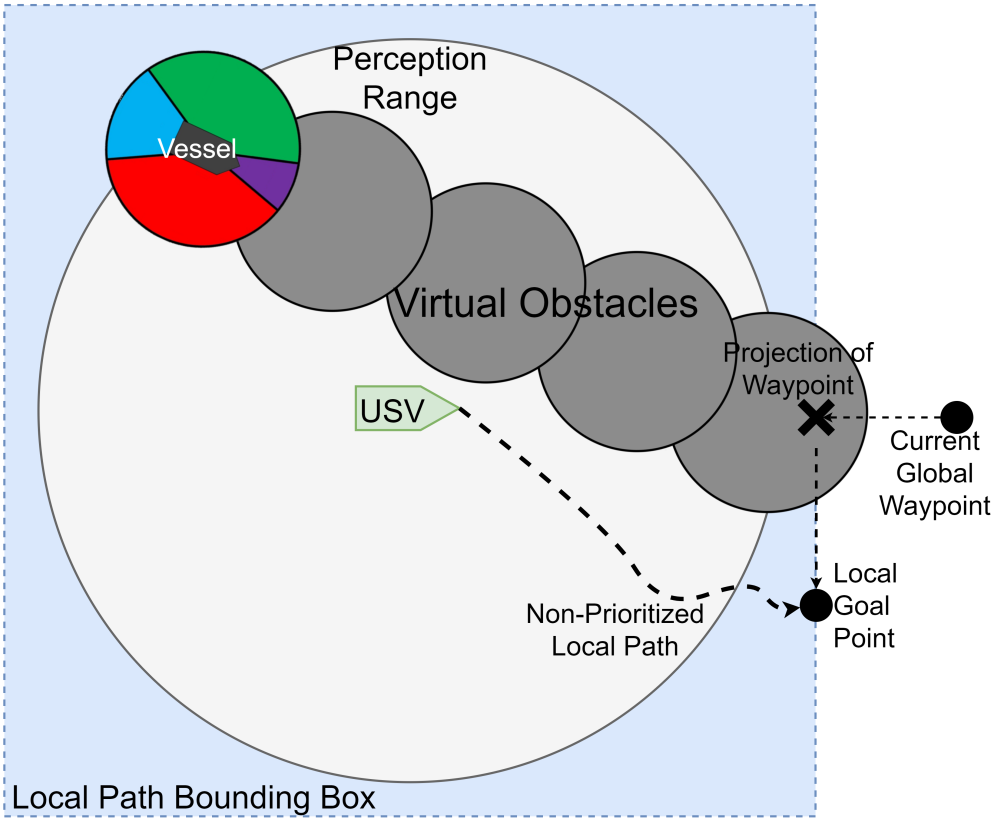


Figure 2.3: Example scenario for non-prioritized motion planning in action.

In the figure, a vessel (shown in dark grey) has entered the perception range of our USV. The ship domain of the vessel is also displayed for reference. It is obvious that directly navigating to the goal point could result in a possible encounter situation or, even worse, a collision risk with the vessel.

To mitigate this risk, the non-prioritized local path is generated, carefully avoiding entry into the vessel's ship domain from the fore or starboard side. By this approach, we ensure that our non-prioritized vessel can navigate safely and efficiently, reducing the likelihood of encounters with other vessels and promoting the overall safety of maritime traffic.

Furthermore, extending our precautions, it is important to ensure that our USV avoids navigating through the fore side of another vessel, even if our USV does not technically enter the ship domain of any other vessel. This preventive measure is necessitated by the forward propulsion of vessels, which could lead to an unintended encounter situation by our USV getting into another vessel's ship domain. To address this concern, we have positioned virtual obstacles which have the same size as ship domain of vessel along the heading axis of each vessel, ensuring that the RRT-based path planning algorithm cannot generate a path that involves navigating through the fore side of any vessel. As a result, our USV is effectively precluded from navigating through the fore side of any other vessel. These virtual obstacles are also visible in Figure 2.3 depicted in gray color.

In addition, we have addressed the challenge of defining an appropriate local goal point for the RRT algorithm to target in order to facilitate local path planning. Given that our vessel's route is determined through global path planning, we decided to project the current global goal onto our local path bounding box, represented in black cross named "Projection of Waypoint". This strategic choice allows us to guide our vessel toward the successful completion of the global path.

Furthermore, we recognized the potential issue of virtual obstacles impeding the RRT algorithm's ability to reach the projected local goal point safely. To mitigate this concern, we considered a solution: if a virtual obstacle collides with our local goal

point or if the RRT algorithm fails to discover a viable path within an acceptable number of iterations, we employ a clockwise adjustment to the projected local goal point. This adaptation is depicted in Figure 2.3, where the projected local goal point colliding a virtual obstacle is depicted as a black cross. The adjusted local goal point, automatically computed by clockwise repositioning inside the local path bounding box, is illustrated by a black circle named local goal point.

By implementing this strategy, we ensure that our RRT-based local planner can find a safe and feasible path to the projected local goal point, even in the presence of virtual obstacles.

Algorithm 1: `Non-Prioritized Local Path Planner`

Input: *local_path_args*, *vessels_in_perception*, *usv_gps*,*global_path*, *last_reached_global_waypoint*
Output: local_path

```
while True do
    // Check generate local path topic
    args_len ← |vessel_n_local_path_args|
    if args_len > 0 then
        // Generate local path
        virtual_obstacle_list = []
        foreach vessel ∈ vessels_in_perception do
            Place virtual_obstacle along the dead ahead axis (+x) of vessel.
            Append the virtual_obstacle to virtual_obstacle_list.
        Calculate local_goal_point using usv_gps, global_path,
         last_reached_global_waypoint
        local_goal_point_is_Free = true
        foreach virtual_obstacle in virtual_obstacle_list do
            if local_goal_point is colliding with virtual_obstacle then
                local_goal_point_is_Free = false
        while local_goal_point_is_Free == false do
            Push local_goal_point clockwise.
            if local_goal_point is not colliding with any virtual_obstacle then
                local_goal_point_is_Free = true
        local_path ← Generate_RRT_path(local_goal_point)
        Smooth local_path by removing unnecessary waypoints
```

The pseudo-code for the `Non-Prioritized Local Path Planner` is provided in Algorithm 1. In line 2, `local_path_args` is checked to determine whether the Non-Prioritized Local Path Planner is invoked to generate a local path. In lines 5-7, virtual obstacles are positioned along the dead-ahead axis of each vessel that perceptual capabilities of our USV can detect. Subsequently, the `local_goal_point` is calculated by projecting the current global waypoint, as outlined in line 8. The algorithm then evaluates whether this projected `local_goal_point` intersects with any virtual obstacles. If so, the algorithm employs lines 9-16 to reposition the `local_goal_point` in a clockwise manner, ensuring its accessibility for the RRT algorithm. Once the accessibility of the `local_goal_point` is ensured, the local path is generated using the RRT algorithm in line 17. Additionally, the generated local path is smoothed by removing unnecessary waypoints that do not have any obstacles in between.

CHAPTER 3: MULTI-VESSEL SIMULATOR

This chapter will outline the process of creating a multi-vessel environment for validating our non-prioritized motion planning algorithm. As previously detailed in Section 1.4, we employed a VRX-based simulation environment to simulate maritime traffic. Our goal is not only to have the vessels in the simulation follow their predetermined paths; we also aim to implement a reactive path planning for the vessels. This approach ensures that our motion planning algorithms can be tested within a context of COLREGs-compliant maritime traffic.

3.1 Motion Planning

Each vessel in the simulation has its own motion planner, which consists of two parts: Local and global planners. Global path of each vessel is assigned according to AIS data provided to simulation. Then, the assigned global path is tracked by a Pure-Pursuit tracker. During the trajectory tracking phase, whenever a vessel encounters another vessel or USV (Figure 3.1), the local planner takes control of the path planning. Therefore, for the local path planning, we employ a modified COLREGs-compliant RRT algorithm. We implement the motion planners as a ROS node.

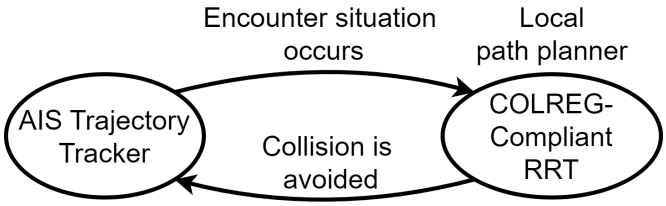


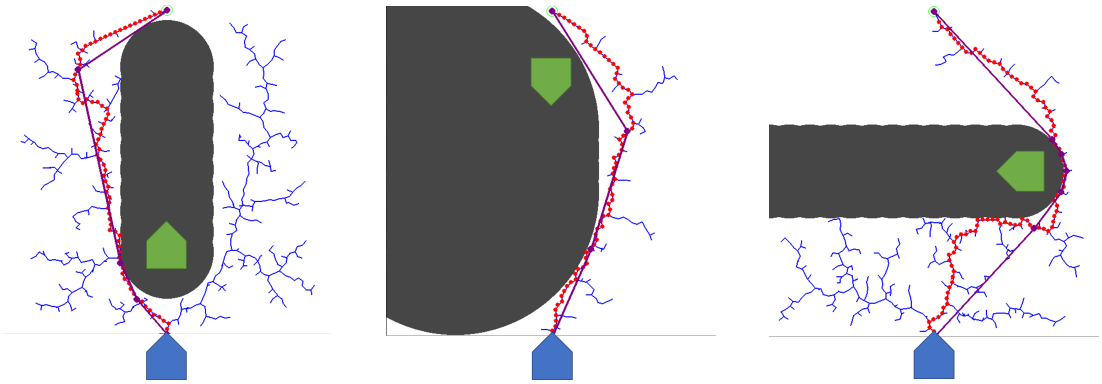
Figure 3.1: State machine of motion planning.

3.1.1 Global path planning

During the procedure of generating the simulation environment, global paths consisting of several waypoints for vessels are extracted from AIS data. If AIS data does not provide vessel information for a long period of time, our approach interpolates appropriate waypoints for the missing period. Pure-pursuit algorithm is implemented for tracking these paths [Chai and Hassani, 2019].

3.1.2 Local Path Planning

The motion planning controller switches to the local path planning if another vessel enters a vessel's ship domain, which is defined by a circular area around the vessel. When an encounter situation is detected, the algorithm determines which COLREG rule will be applied to the situation based on the relative angle between two vessels (Figure 2.2). The COLREG rules involved in these scenarios are Rules 13, 14, and 15 [Cockcroft and Lameijer, 2011]. Rule 13 states that a vessel is overtaking if it approaches to another from more than 22.5° abaft its beam. Therefore, it must overtake by choosing one side. Rule 14 states that if two vessels' courses are opposite, the vessels should pass on the port side of each other. Rule 15 states that if two vessels are crossing with a collision risk, the vessel which has the other vessel on its own starboard side should give way.



(a) Rule 13 (b) Rule 14 (c) Rule 15

Figure 3.2: Scenarios for the application of COLREG Rule 13 (a), Rule 14 (b) and Rule 15 (c) using a RRT-based local path planning.

After deciding the applicable COLREG rule, a COLREG-Compliant RRT algorithm generates the local path. Similar to [Chiang and Tapia, 2018], we define virtual obstacles in order to ensure that the RRT finds a COLREG-Compliant path. A path smoothing algorithm removes the unnecessary waypoints between any two waypoint if there is no obstacle between them [Zhang et al., 2022]. The encounter situations described in the COLREGs can be seen in Figure 3.2 in which virtual obstacles are denoted by gray color, paths generated by the RRT by red and smoothed paths by purple.

3.2 Gazebo-ROS Infrastructure

In this section, we will describe the implementation of our approach, including details about our implemented ROS nodes, scripts and the whole infrastructure. In Section 3.2.1, we describe the method for initializing the simulation environment. In Section 3.2.2, we explain how we emulated the perception using GPS sensors of vessels. In Section 3.2.3, we detail the ship domain method we used for vessels. In Section 3.2.5, we discuss how we designed the local path planner for vessels in order to achieve a COLREGs-compliant motion planning during an encounter. In Section 3.2.6, we describe how we utilized the AIS data in order to simulate realistic vessel traffic with our global path planner ROS node. In Section 3.2.7, we describe our method for tracking both local and global paths through our trajectory tracker ROS node. Finally, in Section 3.2.8, we explain the controller method we used to generate the actuator signals and reach waypoints. The simplified block diagram of the proposed ROS infrastructure is given in Figure 3.3.

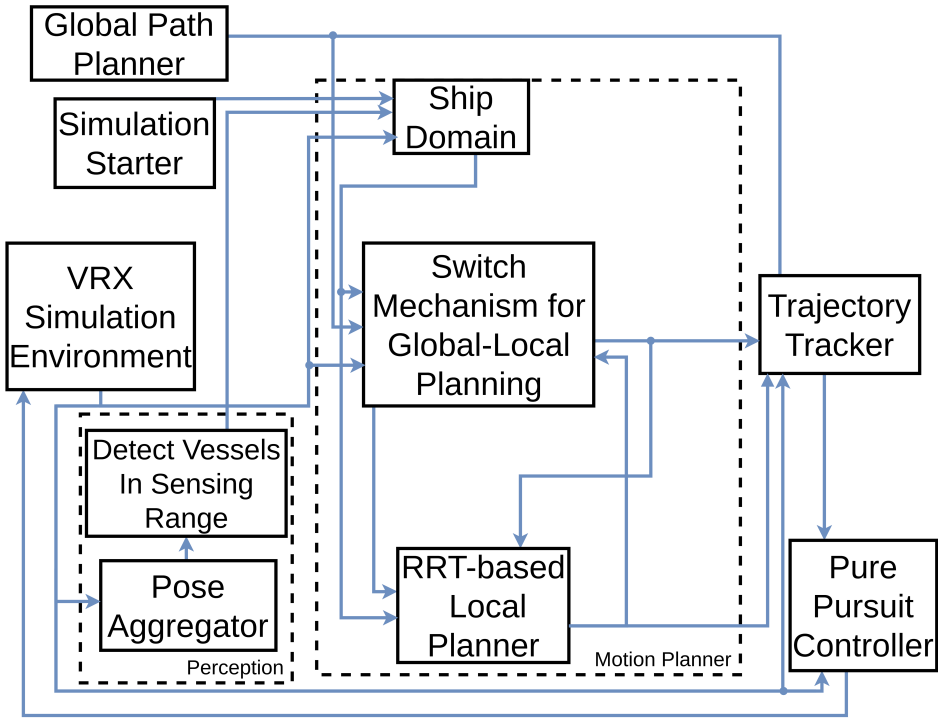


Figure 3.3: Simplified block diagram of the developed Gazebo-ROS infrastructure.

To address the challenge of spawning and operating multiple vessels autonomously, we adopted a unique approach for each ROS node we implemented. Since the number of vessels in the simulation environment varies depending on the scenario, we designed

the ROS nodes to work simultaneously for every vessel in the simulation environment, regardless of how many vessels are apparent. To achieve this, we designed a Node_Class responsible for performing the necessary processes for a single vessel. Additionally, in the main objects of every ROS node, we define an object_list to store the objects assigned to every vessel. Finally, to ensure that the ROS nodes work for any number of vessel, we used multi-threading to simultaneously run each object.

3.2.1 Initializing the Simulation Environment

To initialize the simulation environment, we created a Python script called "Simulation-Starter". This script sets up various attributes of the simulation world, such as the sea surface area and the GPS origin point of the map. It also places the vessels according to the AIS data, sets ROS parameters for the vessel details from the AIS data and calls the launch file to start the Gazebo simulation engine. The pseudo-code of the `Simulation Starter` script is given in Algorithm 2.

Algorithm 2: `Simulation-Starter`

Input: $config\_file, global\_waypoints\_file$, $simulation\_world$
Output: $number\_of\_vessels$, $simulation\_world$, $vessel\_details$

$number\_of\_vessels \leftarrow config\_file.number\_of\_vessels$
$ocean\_size \leftarrow global\_waypoints\_file.ocean\_size$
$origin\_point\_coordinates \leftarrow global\_waypoints\_file.origin\_coordinates$
Edit the $simulation\_world$ attributes such as $ocean\_size$ and $origin\_point\_coordinates$
Save the $simulation\_world$ to simulation_world.world file
foreach $vessel \in global\_waypoints\_file$ **do**
    Read the details of $vessel$ from $global\_waypoints\_file$
    Set $vessel\_details$ ROS parameter
Launch the vrx_multivessel_parametric.launch file `// This launch file launches the Gazebo simulation environment`

3.2.2 Perception Nodes

In default settings, the VRX simulation environment provides a 3-D LIDAR sensor and cameras on the Wave Adaptive Marine Vehicle (WAMV). However, using these sensors for perception requires significant processing power. To address this issue, we developed an approach that utilizes GPS sensors of vessels and proximity-based measurements instead.

Algorithm 3: `Pose Aggregator`

Input: *number_of_vessels*
Output: *gps_list*,*imu_list*, *vessel_pair_list*

```
1  for i ← 1 to number_of_vessels do
2      pose_object_list.append(Vessel_Subscriber_Class(i))
3  Generate vessel_pair_list
4  Write vessel_pair_list to vessel_pair_list.json file
5  while True do
6      gps_list = []
7      imu_list = []
8      foreach pose_object ∈ pose_object_list do
9          gps_list.append(pose_object.latitude)
10         gps_list.append(pose_object.longitude)
11         imu_list.append(pose_object.orientation)
12     Publish gps_list to simulation/gps_list ROS topic
13     Publish imu_list to simulation/imu_list ROS topic
```

Our perception stack consists of two ROS nodes: the `Pose Aggregator` node and the `Detect Vessels In Sensing Range` node.

The `Pose Aggregator` node aggregates every vessel's pose information and publishes it to two ROS topics, /gps_list and /imu_list. Algorithm 3 provides the pseudo-code of this ROS node. In line 3 of Algorithm 3, the vessel_pair_list is generated. This list contains all possible vessel combinations of 2 in the simulation environment and is necessary for the `Detect Vessels In Sensing Range` node. Also, because this node is designed to work multi-threaded, the vessel_pair_list is a list of lists. Each list inside of the vessel_pair_list contains the vessel pairs that a thread will be handling during the simulation.

The `Detect Vessels In Sensing Range` node publishes information about vessels within the perception range to the /vesselX/perception ROS topic for every vessel in the simulation environment. The pseudo-code of this ROS node is given in Algorithm 4. As the number of vessels can vary, perception information for each vessel will be published to their respective ROS topics.

Algorithm 4: `Detect Vessels In Sensing Range`

Input: *perception_gps_list*,*perception_imu_list*, *number_of_vessels*, *vessel_pair_list*, *sensing_range*

Output: *vessel_1_perception*, *vessel_2_perception*,$\cdots$, *vessel_X_perception*

```
while True do
    i ← 0
    while i < |vessel_pair_list| do
        vessel_pair1 ← vessel_pair_list[i]
        vessel_pair2 ← vessel_pair_list[i + 1]
        d ← distance(vessel_pair1, vessel_pair2)
        if d < sensing_range then
            perception_indexes.append(vessel_pair1)
            perception_indexes.append(vessel_pair2)
        i ← i + 2
    vessel_index ← 1
    perception_message_list ← []
    while vessel_index <= number_of_vessels do
        perception_msg ← Generate_Perception_Msg(vessel_index,
          perception_indexes)
        perception_message_list .append(perception_msg)
        vessel_index ← vessel_index + 1
    j ← 1
    while j <= number_of_vessels do
        topic_name ← string("vessel_",j,"_perception")
        perception_msg ← perception_message_list[j − 1]
        Publish perception_msg to the ROS topic topic_name
        j ← j + 1
```

3.2.3 Ship Domain Node

To determine vessel encounters, we propose a circular ship domain. The coefficients for calculating the radius of the ship domain can be assigned from the simulation's configurations file. In default, we calculate the radius of ship domain for vessels by multiplying the coefficient wit the vessels length. The `Ship domain` ROS node performs two processes in each iteration. Firstly, it calculates which vessels within the perception range of the vessel are inside the vessel's ship domain. Secondly, it publishes information about these vessels to the ship domain ROS topic of the vessel. Algorithm 5 provides the pseudo-code for the `Ship Domain` ROS node.

Algorithm 5: `Ship Domain`

Input: *vessel_n_perception*, *vessel_n_gps*, *sensing_range*
Output: vessel_n_ship_domain

```
1  while True do
       // Calculate which vessels are inside ship domain
2      ship_domain_list ← []
3      for i ← 0 to |vessel_n_perception| do
4          ts_pos ← vessel_n_perception[i]
5          os_pos ← vessel_n_gps
6          d ← distance(ts_pos, os_pos)
7          if d < sensing_range then
8              ship_domain_list.append(ts_pos)
9      Publish ship_domain_list to vessel_n_ship_domain ROS topic
```

3.2.4 Switch Mechanism Node

The `Switch Mechanism` ROS node controls the switching between the local path planner and the global path planner by changing the value of the vessel_n_is_on_global_path ROS topic. If this ROS topic is set to 0, it means that the vessel is on a local path. If the ROS topic is set to 1, it means that the vessel is on a global path. If the ROS node is set to 2, it means that the `Local Path Planner` ROS node is currently generating a local path. The pseudo-code of the `Switch Mechanism` ROS node is shown in Algorithm 6.

In line 2 of Algorithm 6, the number of vessels inside the ship domain is determined.

In lines 3-4, an encounter is considered to have occurred if there are any vessels inside the ship domain. The local path arguments to invoke the local path planner are published in line 8. Until the local path is returned by the local planner, vessel_n_is_on_global_path is set to 2 in order to send this information to the `Pure Pursuit Controller` ROS node. As seen in lines 15-19, the path is switched to the local path after the local path is generated. The local path is then being tracked until the local goal point is reached.

Figure 3.4 shows an example scenario of an encounter. If a target vessel is inside a vessel's ship domain, the switch mechanism evaluates it as an encounter. To generate a COLREGs-compliant local path, virtual obstacles are placed for every vessel inside the ship domain into the local path bounding box. The local goal point is defined to be as close as possible to the current global waypoint in order to make the local path head towards it. The local path is generated within these constraints and followed until the local goal point is reached.

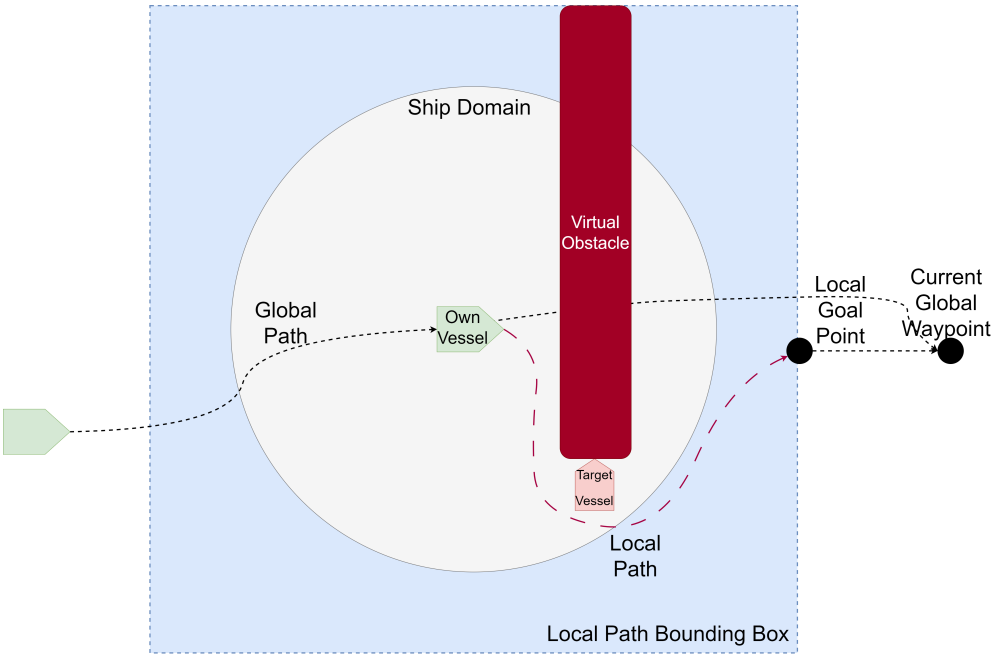


Figure 3.4: Example scenario of a vessel encounter.

3.2.5 Local Path Planner Node

To implement a vessel motion planning that complies with the COLREGs, the RRT path planning algorithm is chosen. During tracking of the global path or local path, the local path planner stays on standby. When the switch mechanism activates the

Algorithm 6: `Switch Mechanism`

Input: *vessel_n_ship_domain*, *vessel_n_gps*, *vessel_n_local_path*
Output: vessel_n_local_path_args, vessel_n_is_on_global_path

while True **do**
  `// Check encounter`
  $l \leftarrow |vessel\_n\_ship\_domain|$
  if $l > 0$ **then**
    $encounter \leftarrow True$
  else
    $encounter \leftarrow False$
    $vessel\_on\_global\_path \leftarrow 1$
  Publish *vessel_on_global_path* to vessel_n_is_on_global_path ROS topic
  if encounter = True **then**
    $vessel\_on\_global\_path \leftarrow 2$
    Publish *vessel_on_global_path* to vessel_n_is_on_global_path ROS topic
    $loca\_path\_arguments \leftarrow$ Generate_Local_Path_Args()
    Publish *local_path_arguments* to vessel_n_local_path_args ROS topic
    $local\_path\_received \leftarrow False$
    while $local\_path\_received$ = False **do**
      `// Wait for local path node to publish local path`
      $l\_local\_path \leftarrow |vessel\_n\_local\_path|$
      if $l\_local\_path > 0$ **then**
        $local\_path\_received \leftarrow True$
        $local\_goal \leftarrow$ generate_local_goal()
    $dist\_local\_goal \leftarrow$ distance$(vessel_n_gps, local_goal)$
    while (Local goal point is not reached) and (local path is not interfered) **do**
      $vessel\_on\_global\_path \leftarrow 0$
      Publish *vessel_on_global_path* to vessel_n_is_on_global_path ROS topic
    $vessel\_on\_global\_path \leftarrow 1$
    Publish *vessel_on_global_path* to vessel_n_is_on_global_path ROS topic

local path planner to generate a new path, it generates the virtual obstacles for other vessels inside the local path planning bounding box. The RRT algorithm is executed to find the COLREGs-compliant path for the encounter situation, and it is published to the vessel_n/local_path ROS topic. The `Trajectory Tracker` ROS node can then make the vessel controller track the local path and the `Switch Mechanism` ROS node can check for interference to the local path by other vessels. The pseudo-code of the `Local Path Planner` ROS node can be seen in Algorithm 7.

Algorithm 7: `Local Path Planner`

Input: *vessel_n_local_path_args*, *vessel_n_ship_domain*, *vessel_n_gps*,*global_path*, *last_reached_global_waypoint*
Output: vessel_n_local_path

```
1  while True do
       // Check generate local path topic
2      args_len ← |vessel_n_local_path_args|
3      if args_len > 0 then
           // Generate local path
4          foreach vessel ∈ vessel_n_ship_domain do
5              Calculate relative_angle of vessel
6              if 5 < relative_angle < 112.5 then
7                  Place virtual obstacles along the dead ahead axis (+x) of vessel.
8              if (5 > relative_angle) or (355 < relative_angle) then
9                  Place virtual obstacles along the starboard abeam (+y) of vessel.
10         Calculate local_goal_point using vessel_n_gps, global_path,
            last_reached_global_waypoint
11         local_path ← Generate_RRT_path(local_goal_point)
12         Smooth local_path by removing unnecessary waypoints
13         Publish local_path to vessel_n_local_path ROS topic
```

In Algorithm 7, lines 6 and 8 compare the relative angle of a vessel in ship domain to numerical values in order to determine where the virtual obstacles will be placed, as defined Figure 2.2 based on the COLREGs. If a target vessel is in the vessel's ship domain and its relative angle falls between 5 and 112.5 degrees, the encounter situation complies with the COLREG Rule 15. Therefore, virtual obstacles are placed along the dead-ahead axis of the target vessel in order to make sure that the RRT algorithm generate a COLREGs-compliant route. Similarly, if a target vessel is in the vessel's ship

domain and its relative angle is smaller than 5 degrees or larger than 355 degrees, virtual obstacles are placed along the starboard axis of the target vessel. This is because the encounter situation complies with either Rule 14 or Rule 13, depending on the velocities and headings of the vessels. And regardless of the velocities and headings of the vessels, this method ensures that the RRT algorithm provides a COLREGs-compliant route. A similar virtual obstacle method is used in [Chiang and Tapia, 2018].

In line 12 of Algorithm 7, the generated RRT path is smoothed by removing the unnecessary waypoints where no obstacles are present in between. A similar path smoothing method is used in [Zhang et al., 2022].

3.2.6 Global Path Planner Node

The `Global Path Planner` ROS node is designed to implement either real-world AIS data or manually generated waypoints. Since the global waypoints for vessels are not expected to change frequently, we decided to utilize the ROS Parameter Server. This ROS node reads from the global_waypoints.json file and sets the vessel_n/global_waypoints ROS parameters to establish the global waypoints for vessels. Additionaly, another set of ROS parameters is defined as vessel_n/last_reached_global_waypoint, which serves as a pointer for the current waypoint.

3.2.7 Trajectory Tracker Node

The `Trajectory Tracker` ROS node is responsible for determining the current waypoint that a vessel should navigate to. It takes the global path, local path and vessel_n_is_on_global_path variable as inputs. The output of this node is the current_waypoint variable, which is used by the `Pure Pursuit Controller` ROS node to generate the actuator signals. The pseudo-code of this node is given in Algorithm 8.

In Algorithm 8, lines 3-12 describe the local path tracking procedure. If the switch mechanism returns 0, it means that the vessel should be on the local path. The local waypoints are then tracked one by one, as described in lines 6-12. If the switch mechanism returns 1, it indicates that the vessel should be on the global path, as described in lines 13-22. The ROS parameter vessel_n_last_reached_global_waypoint is defined

to keep track of the global path by storing the index of the last reached global waypoint. By using this ROS parameter, the algorithm can switch between local and global paths without losing the information about where it left off the local path.

3.2.8 Pure Pursuit Controller Node

The `Pure Pursuit Controller` ROS node generates the actuator signals for a vessel to reach to its current waypoint. Because of the simplicity of the algorithm, the Pure Pursuit controller algorithm is chosen [Paden et al., 2016]. With a constant linear velocity, the pure pursuit controls the steering of the thrusters.

3.2.9 Overall ROS System

Our ROS infrastructure is built using the ROS nodes we described. The block diagram of the proposed ROS infrastructure is shown in Figure 3.5. In this block diagram, ROS topics are depicted in blue, ROS parameters are depicted in green. The ROS nodes are represented as white squares with black borders. In some ROS topics and parameters, three dots indicate that they are parametrically generated for every vessel in the simulation environment.

The description for the ROS parameters and topics in this block diagram is as follows:

- The ROS parameters under /VesselX_details contain information about each vessel in the simulation environment, which are referenced from the AIS data in the Global_Waypoints file.
- The ROS parameters under /VesselX/Global_Trajectory contain the global path for each vessel, while the ROS parameters under /VesselX/Last_Reached_Global_Wp contain the index of the last reached global waypoint for each vessel.
- The ROS topics under /VesselX/Perception contain information about the other vessels within sensing range. The information about each vessel contains location, heading and details such as its length and width.
- The ROS topics under /VesselX/Vessels _In_Ship_Domain contain information

Algorithm 8: `Trajectory Tracker`

Input: *vessel_n_global_path*, *vessel_n_last_reached_global_waypoint*, *vessel_n_gps*, *vessel_n_local_path*, *vessel_n_is_on_global_path*

Output: vessel_n_current_waypoint

```
while True do
    switch ← vessel_n_is_on_global_path
    if switch = 0 then
        // Find current local waypoint
        wp_indx ← 0
        l ← |vessel_n_local_path|
        while wp_indx < l do
            waypoint ← vessel_n_local_path[wp_indx]
            pos ← vessel_n_gps
            d ← distance(waypoint, pos)
            if d < reaching_threshold then
                wp_indx ← wp_indx + 1
            Publish waypoint to vessel_n_current_waypoint ROS topic
    if switch = 1 then
        // Find current global waypoint
        path ← vessel_n_global_path
        index ← vessel_n_last_reached_global_waypoint
        waypoint ← path[index]
        pos ← vessel_n_gps
        d ← distance(waypoint,pos)
        if d > reaching_threshold then
            index ← index + 1
            vessel_n_last_reached_global_waypoint ← index
        Publish waypoint to vessel_n_current_waypoint ROS topic
    if switch = 2 then
        Local waypoint is being generated. Track the path that was already
          being tracked.
```

about the other vessels inside that vessel's ship domain. The information about each vessel contains location, heading and details such as its length and width.

- The ROS topics under /VesselX/Is_On_Global_Path contain information about the vessel's current path, indicating whether it is on the local path, or global path, or if a local path is being generated at the moment.
- The ROS topics under /VesselX/Local_Path_Args contain the arguments generated by by the switch mechanism in order to generate a local path for the vessel.
- The ROS topics under /VesselX/Local_Path contain the local path generated by the local planner.
- The ROS topics under /VesselX/Current_Waypoint contain the current waypoint that the vessel is supposed to be navigating to follow the local or global path.
- The ROS topics described as "Actuator Signals" are the thrust velocity and heading of the vessels in the simulation environment.
- The ROS topics described as "Topics of GPS and IMU sensors of every vessel" are the ROS topics for the GPS and IMU sensors on the vessels in the simulation environment.
- The ROS topics under under /Simulation/GPS_List and /Simulation/IMU_List contain the list of GPS and IMU information of every vessel in the simulation environment. This information is collected and aggregated by the `Pose Aggregator` ROS node and used by the `Detect Vessels In Sensing Range` ROS node.

3.3 Access to the Source Code

Our goal with our work is to build an open-source simulation environment where researchers can test their unmanned surface vehicle motion planning algorithms in a realistically simulated maritime traffic. Therefore, we have decided to maintain this project in a Git repository. Researchers will have access not only to the source code, but also to more detailed information about the repository's files. Instructions of how to use this project will also be available in the repository [Bayrak and Bayram, 2023b].

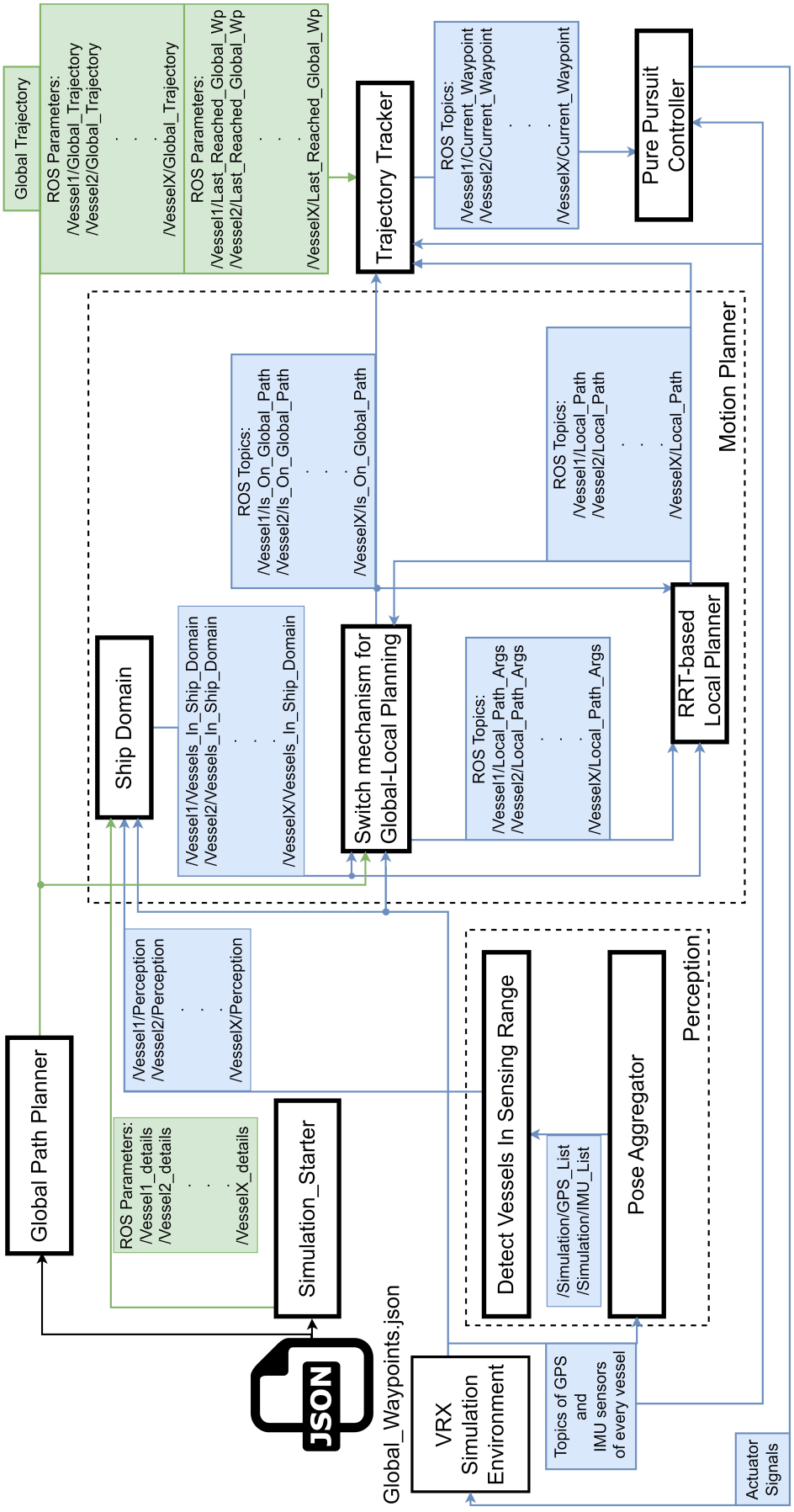


Figure 3.5: Block diagram of the developed ROS infrastructure.

CHAPTER 4: AIS DATA

AIS is a widely-used tracking system for vessels. It can be used for the vessels to monitor the maritime traffic and to be seen by other vessels. Also, it is used by coast guards and government agencies responsible for maritime traffic around the world for managing the vessel traffic from AIS base stations. By utilizing AIS data, it is possible to simulate a maritime traffic similar to real-world conditions and monitor current and historical vessel traffic. In this chapter, we mention the AIS data source we employed for our research in Section 4.1 and we mention our MATLAB infrastructure designed to retrieve smaller and scalable AIS data for our simulation in Section 4.2.

4.1 AIS Data Source

As mentioned above, real-world AIS data is necessary to set up a realistic maritime simulation environment. To access to this data, there are several sources online. However, some of these AIS data sources may require payment or involve more complicated harvesting methods. Although these sources are possible to utilize for our purpose, we decided to employ an easier, user friendly, fast and free of charge method to access to AIS data. Therefore, we chose to utilize AIS data from the marinecadastre.gov website [Office for Coastal Management, 2023] for our simulation environment. This website provides daily AIS data from U.S. coastal waters, which is freely accessible.

4.2 Filtering AIS Data

In this subsection, we will discuss the method we developed to retrieve a clip of AIS data from the daily AIS data downloaded from the website marinecadastre.gov. To achieve this, we developed a MATLAB infrastructure consisting of four functions. Two of these functions, Generate_Quadrangle_Table and Generate_Grid_Table_With_Hours, are used for the discretization of a bigger portion of AIS data. The other two of these functions, Filter_AIS_Data_of_Given_Quadrangle and Generate_Global_Waypoints_JSON, are designed to select a smaller portion from that discretized AIS data. The block diagram of the entire process of filtering and clipping AIS data is shown in Figure 4.1.

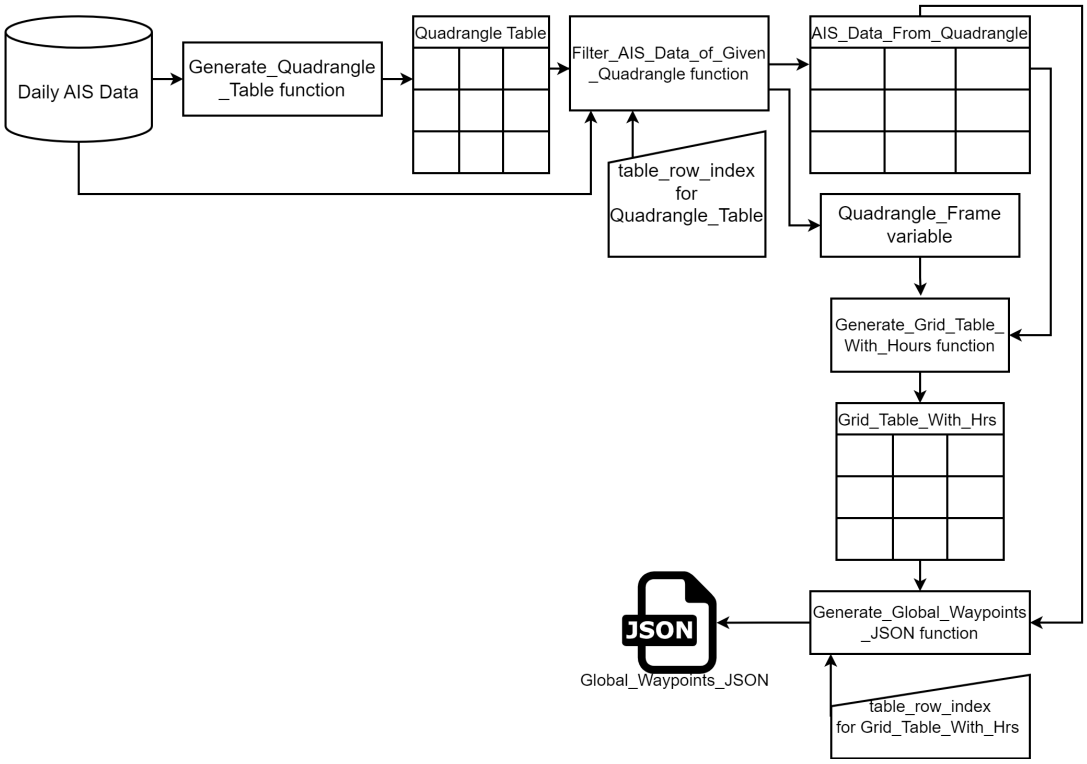


Figure 4.1: Block diagram of filtering and clipping process of AIS data.

First, we download the AIS data of one day from the website, and then we discretize the entire AIS data into quadrangles using the Generate_Quadrangle_Table function. In this context, a quadrangle is a bounding box defined by the latitude and longitude of the top right and bottom left points. The unit of measurement for this function is degrees.

Upon executing the Generate_Quadrangle_Table function, we get a MATLAB table named "Quadrangle Table". The columns of this MATLAB table and the descriptions of these columns are as follows:

- "Top Right Latitude and Longitude" and "Bottom Left Latitude and Longitude" columns define the bounding box of the quadrangle.
- "Number of AIS signals" represents the total amount of AIS signals received from that bounding box within 24 hours.
- "Number of Unique Vessels" represents the number of unique vessels that transmitted any AIS signal within the quadrangle.

- "AIS Index Array" stores the indexes of the AIS signals from the daily AIS data.
- "Table Row Index" column is used to maintain consistent indexing for rows even if the table is sorted by any value.

After using the Generate_Quadrangle_Table function, we can select the quadrangle from which we want to extract our tile. A tile, in this context, refers to a smaller area compared to a quadrangle. The area of a tile is measured in kilometers. Moreover, the time interval of a tile can be less than 24 hours depending on the simulation duration desired by the user. For this function, we have set 25 km^2 and 1 hour as the arguments. The "Grid Table With Hour" is a MATLAB table similar to the Quadrangle Table. The columns of this MATLAB table and their descriptions are given as follows:

- Top Right Latitude and Longitude and Bottom Left Latitude and Longitude columns define the bounding box of the quadrangle.
- "Time Minimum" and "Time Maximum" indicate the time interval for the tile.
- "Number of AIS signals" means the amount of AIS signals that are received from that bounding box along the defined time interval.
- "Number of Unique Vessels" means the number of unique vessels that transmitted any AIS signal inside that quadrangle.
- "AIS Signal Per Vessel" stores the average number of AIS data for each vessel in the tile.
- "Distance Traveled" stores the total traveled distance by the vessels inside the tile. This column is necessary to evaluate whether the vessels are stationary inside the tile or not.
- "AIS Index Array" stores the signals' indexes from daily AIS data.
- "Table Row Index" column is defined to have a consistent indexing for rows even if the table is sorted by any value.

Using this table, it is possible to visualize the AIS data of a tile by plotting it with our function called "Print_AIS_Data_From_Quadrangle_Table". With the help of such

a visualization method, users can decide which tile of AIS data they want to have for their simulation environment. After deciding, "Generate_Global_Waypoints_JSON" function generates a JSON file consisting of AIS data from the given tile index as an argument.

4.3 Example AIS Data

Using our proposed method for filtering daily AIS data into a smaller scale, users can generate a file containing global waypoints for a simulation scenario. To demonstrate, we downloaded the AIS data of January 5th, 2022 from the mentioned website. Next, we executed the Generate_Quadrangle_Table function to discretize the AIS data into quadrangles on the world map. We used the name of the daily AIS data file and the value "1" for the edges' length in degrees as arguments for this function.

After obtaining the Quadrangle_Table, we selected the quadrangle indexed by number 2516, which has a bounding box defined by the top right point's latitude and longitude value of 19,-64 and bottom left point's latitude and longitude value of 18,-65.

To discretize this quadrangle into tiles, we used the Generate_Grid_Table_With_Hours function with arguments of "1" for the edge length for each tile in kilometers and "1" for the one hour time interval for each tile. The output of this function is the Grid_Table_With_Hrs table. By sorting the rows of this table by values in columns such as Distance_Traveled and Distance_Traveled_Per_Vessel, we can find the tiles that contain active marine traffic.

Figure 4.2 provides an example of a tiles AIS data plot. The bounding box of this AIS data is defined by the top right point's latitude and longitude value of 18.27715,-64.94548 and bottom left points latitude and longitude value of 18.23197,-64.99276. The time interval for this AIS data is defined by the starting time of 05-Jan-2022 18:00:00 and the end time of 05-Jan-2022 19:00:00. In this figure, every vessel is represented by a different color.

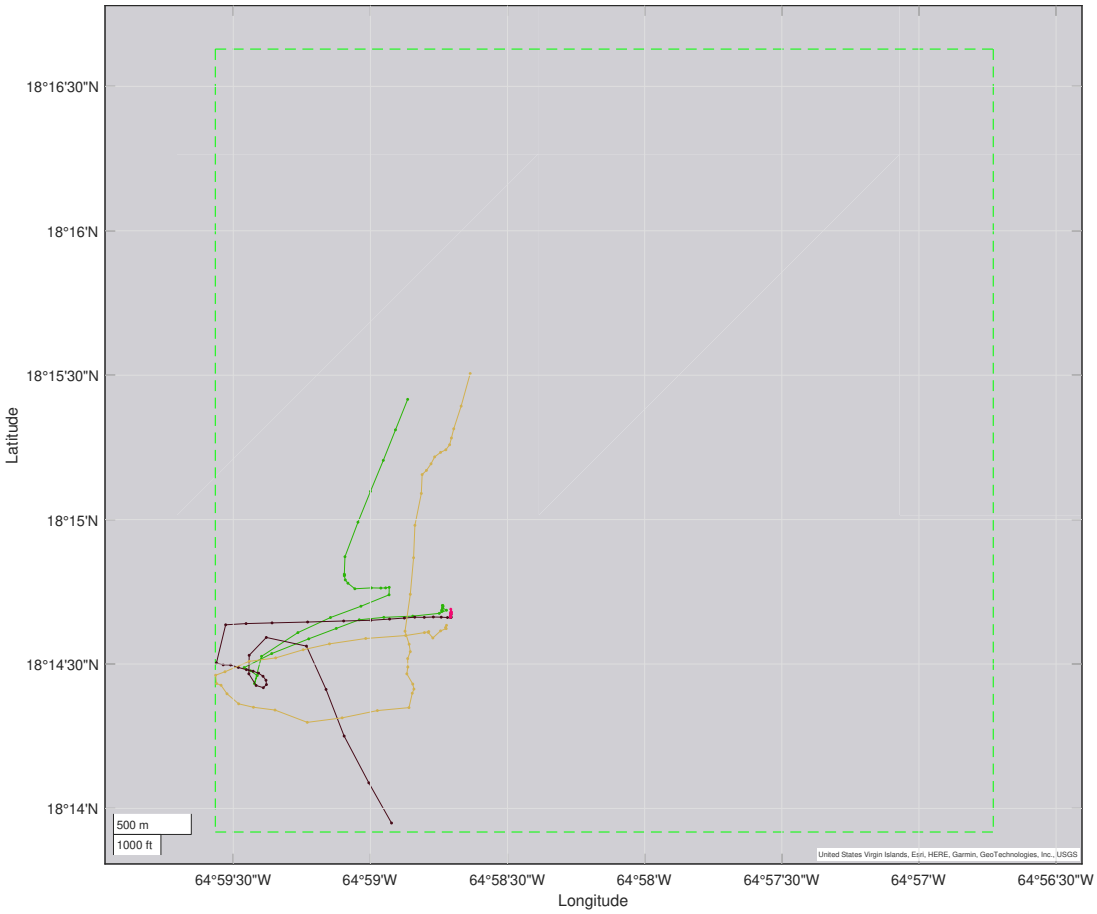


Figure 4.2: Example AIS data from the described coordinates and time.

CHAPTER 5: SIMULATIONS

To assess effectiveness of our proposed non-prioritized motion planning algorithm, we conducted simulations within the framework of our multi-vessel simulator described in Chapter 3. These simulations utilized historical AIS data, as discussed in Chapter 4, from which we extracted a distinct AIS data clip. The visual representation of this AIS data is presented in the top-down view showcased in Figure 5.1. The bounding box of the AIS data used for simulations is defined by the top right point's latitude and longitude value of 18.367510097436337,-64.614884232659918 and bottom left point's latitude and longitude value of 18.322330743229294, -64.662183019787619. The time interval for this AIS data is defined by the starting time of 05-Jan-2022 20:00:00 and the end time of 05-Jan-2022 21:00:00.

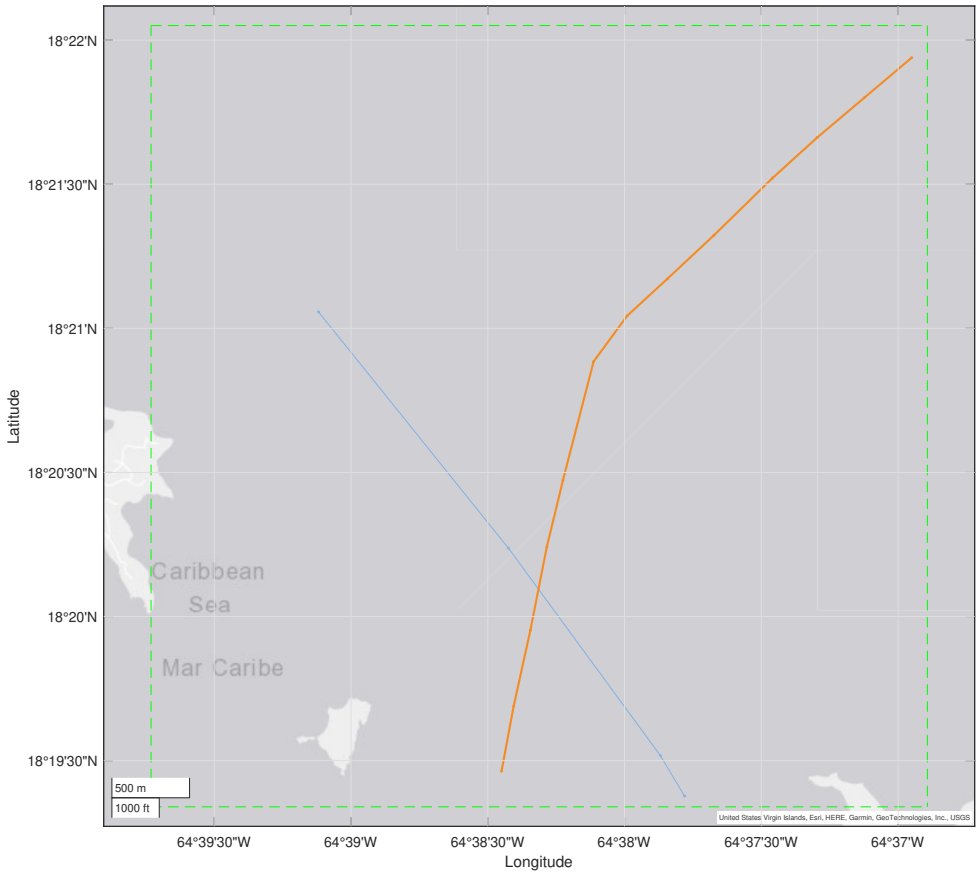


Figure 5.1: Example AIS data from the described coordinates and time. Vessel-1 is depicted in blue, Vessel-2 is depicted in orange color.

In Figure 5.1, Vessel-1 is depicted in blue and Vessel-2 is depicted in orange color. In this AIS data, the length of Vessel-1 is 17 meters and the length of Vessel-2 is

125 meters. According to these values, the simulation settings are calculated in 5.1. Explanation of these values can be found in 2.5.

5.1 Simulation Settings

The simulation settings described in Section 2.5 for the vessels are specified as follows:

- The Ship Domain Radius is calculated as 8 times the length, resulting in 136 meters for Vessel-1 and 1000 meters for Vessel-2.
- Both vessels have a Perception Range of 1000 meters.
- The Local Path Bounding Box is determined as 1.2 times the ship domain radius, giving 163 meters for Vessel-1 and 1200 meters for Vessel-2.
- The Virtual Obstacle Radius is calculated as 3 times the length of the ships, resulting in 51 meters for Vessel-1 and 375 meters for Vessel-2.

Furthermore, the software and hardware specifications for the computer on which the simulations are conducted are provided as follows:

- Processor: Intel Core I5 10300H at 2.5 GigaHertz
- Random Access Memory: 16 Gigabyte DDR4 at 4000 GigaHertz
- Graphics Card: NVIDIA GeForce GTX 1650 Mobile with 4 Gigabytes of GDDR6 Random Access Memory
- Operating System: Ubuntu 20.04 LTS
- ROS: ROS Noetic 1.15.15
- Gazebo: Gazebo 11.12.0

5.2 Performance Metrics

To evaluate performance during the simulation runs, we have considered the following performance metrics:

- Distance Traveled by Vessels: We measure the distance traveled by each vessel and our USV within the simulation environment during a run. This metric helps us to understand the deviation from the original path caused by other vessels in the simulated environment.

- Proximity: We calculate the distance between our USV and other vessels within the simulation environment. This measurement is obtained from both scenarios: when employing the non-prioritized motion planning algorithm and when utilizing the COLREGs-compliant motion planning designed for other vessels. This assessment is conducted throughout the entire simulation run.

5.3 Baseline Approach

As the baseline approach, we executed simulation runs where the USV employed the same COLREGs-compliant motion planning algorithm as the other vessels within the simulation environment. This allowed us to discern the distinctions between a standard COLREGs-compliant motion planning approach and our non-prioritized motion planning algorithm. The baseline approach is the COLREGs-compliant motion planning for vessels as described in Chapter 3.

5.4 Simulation Results and Comparison

We conducted two simulations in which we employed the AIS data presented in Figure 5.1. In the first simulation, all vessels in the environment utilized the COLREGs-compliant motion planning algorithm designed for vessels. The purpose of the first simulation is to obtain data from the baseline approach, which will serve as a benchmark for evaluating our non-prioritized motion planning algorithm. Figure 5.2 illustrates that our USV closely approached other vessels during the simulation run. During the simulation run, the USV triggered Vessel-1 to re-plan its route by entering its ship domain at the 140th second of the run. Vessel-2 is triggered to re-plan its route at the 715th second by our USV.

For our second simulation, we implemented the non-prioritized motion planning algorithm for our USV. As the results demonstrate, our USV maintained a distance from other vessels and adjusted its path to reach its goal point without interfering with the

Table 5.1: Distance traveled by the vessels and proximity metrics during the simulation runs (all values are in meters)

		Distance Traveled	Proximity			
			Min.	Max.	Mean	Std. Dev.
Baseline Approach	Vessel-1	3920	101	1280	767	322
	Vessel-2	5433	160	5175	2313	1116
	USV	4409				
Non-Prioritized Motion Planning	Vessel-1	3889	179	1575	879	452
	Vessel-2	5532	1524	5090	2566	753
	USV	4562				

routes of other vessels. Figure 5.3 shows the proximity graph, illustrating how our non-prioritized USV cautiously navigated around other vessels during the simulation run. Importantly, our USV did not trigger any other vessel to re-plan their route, as the closest points between our USV and other vessels were outside the radius of their ship domains.

As shown in Table 5.1, our USV traveled a longer distance in the second run in which our USV had the non-prioritized motion planning algorithm. However, our USV had a more secure navigation as indicated by the proximity to other vessels in the simulation environment. In the scenario utilizing the baseline approach, the minimum distance observed between our USV and Vessel-1 was 101 meters, and between our USV and Vessel-2 was 160 meters. With the non-prioritized motion planning algorithm, our USV maintained longer minimum distances of 179 meters from Vessel-1 and 1524 meters from Vessel-2. In addition, employing the non-prioritized motion planning algorithm for our USV prevented encounter situations with other vessels.

While the non-prioritized motion planning algorithm is intended to avoid encounter situations, it does not necessarily guarantee that our USV will always travel a longer distance compared to the baseline approach. Similarly, it does not ensure that other vessels, with which our USV does not have an encounter, should have a shorter distance traveled using our algorithm. This is because an encounter situation can potentially lead a vessel to take a shorter path towards its final goal point, resulting in faster navigation. As seen in the case of Vessel-2, it traveled 5433 meters of distance in the baseline approach. However, it traveled 5532 meters when our USV had the non-prioritized motion planning algorithm.

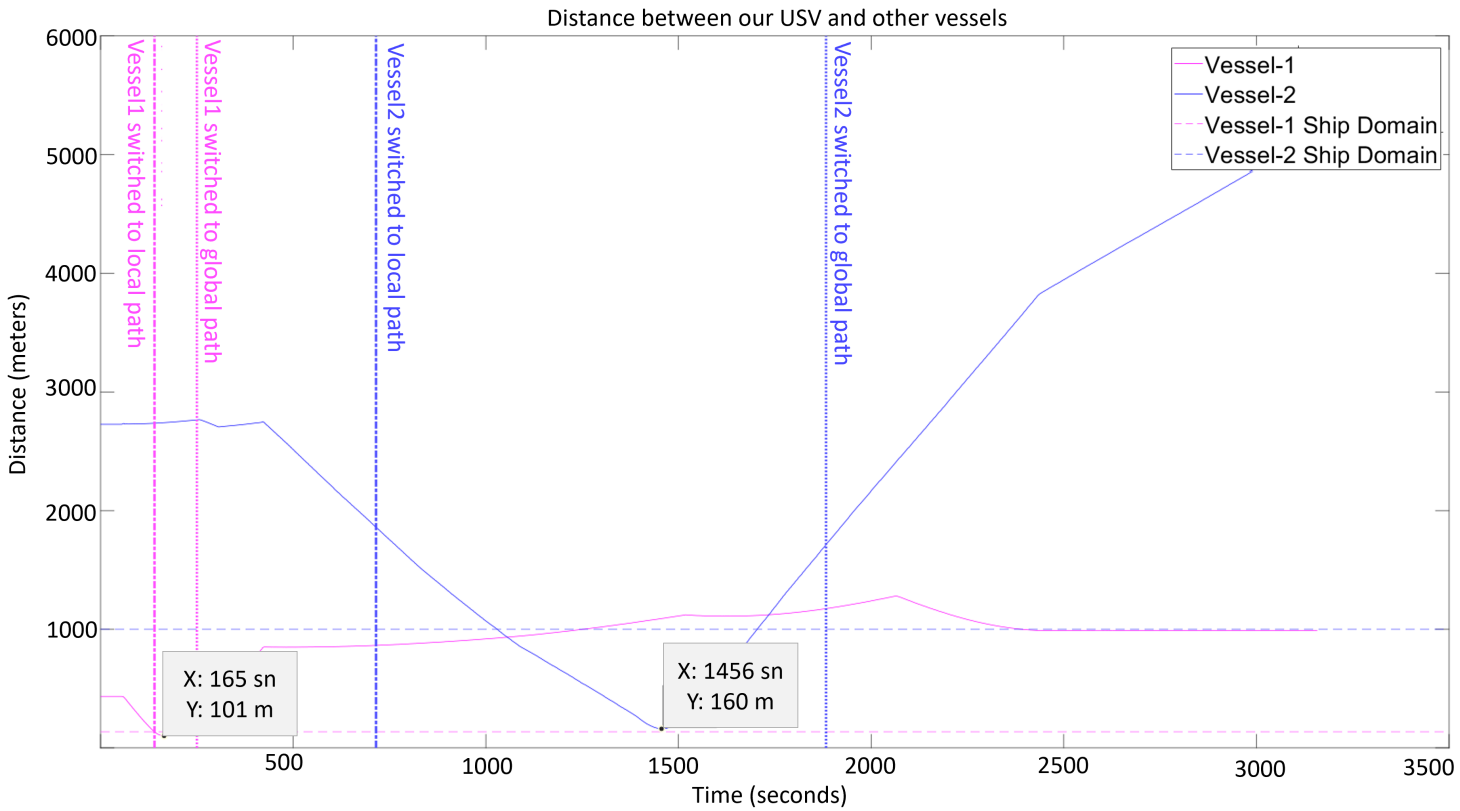


Figure 5.2: Proximity graph illustrating the distance between our USV and other vessels throughout the simulation run.

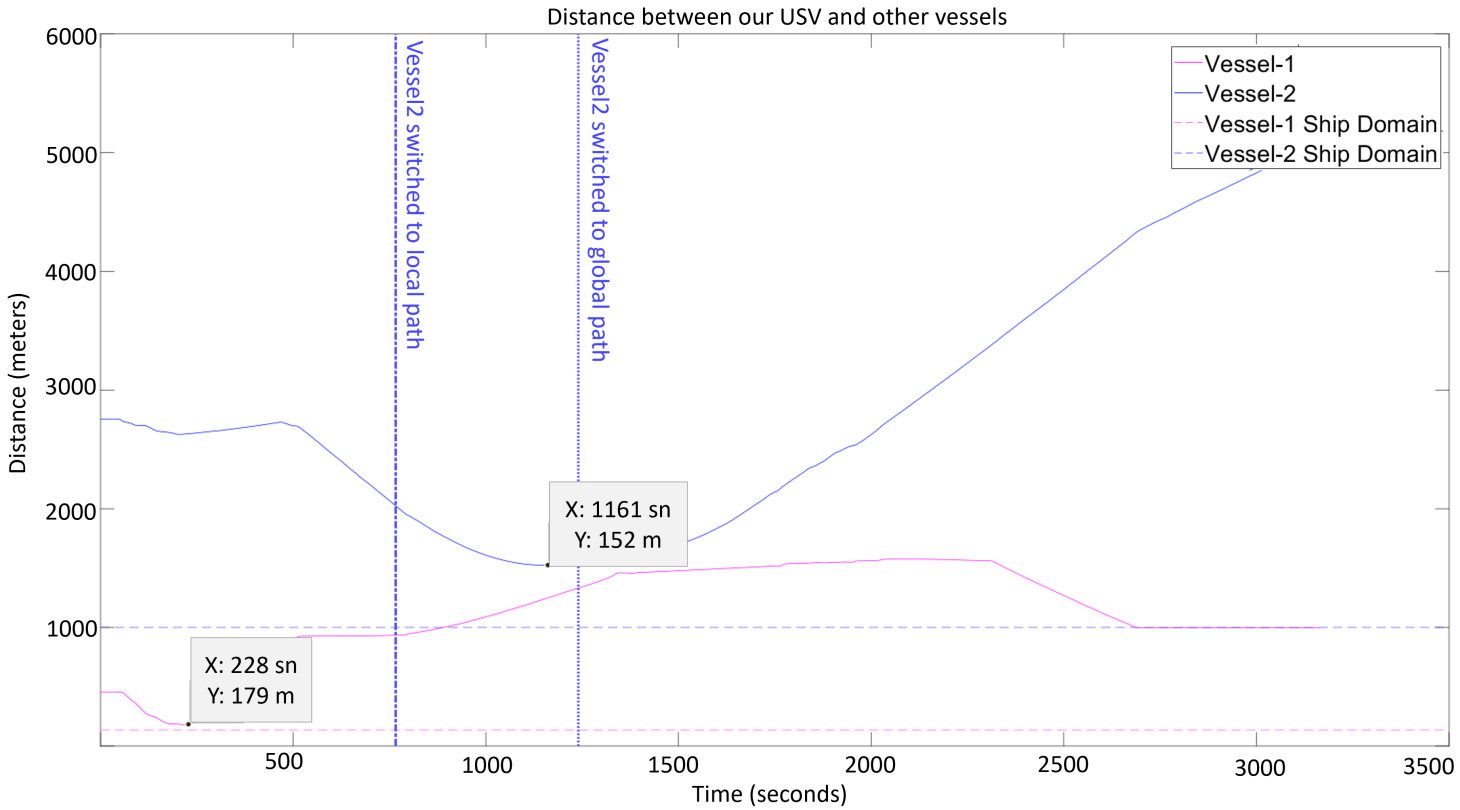


Figure 5.3: Proximity graph depicting the distance between our non-prioritized USV and other vessels throughout the simulation run.

CHAPTER 6: CONCLUSION

In this study, we have introduced a motion planning algorithm that complies with the COLREGs. The primary objective of this algorithm is to ensure that our USV does not interfere with the routes of other vessels, thereby mitigating the risk of endangering maritime traffic.

To achieve this, we made certain assumptions. Firstly, we assumed that our USV possesses long-distance perception capabilities, enabling it to detect other vessels from kilometers of distance. Additionally, circular ship domains are considered for the other vessels. With these assumptions, our low-priority motion planning algorithm is designed not to enter the ship domains of other vessels in a manner that could potentially lead to re-planning for those vessels.

By adhering to the COLREGs and avoiding any interference with the routes of other vessels, our motion planning algorithm aims to enhance the safety and efficiency of maritime navigation for autonomous surface vehicles.

To demonstrate the effectiveness of our algorithm, we performed simulation tests in a realistic marine traffic environment. For these simulations, we utilized historical AIS data to closely resemble real-world scenarios. This approach allows us to thoroughly validate the capabilities of our algorithm under conditions that mirror genuine maritime traffic situations. With these simulation tests, we can confidently assess the performance and reliability of our algorithm in dynamic and challenging marine traffic environments. We have presented that our non-prioritized motion planning algorithm can navigate our USV safely, much distant to other vessels in the environment and without invoking other vessels to re-plan their routes. The use of historical AIS data adds a valuable dimension to our research, ensuring that our algorithm is rigorously tested and prepared for real-life applications in autonomous surface vehicles.

In order to test our proposed motion planning approach and any other USV motion planning, we presented an open-source simulation environment for testing COLREGs-compliant USV motion planning algorithms. To achieve such a simulation environment,

we proposed and implemented a Gazebo-ROS infrastructure to navigate multiple vessels within the simulation environment.

We applied a proximity-based solution for the perception and ship domain. In order to achieve COLREGs-compliant path planning, we implemented an RRT-based local path planning algorithm in case of an vessel encounter. We also implemented a global path planner algorithm to simulate vessel navigation when there is no vessel encounter. We proposed a switch mechanism algorithm to switch between local and global path planners according to the encounters that vessels have. Trajectory tracker and pure pursuit controller algorithms were also implemented to enable the vessels in the simulation environment to track their paths successfully.

To simulate realistic maritime traffic, we decided to use publicly accessible AIS data. In order to achieve this, we proposed a programmatic approach to filter and analyze AIS data. This enables users to identify a subset of AIS data that is smaller in scale in terms of distance and time interval.

In terms of future developments for our non-prioritized motion planning algorithm, our primary objective is to enhance its capabilities to effectively plan routes within significantly denser maritime traffic environments. With our simulation environment, we have the capability of testing and developing our motion planning algorithm under various circumstances. In addition to simulation tests, we are considering to conduct field tests that our USV will employ the non-prioritized motion planning algorithm.

For future work about our multiple vessel simulation environment, there are many opportunities and extensions due to the structure of the simulation environment. ROS 2 migration is necessary for compatibility with VRX 2.0. Modeling realistic vessels for the simulation environment with hulls that match those in the provided AIS data would be appropriate for realistic vessel dynamics.

REFERENCES

[Akdağ et al., 2022] Akdağ, M., Solnør, P., and Johansen, T. A. (2022). Collaborative collision avoidance for maritime autonomous surface ships: A review. Ocean Eng., 250:110920.

[Bakdi et al., 2021] Bakdi, A., Glad, I. K., and Vanem, E. (2021). Testbed scenario design exploiting traffic big data for autonomous ship trials under multiple conflicts with collision/grounding risks and spatio-temporal dependencies. IEEE Trans. Intell. Transp. Syst., 22(12):7914–7930.

[Bayrak and Bayram, 2023a] Bayrak, M. and Bayram, H. (2023a). Colreg-compliant simulation environment for verifying usv motion planning algorithms. In Proceedings of MTS/IEEE OCEANS Conference, pages 1–10, Limerick, IE.

[Bayrak and Bayram, 2023b] Bayrak, M. and Bayram, H. (2023b). Multivessel simulation, https://github.com/fieldroboticslab/multivessel_simulation. Accessed: 2023-08-15.

[Beser and Yildirim, 2018] Beser, F. and Yildirim, T. (2018). COLREGs based path planning and bearing only obstacle avoidance for autonomous unmanned surface vehicles. Procedia computer science, 131:633–640.

[Bingham et al., 2019] Bingham, B., Aguero, C., McCarrin, M., Klamo, J., Malia, J., Allen, K., Lum, T., Rawson, M., and Waqar, R. (2019). Toward Maritime Robotic Simulation in Gazebo. In Proc. MTS/IEEE OCEANS Conf., Seattle, WA.

[Bolbot et al., 2022] Bolbot, V., Gkerekos, C., Theotokatos, G., and Boulougouris, E. (2022). Automatic traffic scenarios generation for autonomous ships collision avoidance system testing. Ocean Eng., 254:111309.

[Chai and Hassani, 2019] Chai, Y. and Hassani, V. (2019). Hybrid collision avoidance with moving obstacles. IFAC-PapersOnLine, 52(21):302–307.

[Chiang and Tapia, 2018] Chiang, H.-T. L. and Tapia, L. (2018). COLREG-RRT: An RRT-based COLREGS-compliant motion planner for surface vehicle navigation. IEEE Robot. Autom. Lett., 3(3):2024–2031.

[Cockcroft and Lameijer, 2011] Cockcroft, A. and Lameijer, J. (2011). A Guide to the Collision Avoidance Rules. Elsevier Science.

[European Maritime Safety Agency, 2022] European Maritime Safety Agency (2022). Annual overview of marine casualties and incidents 2022. EMSA Lisbon.

[Hagen et al., 2022] Hagen, I., Kufoalor, D., Johansen, T., and Brekke, E. (2022). Scenario-Based Model Predictive Control with Several Steps for COLREGS Compliant Ship Collision Avoidance. IFAC-PapersOnLine, 55(31):307–312.

[Hagen et al., 2018] Hagen, I. B., Kufoalor, D. K. M., Brekke, E. F., and Johansen, T. A. (2018). Mpc-based collision avoidance strategy for existing marine vessel guidance systems. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 7618–7623. IEEE.

[Office for Coastal Management, 2023] Office for Coastal Management (2023). 2023: Nationwide Automatic Identification System, https://www.fisheries.noaa.gov/inport/item/67336. Accessed: 2023-08-15.

[Paden et al., 2016] Paden, B., Čáp, M., Yong, S. Z., Yershov, D., and Frazzoli, E. (2016). A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Transactions on intelligent vehicles, 1(1):33–55.

[Peng et al., 2023] Peng, X., Han, F., Xia, G., Zhao, W., and Zhao, Y. (2023). Autonomous obstacle avoidance in crowded ocean environment based on colregs and pond. Journal of Marine Science and Engineering, 11(7):1320.

[Quigley et al., 2009] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A. Y., et al. (2009). ROS: an open-source Robot Operating System. In ICRA workshop on open source software, pages 1–5. Kobe, Japan.

[Rothmund et al., 2022] Rothmund, S. V., Tengesdal, T., Brekke, E. F., and Johansen, T. A. (2022). Intention modeling and inference for autonomous collision avoidance at sea. Ocean Eng., 266:113080.

[Singh et al., 2018] Singh, Y., Sharma, S., Sutton, R., Hatton, D., and Khan, A. (2018). A constrained A* approach towards optimal path planning for an unmanned surface vehicle in a maritime environment containing dynamic obstacles and ocean currents. Ocean Engineering, 169:187–201.

[Wang et al., 2021] Wang, D., Zhang, J., Jin, J., and Mao, X. (2021). Local collision avoidance algorithm for a unmanned surface vehicle based on steering maneuver considering COLREGs. IEEE Access, 9:49233–49248.

[Zaccone et al., 2019] Zaccone, R., Martelli, M., and Figari, M. (2019). A COLREG-compliant ship collision avoidance algorithm. In 2019 18th European Control Conference (ECC), pages 2530–2535. IEEE.

[Zhang et al., 2022] Zhang, J., Zhang, H., Liu, J., Wu, D., and Soares, C. G. (2022). A Two-Stage Path Planning Algorithm Based on Rapid-Exploring Random Tree for Ships Navigating in Multi-Obstacle Water Areas Considering COLREGs. J. Mar. Sci. Eng., 10(10):1441.

[Zhao et al., 2023] Zhao, L., Bai, Y., and Paik, J. K. (2023). Global-local hierarchical path planning scheme for unmanned surface vehicles under dynamically unforeseen environments. Ocean Engineering, 280:114750.