공학석사 학위논문

A Method for Hierarchical Route Planning of Small ships in Coastal Area Using Quadtree Chart

쿼드트리 해도를 활용한 연안 소형선의 계층적 경로 계획 방법

2023년 2월

서울대학교 대학원
조선해양공학과
정 동 근

A Method for Hierarchical Route Planning of Small ships in Coastal Area Using Quadtree Chart

지도 교수 노 명 일

이 논문을 공학석사 학위논문으로 제출함
2022년 10월

서울대학교 대학원
조선해양공학과
정 동 근

정동근의 공학석사 학위논문을 인준함
2023년 2월

위 원 장 _____ 김 태 완 _____
부위원장 _____ 노 명 일 _____
위 원 _____ 남 보 우 _____

Table of Contents

Figures

Tables

Abstract

A Method for Hierarchical Route Planning of Small ships in Coastal Area Using Quadtree Chart

Dong-Guen Jeong

Naval Architecture and Ocean Engineering

The Graduate School

Seoul National University

Route planning of vehicles uses different techniques for the route planning map and route planning algorithms according to the purpose of route planning. The small size of the ship, the coastal areas where the small ships are mainly operated, and the fact that individuals mainly operate them are major considerations for choosing a technique for the route planning map and route planning algorithm.

Electronic navigational charts are generally produced according to the standard form regulated by the International Hydrographic Organization (IHO). Objects on the electronic navigational chart are represented in vector format, making it difficult to use them directly for the route planning. In this study, a quadtree, which can reduce the number of node of the charts compared to regular grid charts by representing topography with non-uniform

grid sizes, was used to represent route planning charts. The quadtree chart can represent areas with complex and small obstacles with a high level of sophistication that small ships can pass through. In addition, since small ships frequently sail inside the marina, the centerline created using the Voronoi diagram was selected as the route planning charts to consider the characteristics of the marina.

For the route planning algorithm, Hierarchical Pathfinding A* (HPA*) was used. HPA* divides the chart hierarchically and sequentially proceeds the route planning according to the hierarchy. Moreover, a multi-step smoothing was applied to improve the quality of the route and enhance user experience. In this study, a route planning program considering various objects for sailing was developed using the quadtree, Voronoi diagram, and HPA* and it verified in various coastal areas.

Keywords: Route planning (경로 계획), Quadtree (쿼드트리), Hierarchical pathfinding A*, Voronoi diagram (보로노이 다이어그램), Small ships (소형선)

Student number: 2021-26237

1. Introduction

1.1. Research background

Route planning of a vehicle varies with purpose, as well as the shortest distance between a departure and a destination. Studies on route planning of ships have been conducted to simultaneously optimize diverse purposes such as fuel oil consumption (FOC), ocean weather data, and seakeeping performance for the economical operation of large commercial vessels. However, optimizing the route considering all the above data in a small ship was inappropriate because it consumes long route planning time and high memory usage. Accordingly, a route planning method suitable for a small ship has been required.

A small ship has a length of less than 12 m and mainly sails in the coastal area. Ministry of Oceans and Fisheries [1] In addition, it is mainly operated by ordinary people, not professional navigators or pilots. Therefore, it is important to plan a high-quality route that guarantees personal safety and enhances the user experience. Also, it is important to provide a route within a short period that does not cause inconvenience to the user.

There are various kinds of small obstacles in coastal areas where small ships mainly travel, and small ships freely navigate between small obstacles due to their small size. To accurately reflect this in route planning, a high-resolution route planning chart is required. Generally, route planning chart with a high-resolution increases route planning time and memory usage. In order to solve this problem, it is necessary to develop route planning chart that can represent marine objects in high resolution but has few nodes.

In addition, small ships frequently sail the marina in coastal areas. Rules and regulations of the marina vary depending on the marina, but small ships must observe the following two rules for safe navigation inside the marina. First, inside the marina, the density of ships and obstacles is high, so it is necessary to plan a route that maintains a safe distance from obstacles inside the marina. Second, when navigating the marina entrances, it is necessary to follow the port-to-port rule to ensure the safety of the route.

In this study, a route planning method for small ships was proposed considering the characteristics of small ships as mentioned above and rules of the marina.

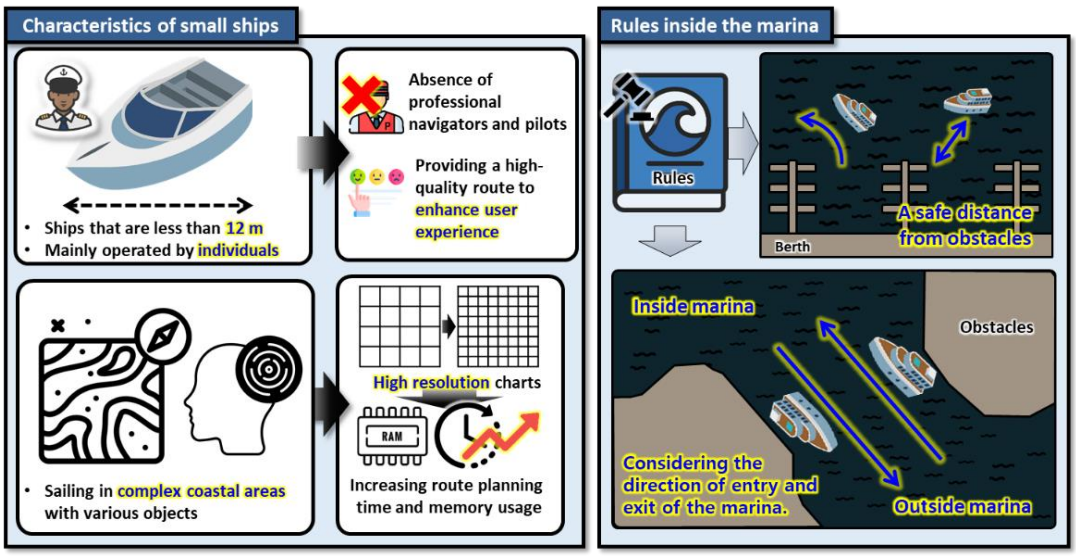


Figure 1. Research necessities

1.2. Related works

Route planning has been continuously developed in various fields, such as robotics and computer games. It is differentiated in various ways with the route planning method and how to represent the route planning map. The representation of a route planning map greatly influences the performance of route planning programs and the quality of routes. Amarel [2] Therefore, it is important to select an appropriate representation technique for a route planning map and a route planning method considering the purpose of route planning.

1.2.1. Representation techniques for route planning map

The representation techniques for the route planning map include a regular grid, quadtree, navigation mesh, Voronoi diagram, and visibility graph.

A regular grid representation consists of grids of a specific size. Each grid divides the space and contains information on the area covered by the grid. And the regular grid representation can describe the marine environment more accurately as the size of one grid becomes smaller. As the grid size decreases, the total number of grids inside the map increases, and the route planning time also increases. Therefore, it is important to set a grid of an appropriate size.

Quadtree representation can represent topography for route planning with a non-uniform node size. It can efficiently describe complex topography with the significantly reduced number of grids compared to the regular grid representation. Samet [3]

Navigation mesh is a representation method of expressing obstacles by dividing the movable area into convex polygons. Compared to the regular grid representation, the file

size of the navigation mesh map is smaller, and the memory usage required for route planning using the navigation mesh map is smaller. Cui et al. [4]

The Voronoi diagram is created by crossing the vertical bisectors of the site points constituting the polygon of the obstacle. The distances between the site point and other points inside the same region are shorter than the distances between the site point and other points in the other region. Fortune [5] Therefore, the edges of the Voronoi diagram are created at a distance from the obstacle, which is unsuitable for a route planning map for the shortest distance.

Lastly, the visibility graph is created by preserving only the edges that do not overlap with obstacles among the all edges connecting vertexes consisting of all obstacles in the chart and departure and destination. Thus, a new visibility graph needs to be created when a new departure and destination are given. As the number of obstacles increases, the time to create the visibility graph also increases, so it is suitable for route planning map of topography with a few obstacles. Welzl [6]

1.2.2. Route planning map methods

Route planning methods include a graph-based method, a sampling-based method, and a method using metaheuristic algorithms.

Graph-based methods plan routes based on graph-based maps for route planning. A grid is the same as a node of a graph in a grid representation, so the graph-based method includes grid-based methods. In the graph-based method, a route is constructed by selecting a node to move. The optimal route by the graph-based method is planned considering the distance and relationship between the nodes. Ghandi and Masehian [7] Representative examples of

graph-based methods are Dijkstra, A*, Hierarchical Pathfinding A* (HPA*), Theta*, and Jump Point Search (JPS).

Dijkstra is an algorithm that searches for a route with the shortest distance by repeating the node search until reaching the goal node. In Dijkstra, a node is searched in order of the cost of the node from the smallest. The cost of the node is calculated with the sum of the cost of the previous node and the cost function. The cost function generally uses Euclidean distance or Manhattan distance when aiming for the shortest distance. Dijkstra [8]

A* is an algorithm based on Dijkstra. Node is searched in order of the sum of the cost of the node and the heuristic function. The cost defines like Dijkstra. The heuristic function is usually defined as the Euclidean distance or Manhattan distance between the current node and the goal node. The heuristic function reduces the number of searched nodes. Therefore, A* can plan the route faster than Dijkstra, which searches the entire area. Peter Hart, Nils Nilsson [9]

HPA* is a route planning method from the general idea that when people plan routes between distant regions, they select the waypoints in a high-level map (e.g., states or cities), plan detailed routes accordingly, and then combine them. In order to plan a route with HPA*, a preprocessing process of creating a hierarchical map is required. Route planning using A* starts from high-level maps. Then, based on the route planning result of the high-level maps, the route is planned at the low-level maps, and the final route is created by integrating the routes at the low-level maps. The number of searched nodes can be reduced by route planning in multiple levels of maps, and as a result, it is possible to plan a route quickly with less memory usage than A*. Botea et al. [10]

Theta* was developed to overcome the problems that grid-based route planning methods such as A* do not create smooth routes because the routes are based on discrete nodes. Theta* increases the optimality of the route, considering whether there are obstacles between the nodes. When searching the neighboring nodes in Theta*, if the line connecting the parent node and the neighbor node does not meet an obstacle, the current node is not considered. However, there is a disadvantage in that route planning time increases due to the additional process of considering obstacles in A*. Nash et al. [11]

The JPS is an algorithm that significantly reduces the route planning time compared to A* by proposing a new way of searching nodes. But, it has a disadvantage that it can only be used on regular grid maps. Harabor and Grastien [12]

The sampling-based route planning method searches a node by randomly selecting a point within an area. A Rapidly-exploring Random Tree (RRT) is the basis of other sampling-based route planning methods. RRT is an algorithm that randomly searches nodes in an area within a specific radius from the current node to reach the goal node. Steven M.LaValle [13] However, RRT does not plan the optimal route but only connects randomly searched nodes.

Like RRT, RRT* randomly finds a node in an area within a specific radius from the current node, but when connecting nodes, it considers the cost of the nodes for an optimal route. Karaman and Frazzoli [14] RRT* is guaranteed to find the optimal route with an infinite number of iterations. However, since it is practically impossible to repeat infinitely, many algorithms that improve RRT* have emerged to calculate a fast and accurate route within a set number of iterations.

Informed RRT* can converge more quickly to a route by searching nodes within an

ellipse considering the start and goal nodes. Gammell et al.[15]

Batch Informed Trees (BIT*) increases the convergence speed of the route compared to Informed RRT* by initially searching nodes in a wide area and then narrowing down the area to find the optimal route. Gammell, Jonathan D., Siddhartha S. Srinivasa [16]

Metaheuristic algorithms are algorithms used to obtain an optimal solution that satisfies various objectives at the same time, and there are lots of algorithms, such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO). Dokeroglu et al. [17] Therefore, a metaheuristic algorithm is used to plan a route that simultaneously satisfies many purposes other than the shortest distance.

1.2.3. Route planning researches of the ship

Recently, studies on route planning of the ship have been conducted using various methods, and Table 1 summarizes them. Tsou and Cheng [18] proposed a route planning method optimizing the Estimated Time of Arrival (ETA) and distance of the route using the Ant Colony Algorithm by calculating the seakeeping performance according to the ocean weather data on the regular grid chart. Vettor et al.[19] optimized the route of a fishing vessel using SPEA2 and Dijkstra for distance, FOC, ETA, FOC, CO2 and NOx emissions, and risk coefficient on a regular grid chart. FOC, CO2, and NOx emissions are calculated based on the engine model, and seakeeping performance predicted with ocean weather data was reflected in the optimization process. Lee et al. [20] used NSGA-II, one of the genetic algorithms, to plan a route with optimal distance, ETA, and FOC. In addition, the engine RPM and seakeeping performance calculated with the ocean weather data were considered.

Liu et al.[21] proposed the risk coefficient of the ship to each grid of the chart. The risk

coefficient was defined by the distance to the obstacle, ocean current, and passage separation line. The risk coefficient was added to the cost function of A* when searching the nodes. It helps to plan a safer route. The final route was generated by curving the route considering the maneuverability of the ship.

Han et al.[22] created a route network by clustering AIS data of ships. In the region where the route network, such as the waterway, exists, a route was planned with the route network using Dijkstra. In the region where the route network does not exist, a route was planned with a regular grid chart using Theta*. The final route was generated by combining the two routes mentioned above.

For a safe route with a distance from obstacles, Lee et al. [23] expanded the coastline with a polygon offsetting algorithm. Lee et al. represented the chart with a quadtree and then used it to create a visibility graph chart considering the departure and destination. A route minimizing distance and ETA was planned using Dijkstra with ocean weather data on the generated visibility graph chart.

Table 1. Related works

Related Works	Year	Environment representation	Route planning Method	Objectives
Tsou and Cheng [18]	2013	Regular grid	Ant Colony Algorithm	Distance, ETA
Vettor et al.[19]	2016	Regular grid	SPEA2, Dijkstra	Distance, ETA, FOC, CO_2 and NO_x emissions, Risk coefficient
Lee et al.[20]	2018	Regular grid	NSGA-II	Distance, ETA, FOC
Liu et al.[24]	2019	Regular grid	A*	Distance, Risk coefficient
Han et al. [22]	2020	Regular grid, AIS trajectories	Dijkstra, Theta*	Distance
Lee et al. [23]	2021	Visibility graph, Quadtree	Dijkstra	Distance, ETA
This study	2023	Quadtree, Voronoi diagram	HPA*, A*	Distance

1.3. Summary of this study

This study proposes a route planning method considering the characteristics of small ships, and the configuration diagram is shown in Figure 2.

This study used electronic navigational charts created in accordance with S-57, the International Hydrographic Organization (IHO) transfer standard for digital hydrographic data. The electronic navigational chart created in accordance with the S-57 is produced in different scales depending on the purpose of the chart. All objects necessary for navigation are represented in vector format composed of latitude and longitude coordinates. Since it

is difficult to use it directly for route planning, it is necessary to convert the electronic navigational chart into a different representation suitable for route planning.

In this study, among the objects of the electronic navigational chart, the objects necessary for the route planning of the small ship were extracted. In order to plan an exact route, the same objects produced in different scales in the same area were integrated into the smallest scale. After that, the vector format objects integrated into the smallest scale were converted to a different representation suitable for route planning. All objects converted to a suitable representation for route planning are combined into the chart. A quadtree representation was used for route planning inside and outside the marina. The centerline of the marina shape created using the Voronoi diagram was used for route planning inside the marina.

The route planning method is divided into the route planning at a high-level chart and the route planning at low-level charts. A high-level chart and low-level charts are made by dividing the combined chart. When the user inputs the departure, destination, and information of the ship, route planning is performed first on the high-level chart. A proposed optimization process for the route at high-level chart improves the quality of the route by choosing the optimal node. If the node of the high-level route is outside the marina, the route planning outside the marina at the low-level chart is applied, and if it is inside the marina, the route planning process inside the marina at the low-level chart is applied. The low-level routes need to go through an additional smoothing process. The final route is generated by combining the low-level routes. The proposed method was evaluated and improved by analyzing the distance and safety of the route, route planning time, and memory usage in various sea areas.

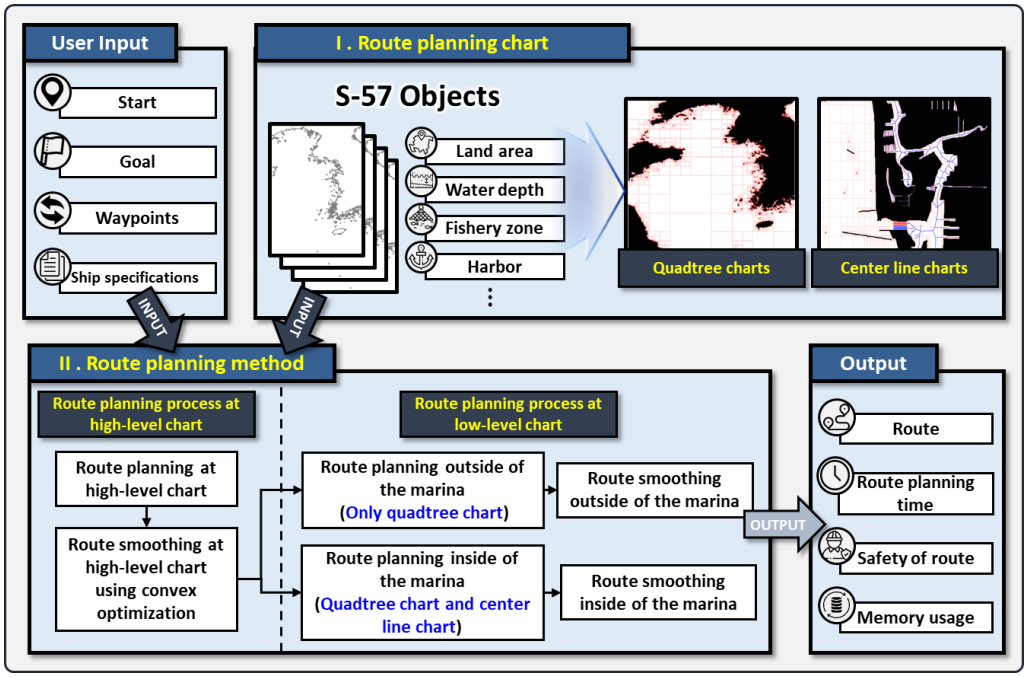


Figure 2. Configuration diagram of this study

2. Generating method for route planning charts

2.1. Integrating electronic navigational charts

The information for generating route planning charts for the small ship is from electronic navigational charts created in accordance with the S-57 standard. The S-57 standard is a rule established by the IHO. It is designed to contain information for safe

navigation. Information in the S-57 standard warns navigators of dangerous situations such as stranding or collision and delivers optimal route plans.

All information on the electronic navigational chart is expressed as an object. Objects are classified into four types: Geo, Meta, Collection, and Cartographic. Geo expresses real objects in reality and represents the location and shape of objects using latitude and longitude coordinates. Meta represents the information of the objects, such as the accuracy of data and the range of objects. The collection describes the association between two objects, and the Cartographic contains objects related to how to make charts. International Hydrographic Organization [25]

Electronic navigational charts are created with different scales depending on the navigational purpose of the electronic navigation chart. Electronic navigational charts with different scales contain different information about objects. Table 2 shows the scale range according to the use of the electronic chart. If route planning charts are created with only overview charts, the objects that require to be elaborated, such as anchorages, are inadequate for route planning for a small ship. All regions should be described on the chart at the smallest scale in existence. Thus, the process of integrating electronic navigational charts in the different scale range is required.

Table 2. Scale range according to navigational purpose

Usage Band	Navigational Purpose	Scale Range
1	Overview	1500000 ≤ 0
2	General	350,000 ~ 1,499,999
3	Coastal	90,000 ~ 349,999
4	Approach	30,000 ~ 89,999
5	Harbor	7,500 ~ 29,999
6	Berthing	< 7,500

Figure 3 is the example of the electronic navigational charts with different usage. The navigational purpose of the chart in red box is Approach, and the navigational purpose of the chart in blue box is Coastal.

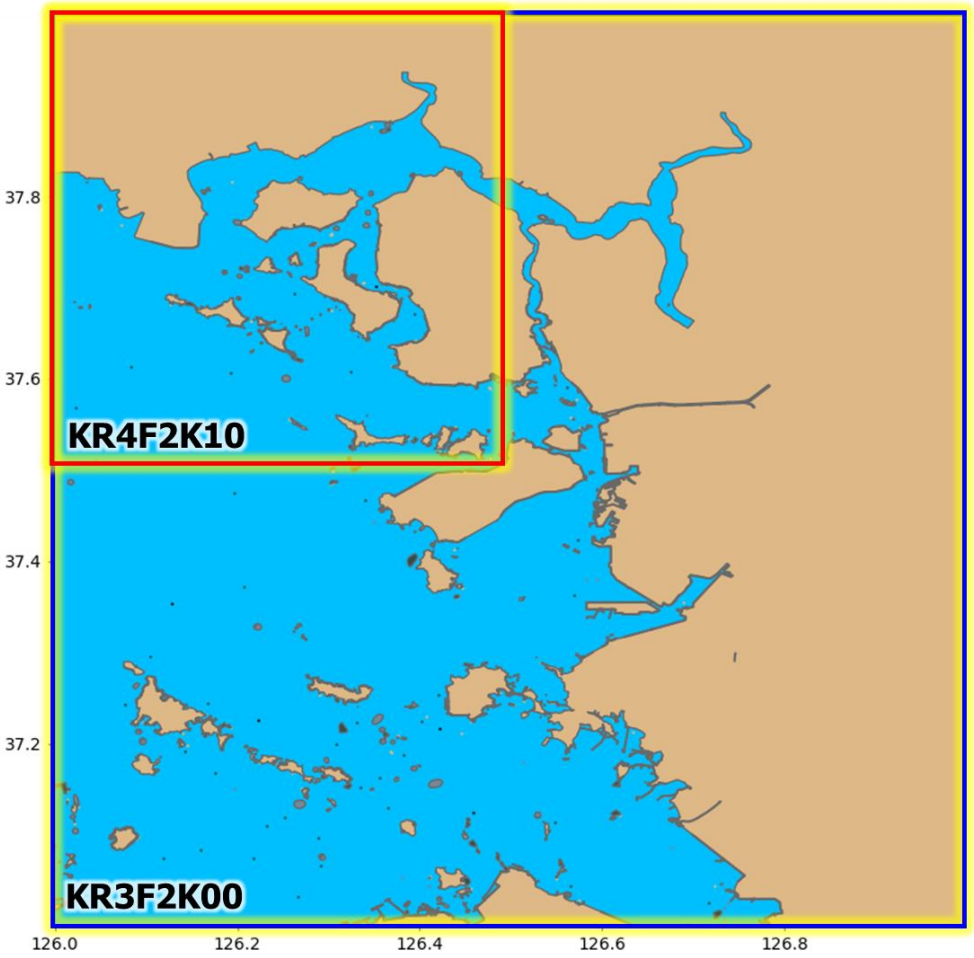


Figure 3. Example of the electronic navigational charts

The process of integrating electronic navigational charts produced in different scales is shown in Figure 4.

The classification of charts with different scales is needed before integrating electronic navigational charts. Electronic navigational charts representing overlapping areas should be in the same category. An object named Coverage in S-57 is used to define the area. It

indicates the range of topographical information of the objects. Generally, electronic navigational charts with a large scale have broad coverage compared to electronic navigational charts with a small scale, so charts with a larger scale contain the charts with a smaller scale in the same area.

A set called combined_chart is for storing integrated charts in the same category. The chart with the largest scale in a category is designated as the current_charts. The parts of the objects in the current_charts overlapping with all the Coverages of the charts with a scale lower than the current_charts are subtracted. The remains of the current_charts are added to the combined_chart. Then, the charts with the next smaller scale are assigned to the current_charts. If the current_charts is the smallest scale in the current category, the chart is just added to combined_chart. Then, the integrating process of the current category ends, and the integrating process of the next category starts. If the current_charts is not the smallest scale in the current category, the subtraction process is repeated.

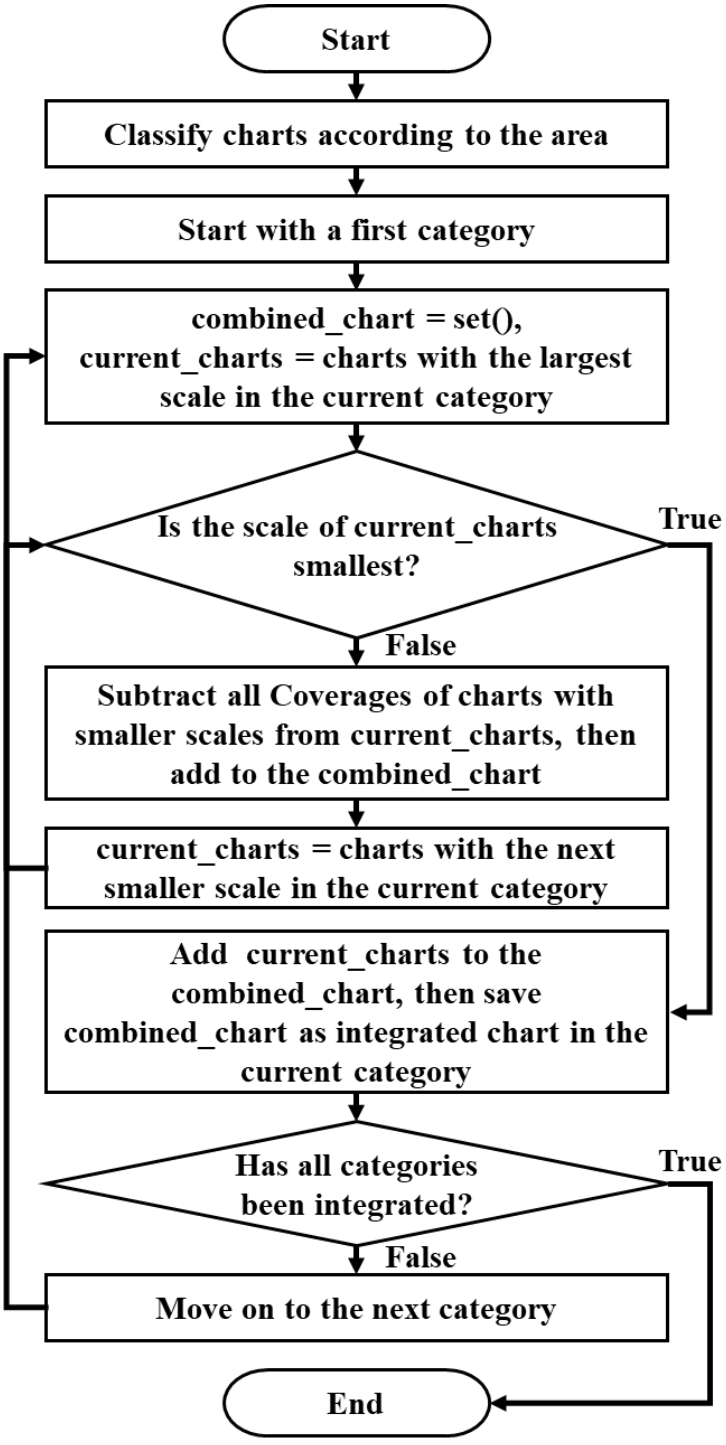
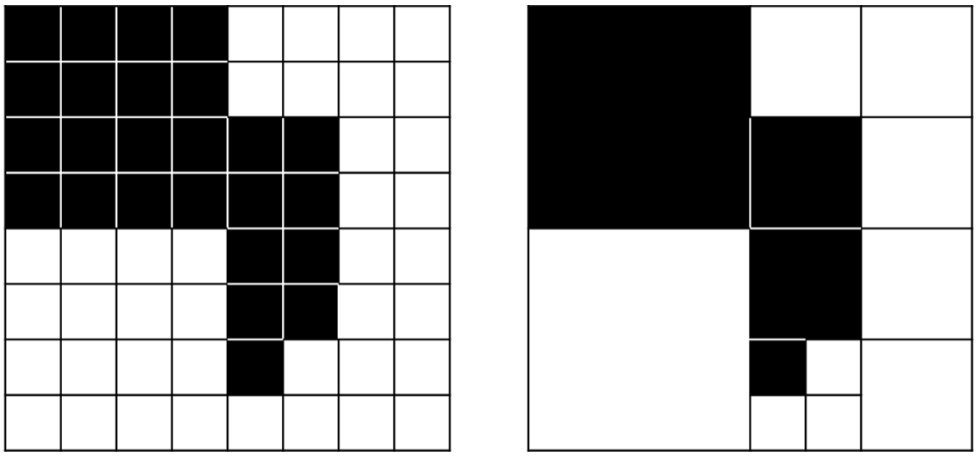


Figure 4. Flow chart of integrating electronic navigational charts in different scales

2.2. Quadtree charts

2.2.1. Quadtree

The integrated chart expressed in latitude and longitude coordinates needs to be converted to the quadtree representation for route planning. As explained in 1.2, the quadtree is a tree structure with only four child nodes. The quadtree representation for route planning can express complex topography more efficiently than the regular grid representation. Since quadtree represents the charts with a non-uniform resolution, charts can be created with a drastically smaller number of total nodes than regular grid representation. Figure 5 (a) is an image represented in a regular grid. Figure 5 (b) is an image represented in a quadtree. It can be seen that an image represented in a quadtree has much smaller nodes than an image represented in a regular grid, even though the image has the same resolution.



(a) Image representation with regular grid

(b) Image representation with quadtree

Figure 5 Comparison between regular grid representation and quadtree representation

The properties of the quadtree node used in this study are as below.

- Corner: bottom left coordinate of node
- Width: width of node
- Height: height of node
- Depth: number of the decomposition of this node
- n: identification number of the node
- Children: child nodes of node
- Water_depth: water depth of node
- MAX_DEPTH: Maximum number of the decomposition
- Info: 0 or 1 or 2 (0: blocked node, 1: unblocked node, 2: node with children)
- OBJLs: identification number of the object which the node represents

The location of a quadtree node is expressed by the Corner property, which is the latitude and longitude coordinates of the lower left corner of the quadtree node. And the size of a quadtree node is expressed by the Width and Height property which is the width and height of the quadtree node.

The Depth property of a quadtree node means the number of the decomposition until the current quadtree node is created. The higher the number of the decomposition, the smaller the Width and Height of the quadtree node. MAX_DEPTH property of the quadtree node means the maximum number of the decomposition of the current chart. MAX_DEPTH property determines the resolution of the quadtree chart.

The n property is a unique identification number of a quadtree node, and quadtrees in a quadtree chart have different n. A child quadtree node is one of the quadtree nodes created by dividing a parent quadtree node. A parent quadtree node stores n of child quadtree nodes in the Children property. The relationship between the child quadtree node and the parent quadtree node is shown in the Figure 6.

The Depth of a child quadtree node is the value of adding 1 to the depth of the parent quadtree node. And the n of a child node is the n of a parent node multiplied by four plus k. The direction of the child node determines the value k. If the quadtree node is at SW, the value of k is one. If the quadtree node is at SE, the value of k is two. If the quadtree node is at NW, the value of k is three. If the quadtree node is at NE, the value of k is four.

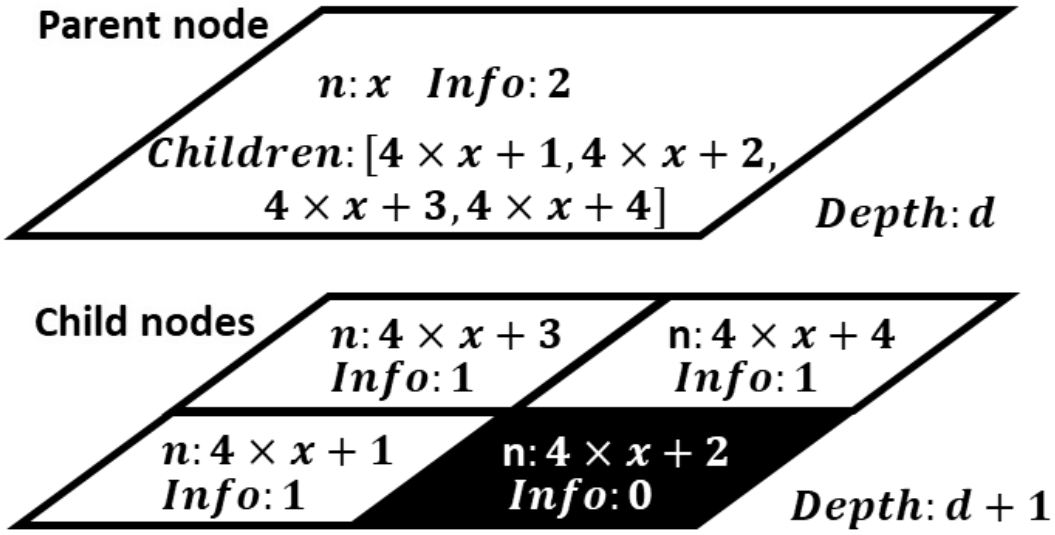


Figure 6. Relationship between the child quadtree nodes and the parent quadtree node

A quadtree node without a child quadtree node is called a quadtree leaf node. The Info property of a quadtree node has values of 0, 1, and 2. 0 means an unpassable quadtree node among the quadtree leaf nodes, 1 means a passable quadtree node among quadtree leaf nodes, and 2 means a quadtree node with child quadtree nodes.

The Water_depth property of a quadtree node is the water depth in the area represented by a quadtree node. The OBJLs property of a quadtree node contains the type of object represented by a quadtree node. Figure 7 shows the hierarchy of the quadtree nodes to represent charts in this study.

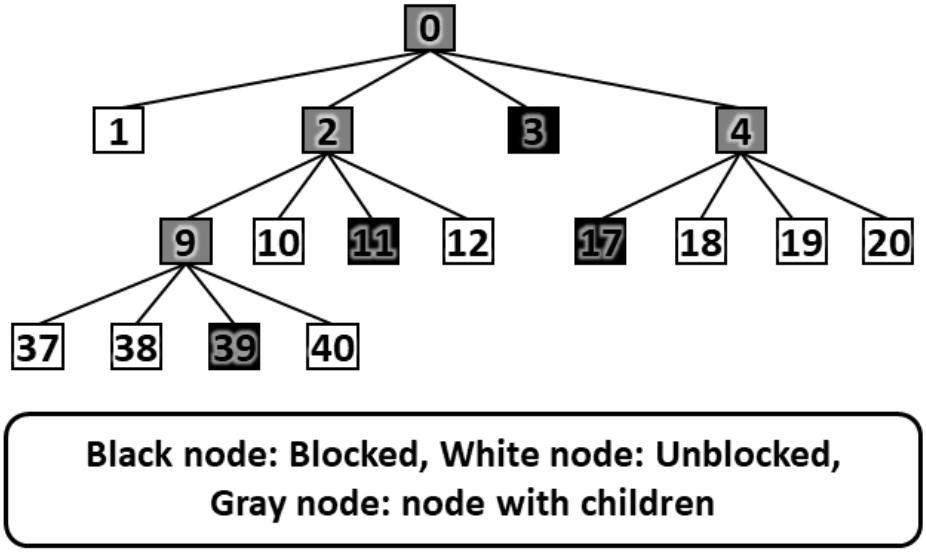


Figure 7. Hierarchy of the quadtree nodes

2.2.2. Generating method of quadtree charts

Prior to the generation process of the quadtree chart, there are two basic ways to generate a quadtree: Point Region (PR) method and Polygon Map (PM) method. Samet and Webber [26]

In the PR method, if there are more than c points in a quadtree node, the quadtree node is divided to create child quadtree nodes. If there are fewer than c points in a quadtree node, quadtree nodes with points contain information about the points, and quadtree nodes without points are saved as empty quadtree nodes.

The PM method is a method in which line segments are additionally considered in the PR method. In addition, if the PR method divides until the number of internal points is less than c, the PM method divides up to the pre-specified maximum number of divisions. The

PM method includes the PM3 method, the PM2 method, and the PM1 method according to the degree of consideration of line segments.

The PM3 method considers only points like the PR method, and the PM2 method splits a quadtree node when a quadtree node includes points or lines that do not meet.

Finally, the PM1 method is the strictest method in which the PM1 method splits a quadtree node when a point or a line segment exists within a quadtree node. In this study, the PM1 method was applied to generate the quadtree chart to represent sophisticatedly. [27]

In this study, the process of generating a quadtree chart by applying the PM1 method is shown in Figure 8. The first step is the first decomposition which is performed using the PR method based on the latitude and longitude coordinates of the object in the integrated chart. The maximum number of the decomposition depends on the minimum quadtree node size required for each object. As shown in Figure 8 (a), The result of the first decomposition does not consider line segments, so a process of interpolating between points is required as second step of generating a quadtree chart process.

If interpolation with equal intervals is applied between two points for considering line segments, the interpolated points sometimes miss the quadtree nodes which intersect the line segments, as shown in Figure 9 (a). Therefore, the interpolation method considering virtual quadtree leaf nodes, which meet the line segment and have a MAX_DEPTH as Depth, is implemented.

As shown in Figure 9 (b), blue points, which are the center points of the virtual quadtree leaf nodes, are the result of the interpolation method of this study. The third step is the

second decomposition. It divides the quadtree nodes considering interpolated points. The result of the second decomposition is boundaries of obstacles represented by a quadtree.

Finally, the quadtree chart was completed by adding object information, such as Info or OBJLs, to the quadtree nodes inside the obstacles.

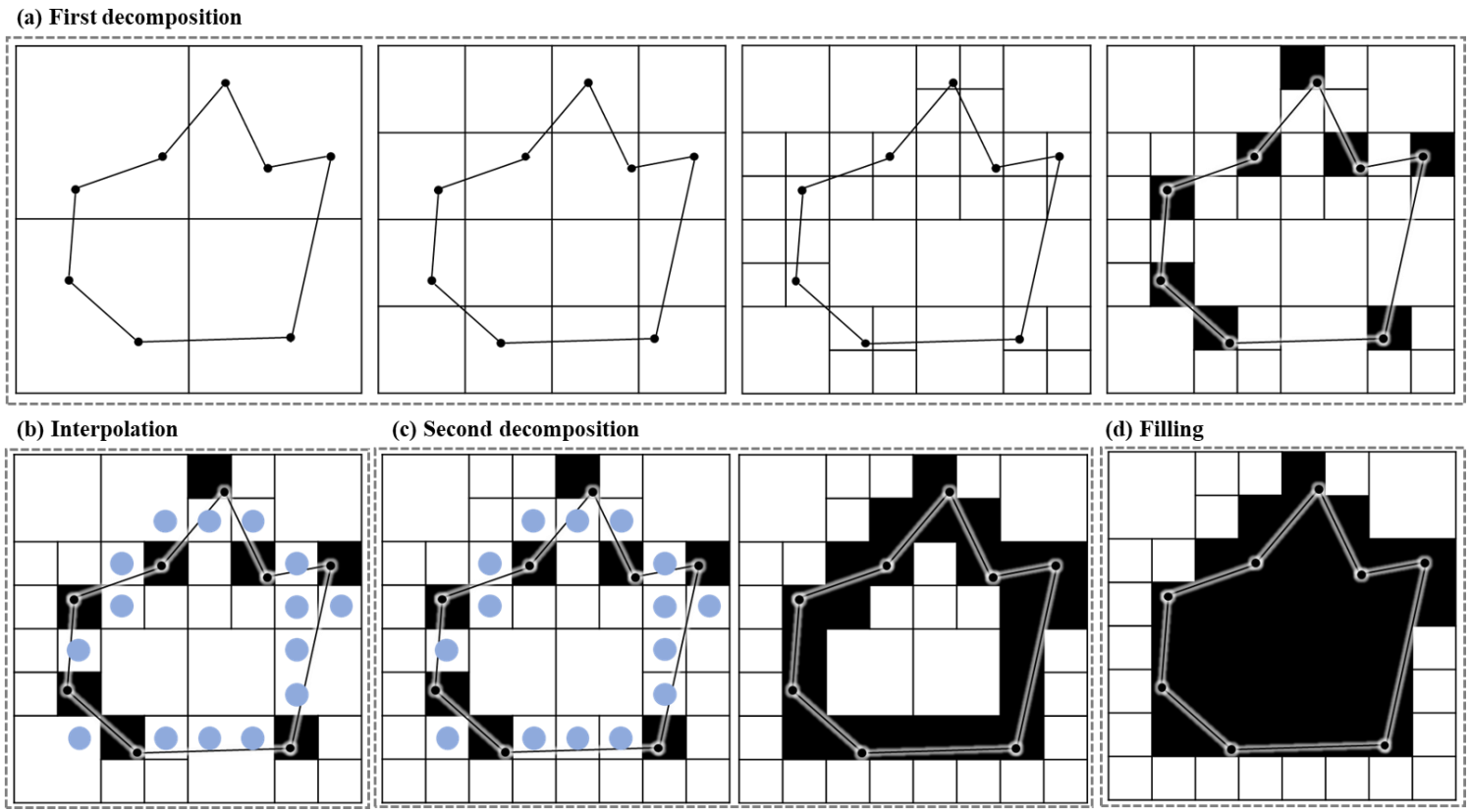
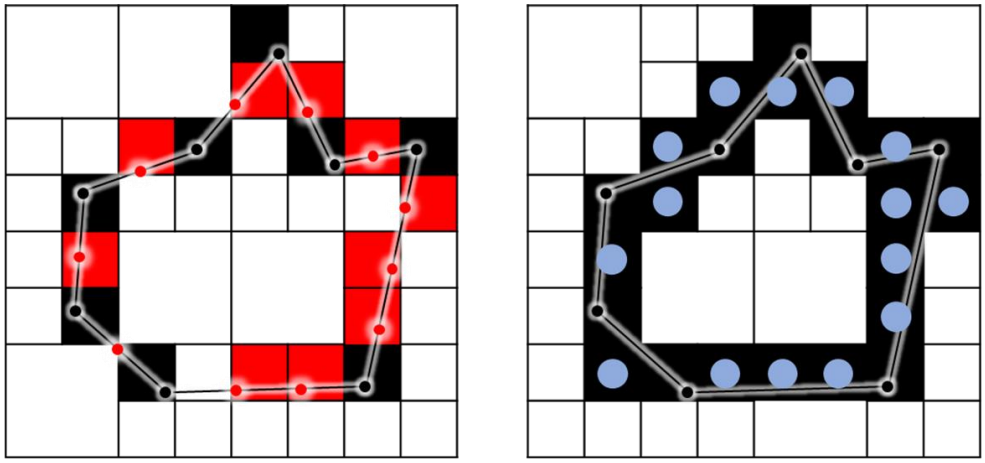


Figure 8. Generating process of the quadtree chart



(a) Interpolation using equally spaced points

(b) Interpolation considering virtual nodes

Figure 9. Comparison of interpolation methods for generating the quadtree chart

Objects of the electronic navigational charts have different resolutions depending on the characteristics of the objects, and specific objects are required to be expressed with very high accuracy. However, there is a problem in that the total nodes of the chart become excessively large if all objects are depicted like the objects that require the highest accuracy.

In order to solve this problem, this study used a method of integrating after creating quadtree charts with different minimum quadtree node sizes for each object. The different quadtree objects in the same area have the same Height and Width ratio and different MAX_DEPTH.

Integrating quadtree objects proceeds through integrating the two quadtree objects in Figure 10 for each object. The order of the integrating quadtree objects proceeds in the order of the largest minimum node size of the quadtree objects. The base_world in Figure

10 means the quadtree objects with the largest minimum node size in the first integration process and the integrated objects after the first integration process. The world in Figure 10 is an object added to the base_world, and the combined_world is a combined object which is copied from the result of integrating the base_world and the world.

The world[x] is a quadtree node in the world having x as n. The combined_world[x] and the base_world[x] tell the same thing as the world[x]. A quadtree with the same x on the different quadtree in the same area has the same Corner, Width, Height, Depth, and n, but it has different Info, Children, and OBJLs.

Therefore, if there are quadtree nodes with the same x in the world and the base_world, it is required to change the Info, children, and OBJLs of the combined_world for integrating two objects. If at least one of the Info of the world[x] and the base_world[x] is 0, the Info of the combined_world[x] is set to 0, and the OBJLs of the world[x] is added to the OBJLs of combined_world[x]. If the Info of world[x] is 2 and Info of base_world[x] is 1, combined world[x] is replaced with world[x], and if the Info of world[x] is 2 and the Info of base_world[x] is 0, combined_world[x] and base_world[x] is equal.

If x does not exist in the base_world, world[x] is under the quadtree leaf nodes of the base_world. The n of the quadtree leaf node in base_world is named as parent x. If one of the Info of the world[x] and base_world[parent x] is 0, the Info of combined_world[x] is 0, and OBJLs of world[x] is added to the OBJLs of the combined_world[x] .

Finally, after integrating all objects, the quadtree chart is completed by adding water depth to the integrated quadtree chart.

The summary of the process of generating a quadtree chart is as follows. First, the first decomposition is performed for each object based on the latitude and longitude coordinates of the objects of the electronic navigational chart. Second, the coordinates of the object are interpolated by considering the virtual quadtree leaf node, which is the quadtree node with the minimum size. Third, the second decomposition using interpolated points is performed. Fourth, the quadtree chart of an object is filled with the object information. Finally, quadtree charts for route planning are completed by integrating all quadtree objects and adding water depth information.

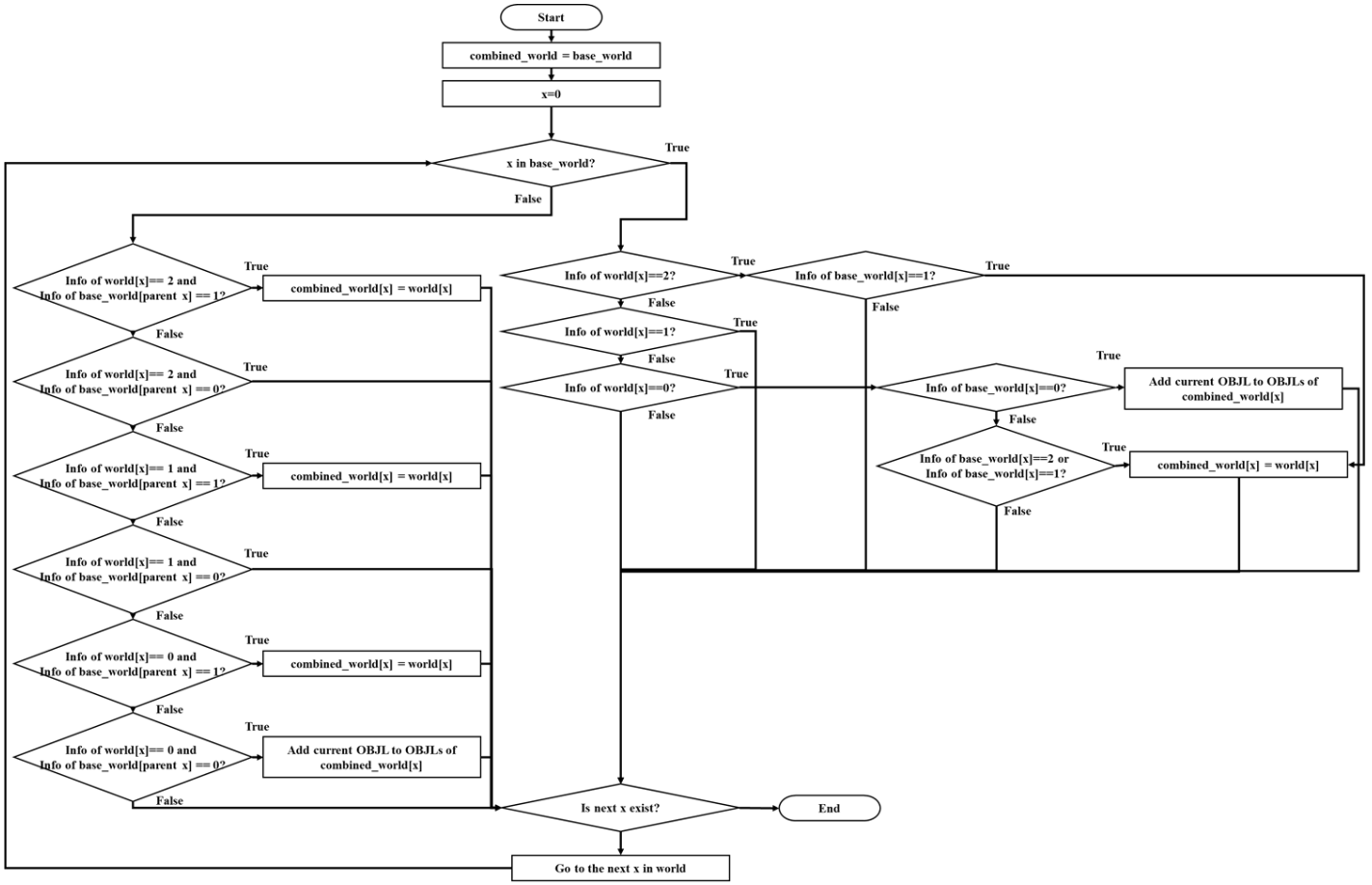


Figure 10. Flow chart of integrating quadtree objects

2.3. Centerline charts

A marina is where ships are anchored and moored. Mostly, the shape of the marina is complex, and the density of ships is high at the marina. Therefore, the route for the small ship needs to have a certain distance from obstacles for a safe voyage.

For that reason, the centerline of the marina was adopted as a route planning chart inside the marina. The shape of the marina was extracted with latitude and longitude coordinates. The passage separation lines were added to the shape of the marina to follow the port-to-port rule at the entrances and exits of the marina. The two centerlines in the entrances and exits of the marina generated by the passage separation lines are directed graphs to follow the port-to-port rule.

2.3.1. Generating method of centerline charts

The centerline of the marina was generated using the Voronoi diagram. The site points are the points consisting of the marina shape. The Voronoi diagram is a method of dividing a region based on site points, and the region of a site point is defined as a set of points closer than all other site points. Fortune [28]

Figure 11 is an example of a Voronoi diagram. The boundaries of the Voronoi diagram are called Voronoi edges. Each Voronoi edge is created as a perpendicular bisector of two site points. A Voronoi region is a region surrounded by Voronoi edges and has only one site point inside the Voronoi region. Also, Voronoi vertices are the result of the vertices formed by intersecting Voronoi edges. Due to the characteristic of the Voronoi edge that is

the same distance from the two site points, the centerline charts created by the Voronoi diagram is composed of Voronoi edges.

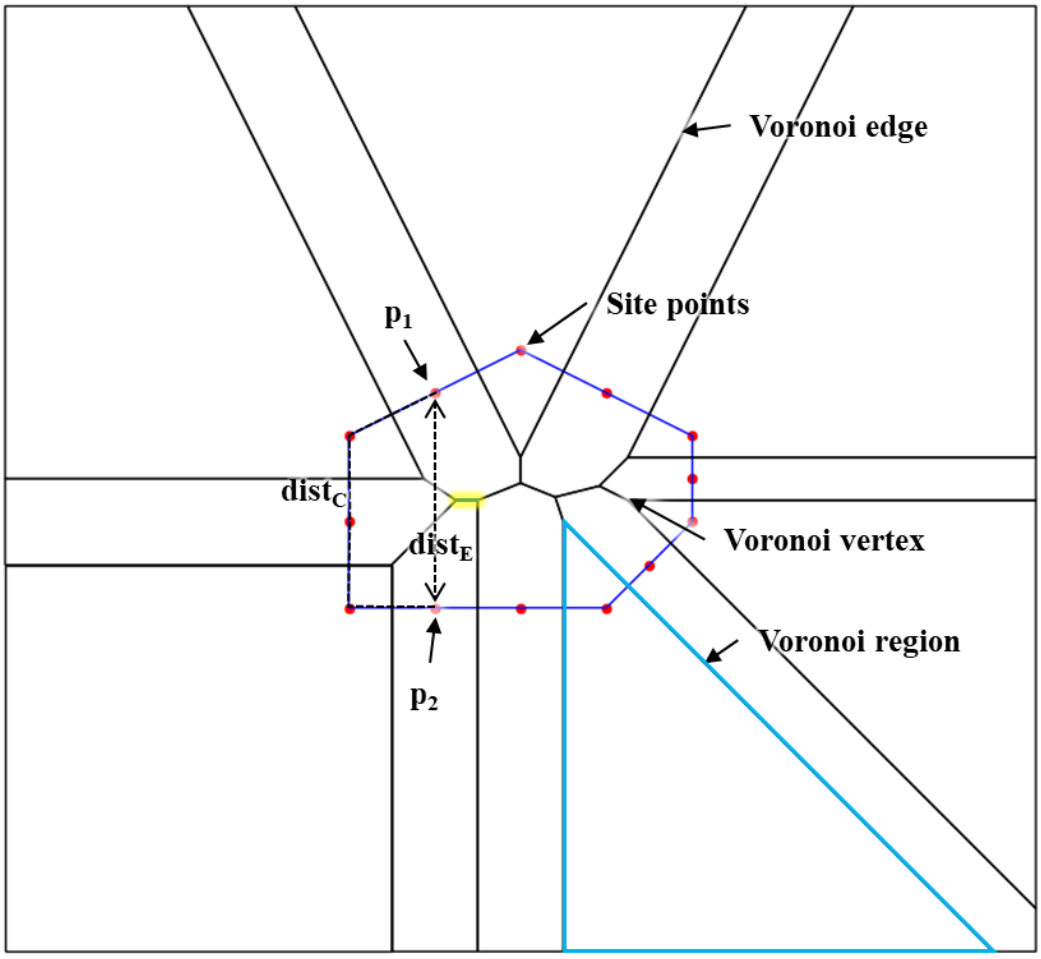


Figure 11. Example of Voronoi diagram

The centerline of the marina shape can be obtained by removing some Voronoi edges of the Voronoi diagram generated based on the site points of the marina shape. Younas and

Figley [29] Since all Voronoi edges are part of the perpendicular bisector of two site points, one Voronoi edge corresponds to two site points.

The two site points p_1 and p_2 correspond to the green highlighted Voronoi edge in Figure 11. The $dist_E$ between two site points is defined as the Euclidean distance between the two site points. The $dist_C$ is the distance of the contour of the shape from p_1 to p_2. Two types of distances are calculated depending on the direction, clockwise and counterclockwise.

The shorter distance between the clockwise and counterclockwise distances is defined as the $dist_C$. D_C refers to the half of the circumference of the circle with $dist_E$ as the diameter. If $dist_C$ of the two site points is less than D_C, the current Voronoi edge is removed, and if $dist_C$ of two site points is longer than D_C, the current Voronoi edge remains the centerline. The highlighted green Voronoi edge in Figure 11 is evaluated as the centerline because $dist_C$ is longer than D_C. The above process was applied to Voronoi diagram of the shape of the marina, and the Voronoi edges that remained were determined as the centerline of the marina.

3. Route planning method

3.1. Route planning at high-level chart

The graph-based method was chosen among the graph-based method, sampling-based method, and method using a metaheuristic algorithm. The sampling-based method creates a route fast, even for a long route, but the quality of the route varies even with the same sampling number. In addition, the method using the metaheuristic algorithm has the disadvantage of taking a long time to plan a route.

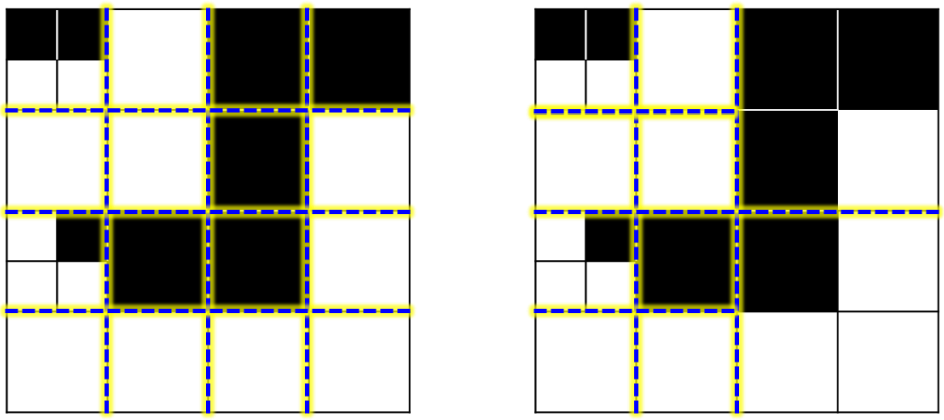
In this study, HPA* was used among the graph-based methods. Computers used in small ships are mostly not a high specification, and when used by individual users, long route planning time gives users an unpleasant experience. Therefore, HPA*, which can quickly plan a route using a small amount of memory even in complex coastal areas and which can be used with the quadtree chart, is suitable for route planning of a small ship.

Figure 13, Figure 14, Figure 15, Figure 16, Figure 17, Figure 18 are figures to explain the route planning method applied in this study. For clear understanding, it describes the route planning on the regular grid representation, not the quadtree representation.

As HPA* plans the route hierarchically, a pre-processing of creating a hierarchical chart using existing quadtree charts is required. As HPA* plans the route hierarchically, a pre-processing of creating a hierarchical chart using existing quadtree charts is required. First, the quadtree charts are divided. There are two ways to divide the quadtree charts. One is to divide the chart based on latitude and longitude coordinates, as shown in Figure 12 (a). Due to the nature of the quadtree, the number of quadtree nodes within a chart varies

depending on the complexity of the chart, even if it is divided into the same latitude and longitude coordinates. If a route is created on a quadtree chart with many quadtree nodes, it takes a long time to load the quadtree chart and plan the route.

Thus, in this study, the chart was divided based on the number of quadtree nodes. The method of dividing based on the number of quadtree nodes is as follows. If the number of quadtree nodes existing inside the area exceeds ten thousand, it is divided into four quadtree charts. Each divided chart corresponds with four child nodes of the node with the lowest depth in the current chart. If the number of quadtree nodes existing inside the area is less than ten thousand, the area is defined as a low-level chart, and the above process is repeated until the number of quadtree nodes of all quadtree nodes becomes less than ten thousand. Figure 12 (b) shows the chart divided based on the number of the quadtree nodes. It is divided when the number of quadtree nodes existing inside the area is less than four. The number of quadtree nodes in a low-level chart is distributed evenly.



(a) Chart divided based on latitude and longitude coordinates

(b) Chart divided based on the number of the quadtree nodes

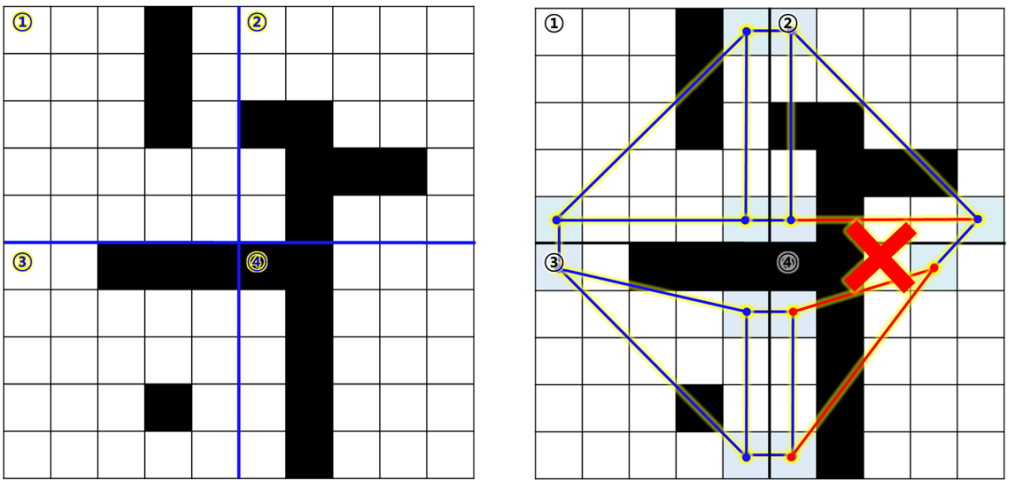
Figure 12. Two methods of the dividing the quadtree charts.

Unlike the quadtree charts, the centerline chart created based on the Voronoi diagram inside the marina has a small file size because most of the marina is small. Therefore, an additional dividing process is not required for the centerline chart. Hence, the centerline chart is one low-level chart.

The high-level chart is a graph with the entrance as the vertex. The entrance of the high-level chart is a passage for the low-level charts. Some of the nodes on the side where the lower-level charts meet each other are designated as the entrance. Generally, the entrance is composed of nodes in the middle or at both ends of the edge.

If the ship can sail from one entrance to another, the edge between entrances inside a low-level chart is formed. In this study, the route planning between two entrances using Dijkstra is implemented to check whether the ship can pass between entrances. If the route is planned successfully, the edge between entrances is formed. Figure 13 (b) is the example

of a high-level chart.

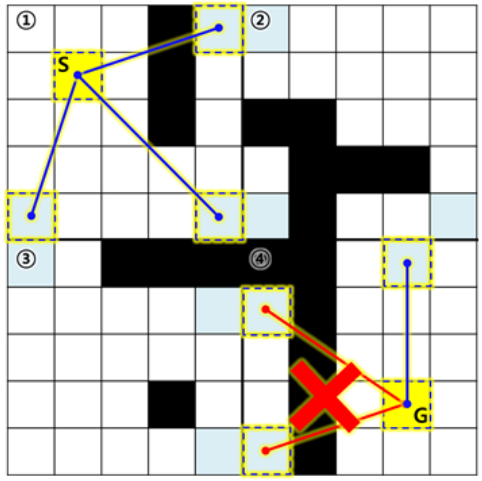


(a) Charts divided into 4 parts **(b) The graph of entrances (high-level chart)**

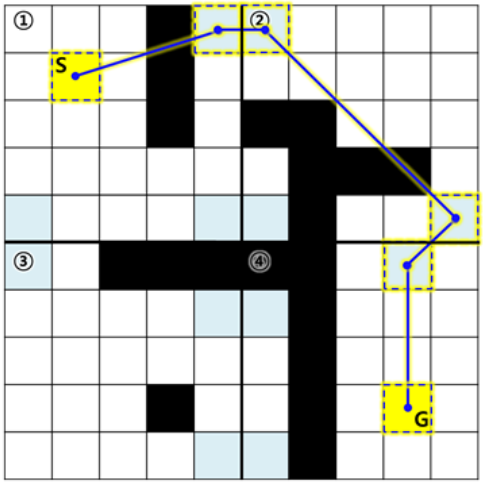
Figure 13. Example of chart preprocessing for HPA*

Figure 14 shows the route planning method that proceeds after the high-level and low-level charts are created. Black nodes are obstacle nodes, white nodes are passable nodes, and light blue nodes are nodes selected as entrances. As shown in Figure 14 (a), select the chart that includes the selected node (e.g., the departure node, the arrival node). As shown in Figure 15, whether the selected node and the entrance are passable is determined by whether they are in the same polygonized navigable region. The high-level chart is changed by forming the edge between the entrance inside the selected chart and the selected node. After that, as shown in Figure 14 (b), the route is planned by A* on the high-level chart. It determines what entrances to pass in the low-level chart. Finally, as shown in Figure 14 (c),

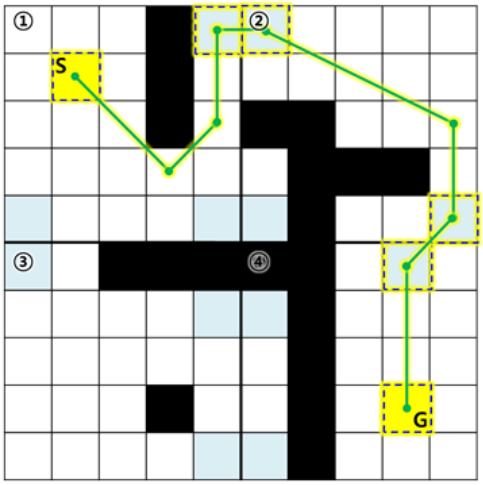
the route at the low-level chart is planned for all pairs of entrances.



(a) Connecting start and goal node with entrances



(b) Route planning using A* at high level chart



(c) Route planning using A* at low level chart

Figure 14. Process of HPA*

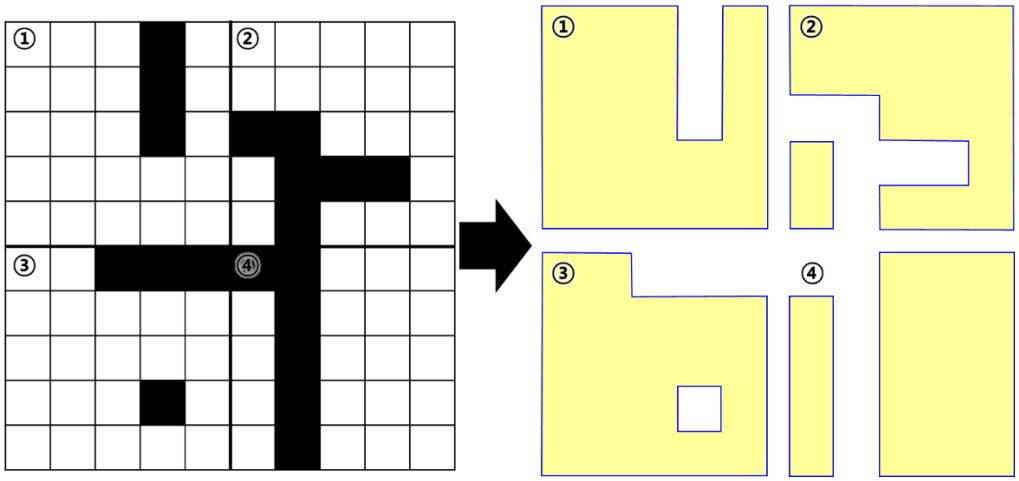


Figure 15. Polygonization of navigable regions

However, if the nodes at a specific location are designated as the entrances like the original HPA* mentioned above, some nodes cannot be chosen for an optimal route, as shown in Figure 16. To resolve this problem, we define entrance differently and add a smoothing process to plan a better-quality route.

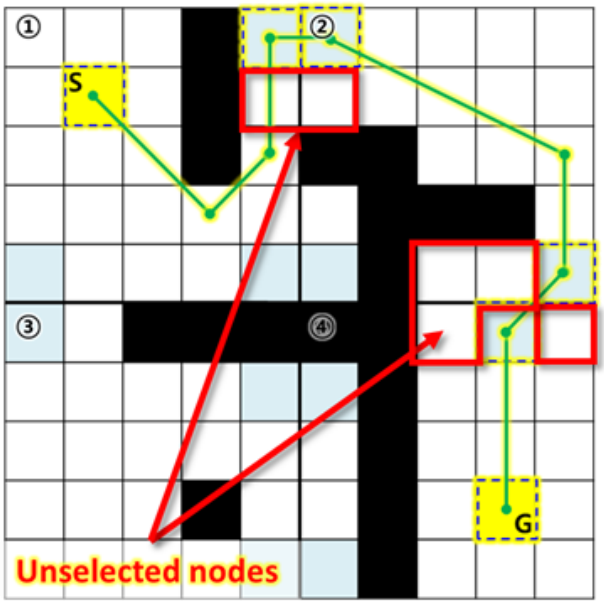


Figure 16. Example of the sub-optimal routes because of the unselected nodes

In this study, like in Figure 18 (a), a bunch of nodes in contact with other charts is defined as an entrance rather than a specific node. Then, as shown in Figure 18 (b), the route is planned at the high-level chart using the A*. The heuristic function of A* is selected as the Euclidean distance between the midpoints of the entrances. The cost function uses the distance of the route planned in the creating a high-level chart through A* on the quadtree chart when the midpoint of the two entrances is set as the departure destination. Selecting one node for planning the shortest route inside the entrance is necessary. The following optimization process is for choosing an optimal node.

The optimization problem is formulated to minimize the sum of the distances of the edges composing the route at the high-level chart. Constraints were added to consider the shape of the obstacle. The obstacles in a chart are divided based on the line connecting the

midpoints of the pair of entrances. After creating the convex hull of the two divided areas, the smaller one is used for optimization. Assuming that the vertices of the selected convex hull are the fixed points, optimization is performed in which the sum of the lengths of the lines connecting fixed points and the route points in the entrance area is minimized. Figure 17 illustrates the example of selecting the constraint points.

The first chart in Figure 17 shows a line segment $\overline{X_1X_4}$ connecting the departure and midpoint of the entrance. The line $\overline{X_1X_4}$ divides the convex hull of the obstacles and points X_1, X_4, and it creates two convex hulls S_1, S_2. Since S_1 is smaller, points X_2, X_3 are selected as the fixed points.

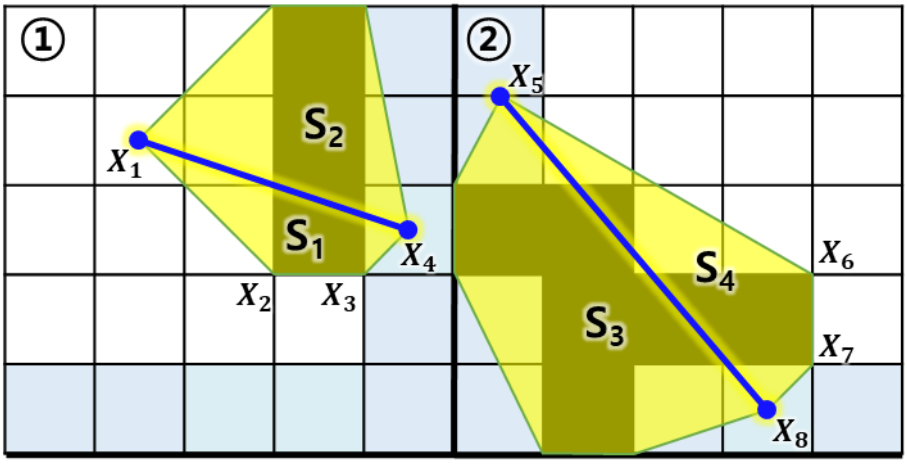


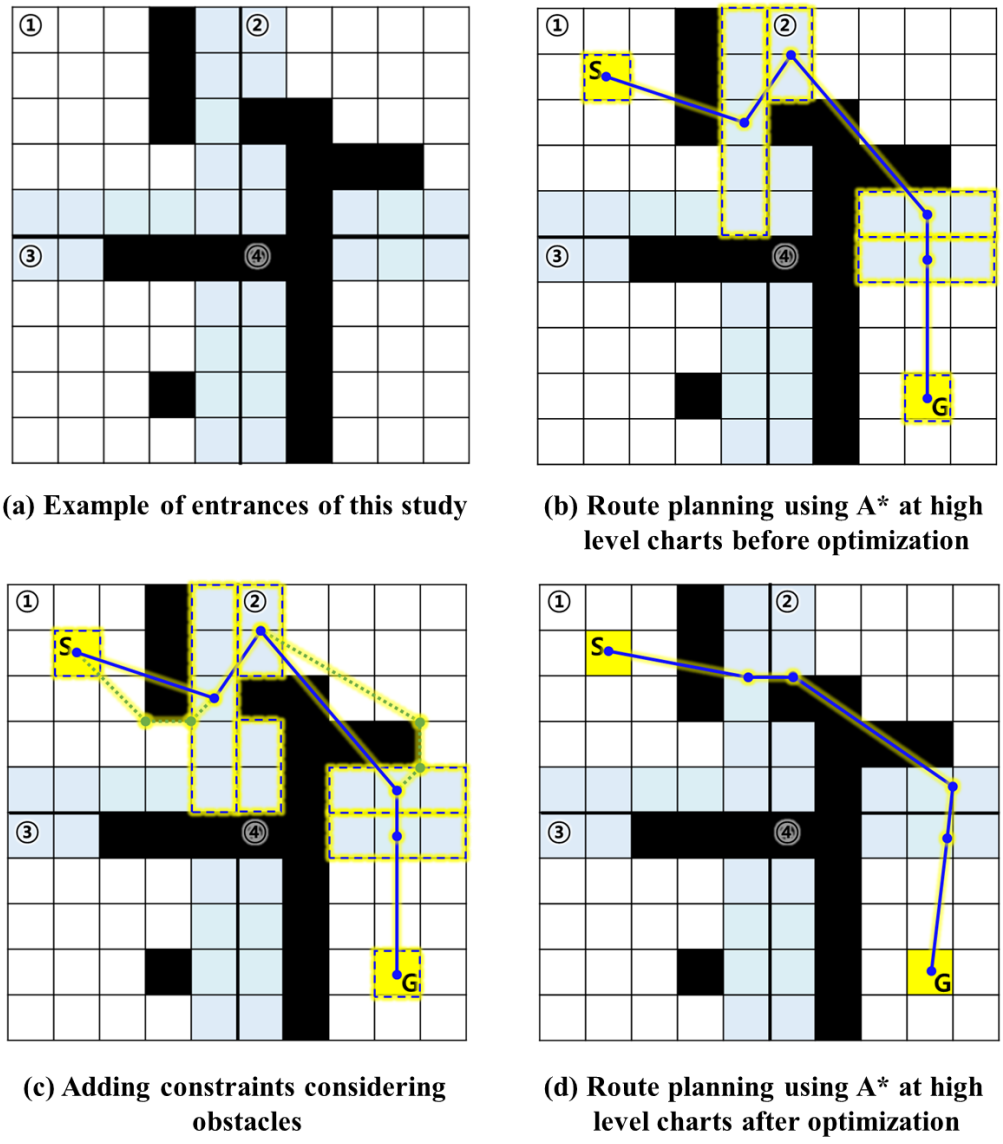
Figure 17. Selecting the constraint points

The design variables, objective function, and constraints of the optimization process are as follows.

- Design variables: $\mathrm{X_i} = (x_i, y_i)$
- Objective function: $minimize\left(\sum_{i=1}^{n-1}\|X_i - X_{i+1}\|_2\right)$
- Constraints: $\mathrm{a_i} \leq x_i \leq b_i, c_i \leq y_i \leq d_i$

$\mathrm{X_i}$ is the latitude and longitude coordinates of a route point on the entrance or the coordinates of a fixed point. The n is the sum of the number of entrances and the number of fixed points. If $\mathrm{X_i}$ is the route point in the entrance, $\mathrm{a_i}, b_i, c_i, d_i$ of the constraints are the minimum and maximum coordinates of the longitude and latitude of the entrance. If $\mathrm{X_i}$ is a fixed point, $\mathrm{a_i}, b_i$ are the longitude of a fixed point and c_i, d_i are the latitude of the fixed point. The objective function is to minimize the sum of distances between $\mathrm{X_i}$.

It is evident that a $\|\cdot\|_2$ is convex by Minkowski inequality, and since sum of the convex functions is also convex, the formulated problem is a convex optimization problem. We used CVXPY, a Python library, to solve the convex optimization problem. Agrawal et al. [30] The green points in Figure 18 (c) are the fixed points and blue points are the route points which need to be optimized. Figure 18 (d) shows the route at the high-level chart when the proposed convex optimization is applied.



(a) Example of entrances of this study

(b) Route planning using A* at high level charts before optimization

(c) Adding constraints considering obstacles

(d) Route planning using A* at high level charts after optimization

Figure 18. Routes at high-level chart using entrances composed of the bunch of the nodes

3.2. Route planning outside the marina at low-level chart

The route planning at a low-level chart uses different route planning methods if the route point at the high-level chart is outside and inside the marina. Route planning outside the marina is using A* on the quadtree chart.

When latitude and longitude coordinates are input as a departure and a destination, it is necessary to search for quadtree leaf nodes that include the input coordinates. When looking for a quadtree node containing a specific point, the search starts from the top node with n and depth as 0. The start and goal quadtree nodes are decided by increasing the Depth and choosing the child quadtree node containing the current point until the Depth is MAX_DEPTH or the current quadtree node reaches the quadtree leaf node.

When planning a route using A* on the quadtree chart, the neighboring quadtree nodes of a current quadtree node are searched. Generally, the neighboring nodes are nodes of the four cardinal or the eight cardinal. However, unlike the regular grid, a quadtree node does not always have eight nodes near one node. In this study, neighboring quadtree nodes are defined as all nodes near one quadtree node, and all neighboring quadtree nodes are used for planning an optimal route.

The method for searching the neighboring quadtree nodes is as follows. To search the neighboring quadtree nodes, quadtree nodes of the same Depth with a current quadtree node should be found. If a neighboring quadtree node does not have the same Depth, a virtual quadtree leaf node with the same Depth is searched before the real quadtree leaf

node is searched, as shown in Figure 19.

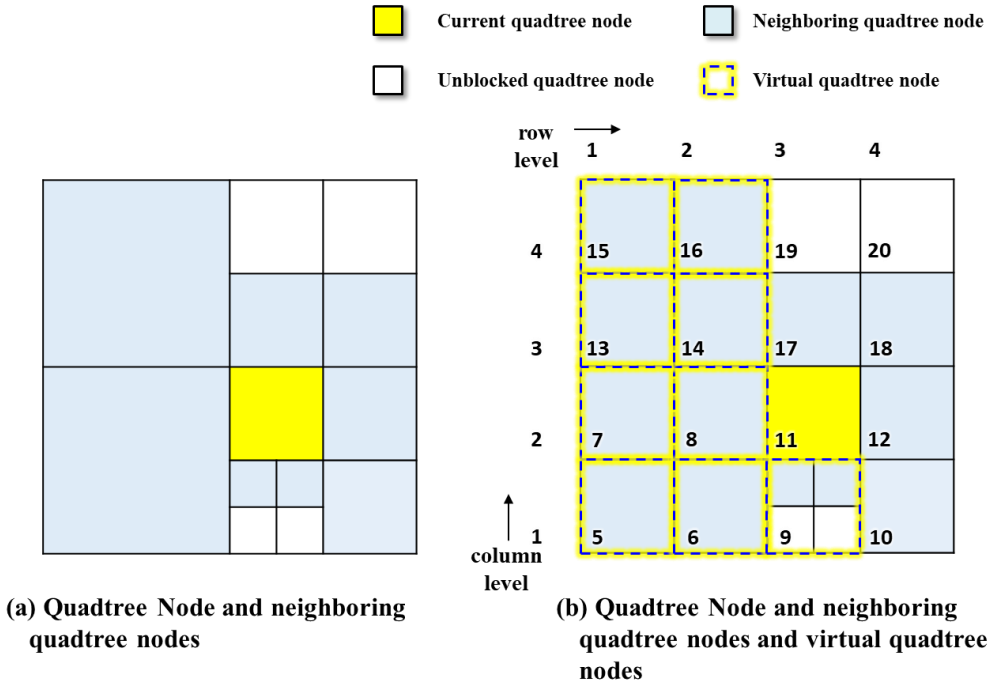


Figure 19. Example of searching the neighboring quadtree nodes

Searching the neighboring quadtree nodes in the four cardinal directions (N, W, S, E), frequently used in the regular grids, is executed first. As described in 2.2.1, there are quadtree nodes with different n on one quadtree chart. Therefore, searching for the neighboring quadtree node is equivalent to searching the n of the neighboring quadtree node. The difference of n between two quadtree nodes at the same Depth and location within a quadtree chart is fixed. Thus, the n of the quadtree node in a specific direction is calculated by adding the difference between two quadtree nodes.

Figure 20 is a flowchart for calculating the difference of n between the current quadtree node and the quadtree node moved in a certain direction. All quadtree nodes inside the quadtree chart have positive n. If the quadtree node moved in a certain direction is outside the current chart, the difference of n between the two nodes is negative infinity. The function Get_direction_list is used to find out the column or row level of a current quadtree node represented by n. If n is positive, The Get_direction_list saves the directions of all parent quadtree nodes from the top quadtree node.

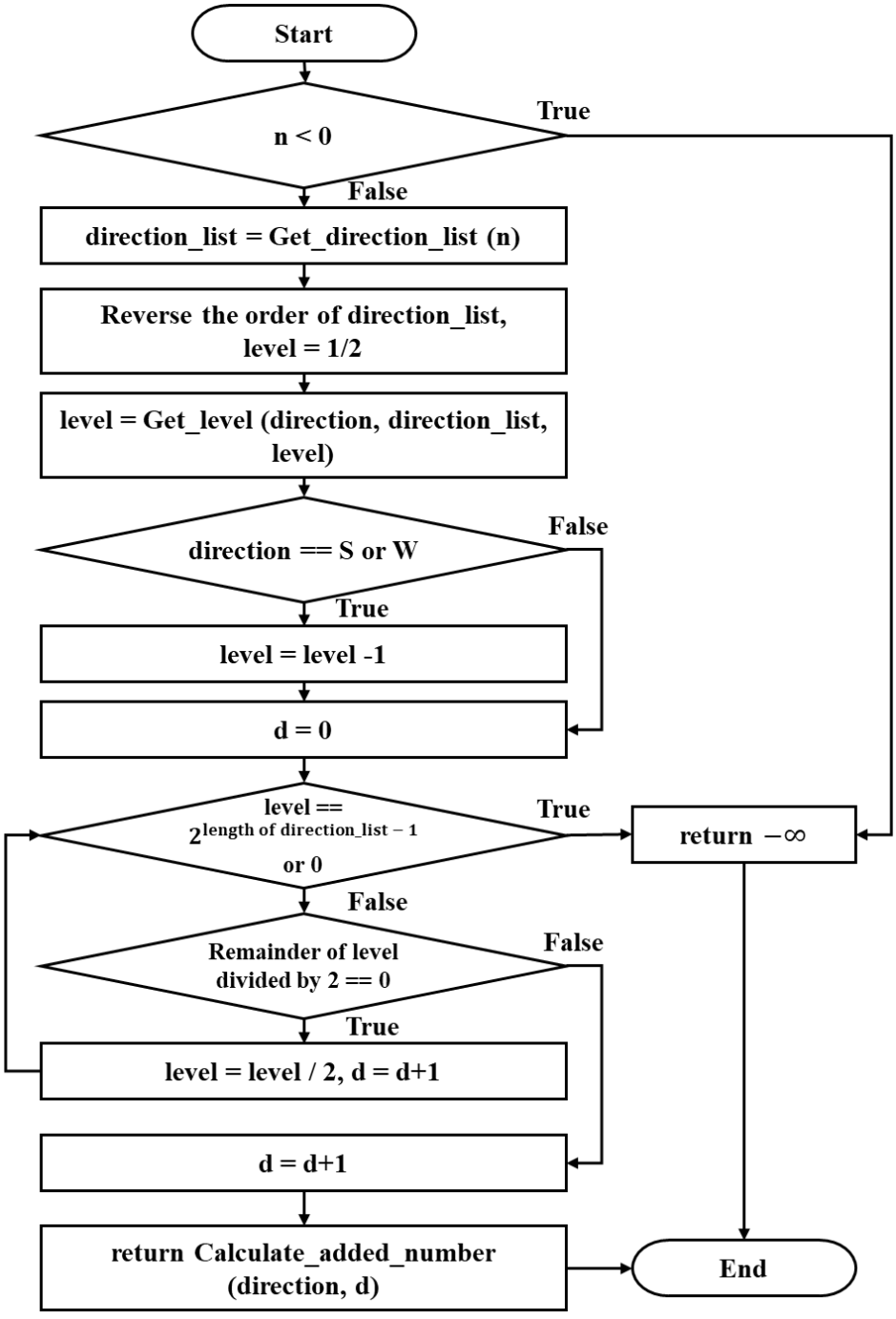


Figure 20. Find_four_cardinal_quadtree_nodes

The function Get_direction_list is described at Figure 21. The direction_list is initialized first. Then, if n is not 0, the remainder of n divided by 4 is added to the direction_list. If the remainder of n divided by 4 is 0, then the quotient of n divided by 4 minus 1 is assigned to n. If the remainder is not 0, the quotient of n divided by 4 is assigned to n. If the newly assigned n is 0, it means that all parent nodes have been searched, so a direction_list is returned.

Using the directon_list, column or row level of quadtree node moved in to the certain direction, which is depicted at the Figure 19 (b) is calculated through the function Get_level. Before applying the direction_list to the Get_level, it is necessary to reverse the order of the direction_list and the level should be initialized to 1/2. The detailed process of the Get_level is shown in Figure 22. 0, 1, 2, and 3 in the direction_list mean NE, SW, SE, and NW directions, respectively, and the direction means the direction of the neighboring quadtree node.

Regardless of the direction, if direction_list[i] is 0, twice the current level is assigned to the level. If direction_list[i] is 1, subtracting 1 from the doubled current level is assigned to the level. The direction should be considered if direction_list[i] is 2 or 3. If direction_list[i] is 2 and the direction is E or W, the level is doubled. If direction_list[i] is 2 and the direction is N or S, subtracting 1 from the doubled current level is assigned to the level. When direction_list[i] is 3, it is reversed from when direction_list[i] is 2. After the Get_level, if the direction is S or W, which are the directions of decreasing level, 1 is subtracted from the level.

In Figure 19, the left side of a quadtree node with 6 as n has 5 as n, and the right side has 9 as n. When moving to E and W, quadtree nodes under the same parent quadtree node differ by 1, and when moving to N and S, nodes under the same parent quadtree node differ

by 2. The d in Figure 20 is the difference in the Depth to reach the same parent quadtree node with the quadtree node in the direction to move. By dividing the level by 2, 1 is added to the d until the remainder of the level becomes 1.

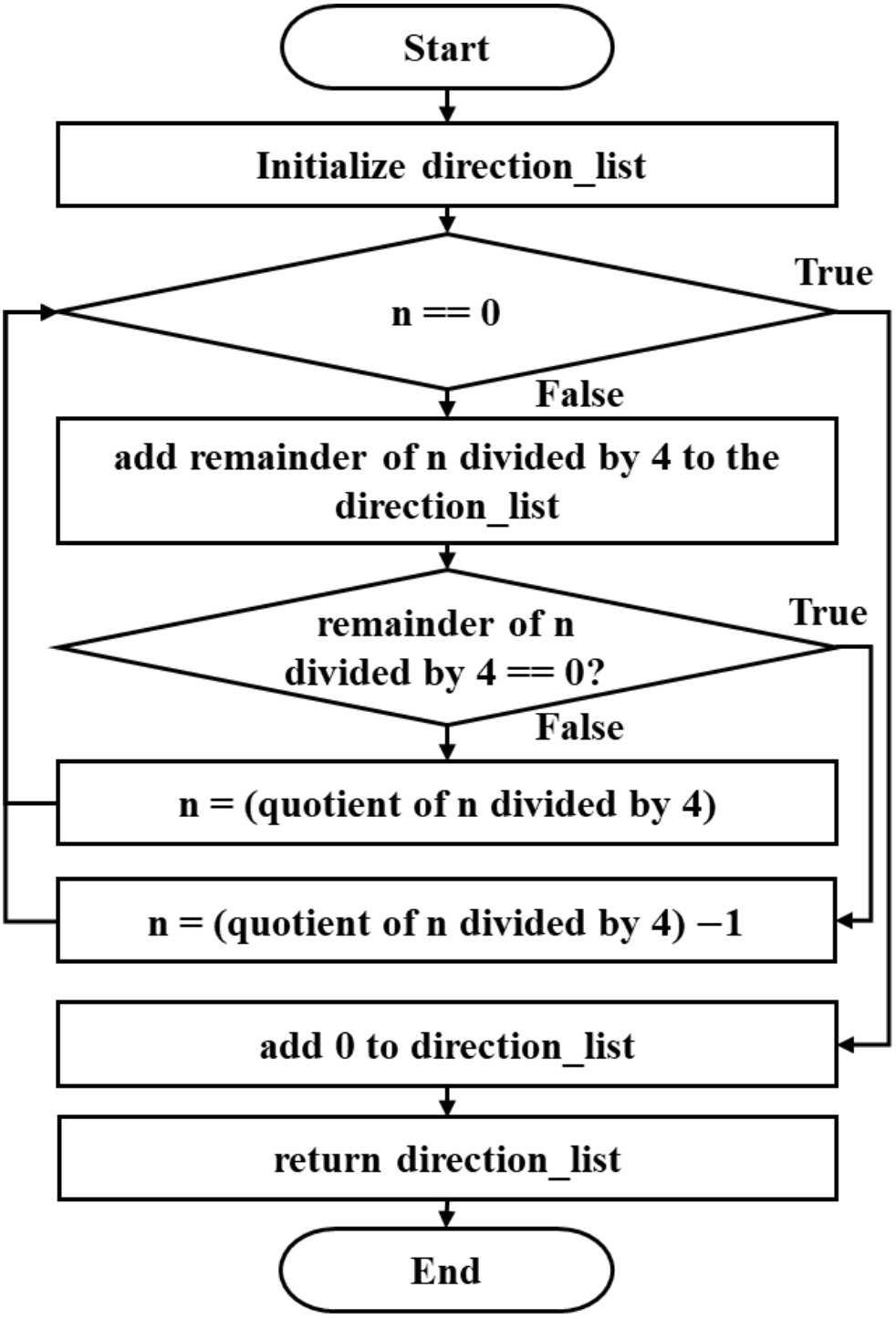


Figure 21. Get_direction_list

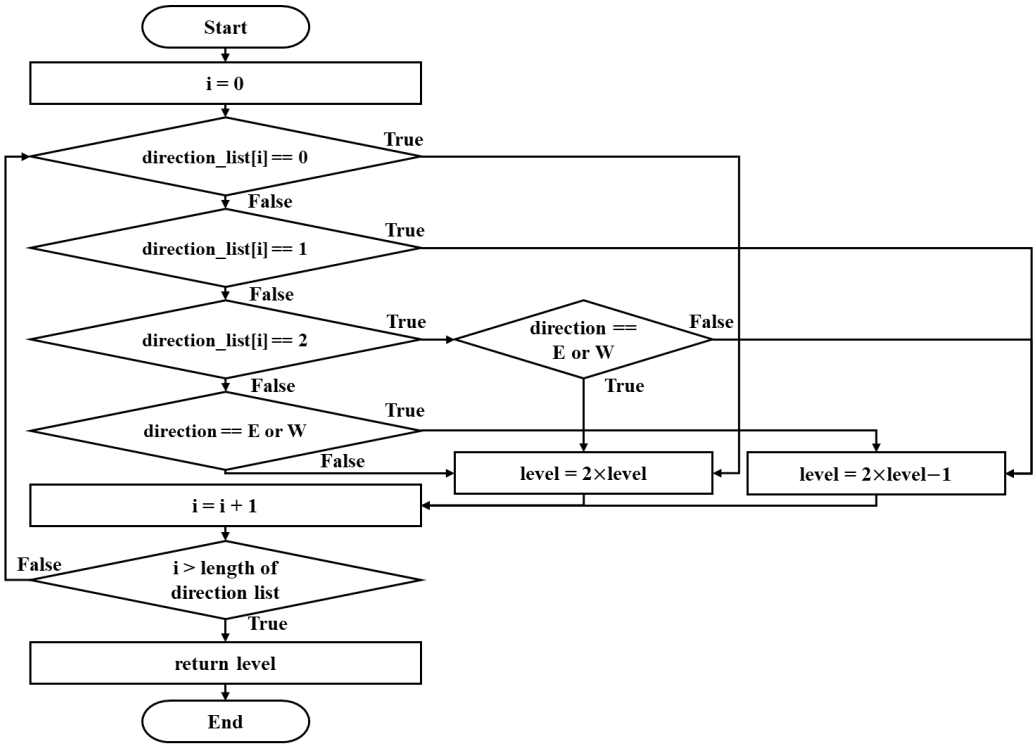


Figure 22. Get_level

The function Calculate_added_number in Figure 23 shows the formula of calculating the difference of n between two quadtree nodes with d. The n for the quadtree node moving in a certain direction is completed by adding a value returned by the Calculate_added_number to the n of the current quadtree node.

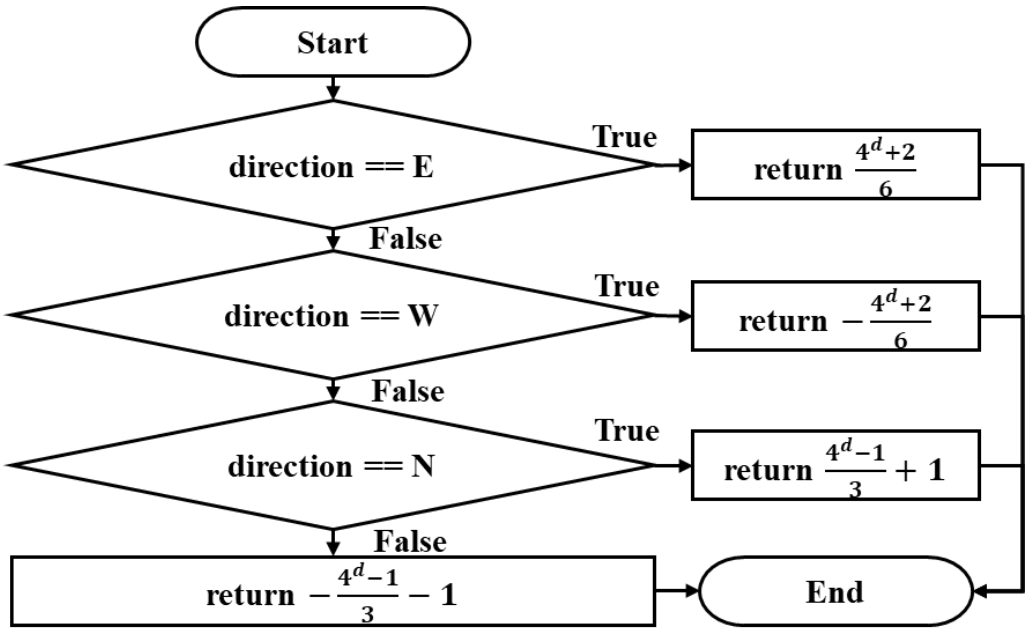


Figure 23. Calculate_added_number

The n calculated by the Find_four_cardinal_quadtree_nodes is the n considering the virtual quadtree leaf node, so it is necessary to convert it to the n existing on the actual quadtree chart. When a virtual quadtree node is not a quadtree leaf node, the virtual quadtree node must be converted to an actual quadtree node, and there are two cases. First, in the case of n, which does not exist on the current quadtree chart, it means that a quadtree leaf node exists at a depth smaller than the virtual quadtree node, so an actual quadtree leaf node that includes the area of the virtual quadtree node should be returned. The function Go_parent in Figure 24 finds the actual quadtree node. The n of the parent quadtree node is the quotient of the current n minus 1 divided by 4. Therefore, if n exists on the current quadtree chart, the n of the parent quadtree node is calculated. It repeats until the n, which is on the current quadtree chart, appears.

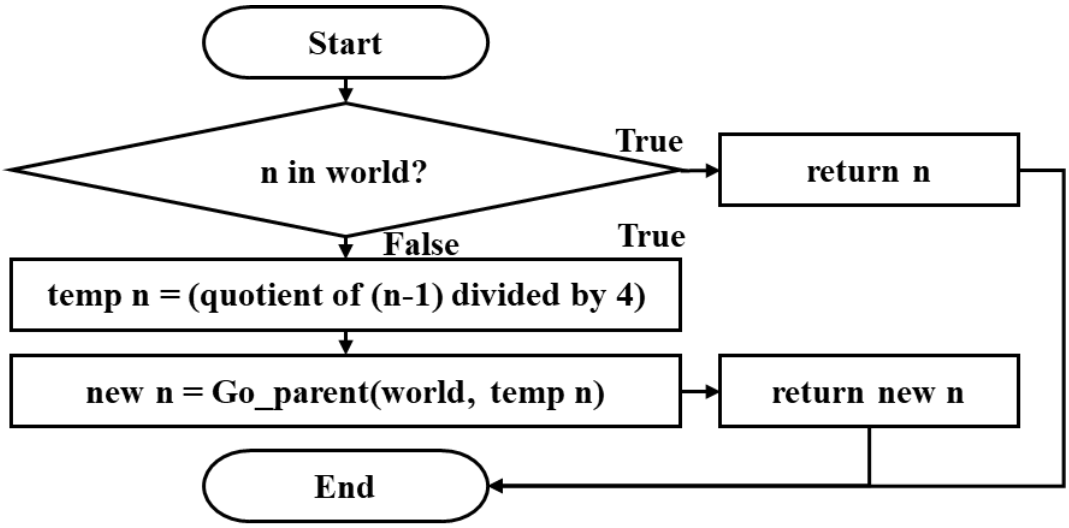


Figure 24. Go_parent

Second, if the Info of the virtual quadtree node is 2, it means that there is a quadtree leaf node. The function Get_children in Figure 25 shows how to calculate the neighboring quadtree leaf nodes with a virtual quadtree node. The children _list is initialized as an empty list, and n is n of the virtual quadtree node calculated by the Go_parent. For example, when moving in the E direction from the current quadtree node, the neighboring quadtree leaf nodes are located in the NW and SW directions among the child nodes of the virtual neighboring quadtree node. And, when moving in the N direction from the current quadtree node, the neighboring quadtree leaf nodes are located in the SE and SW directions among the child nodes of the virtual neighboring quadtree node. The Get_children is repeated and added to the children_list until the quadtree leaf nodes are found. As a result, the children_list has all quadtree leaf nodes moved in one direction.

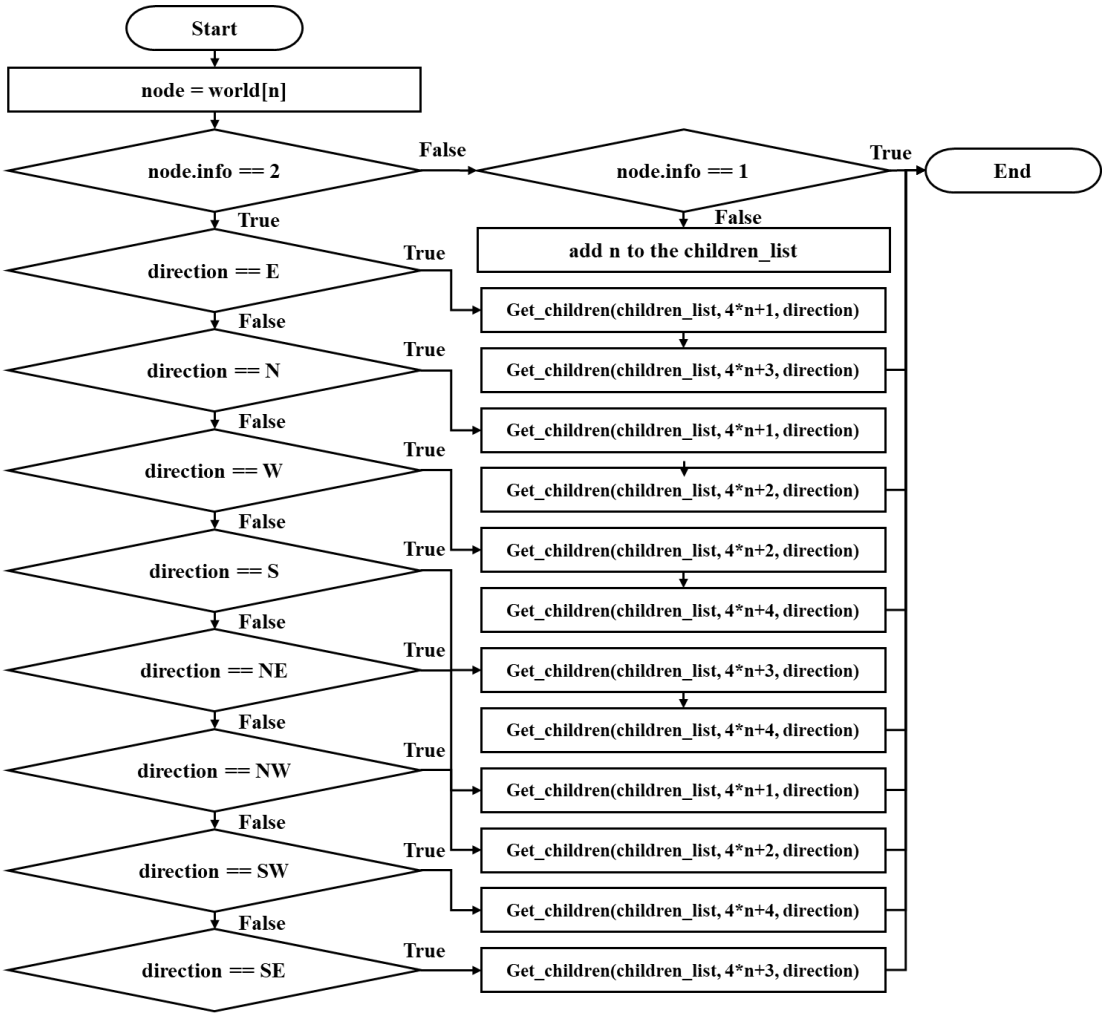


Figure 25. Get_children

The neighboring nodes in four cardinal directions have been calculated so far. The neighboring quadtree nodes that exist in the diagonal direction from the current quadtree node are obtained by the function Get_diagonal_n in Figure 26.

In a regular grid, the node moved in the N direction and moved in the E direction is the same with the node moved in the E direction and moved in the N direction. However, they are not the same in the quadtree. Therefore, the order of N direction and E direction must be determined in order to move in the NE direction.

In this study, two candidates of quadtree nodes were used to determine the order of the directions. The n1 and n2 are the n of the virtual quadtree nodes in a specific direction. For example, to find n in the NE direction, the values substituted for n1 and n2 are n + Find_four_cardinal_quadtree_nodes(n, N), n + Find_four_cardinal_quadtree_nodes(n, E). Looking closely at Figure 26, if n1 and n2 are less than 0, negative infinity was returned because the quadtree node outside the chart was selected.

Then, if n does not exist on the current quadtree chart, it is moved to a quadtree leaf node through the Go_parent, and if it exists on the current chart, even if it is not a quadtree leaf node, it is saved. After that, the quadtree node with the smaller Width between the two nodes was moved first. Like the four cardinal quadtree nodes, since the nodes could be the virtual quadtree nodes, the diagonal quadtree nodes are finally determined after the Go_parent and the Get_children. After searching all neighboring directions, duplicate nodes are removed.

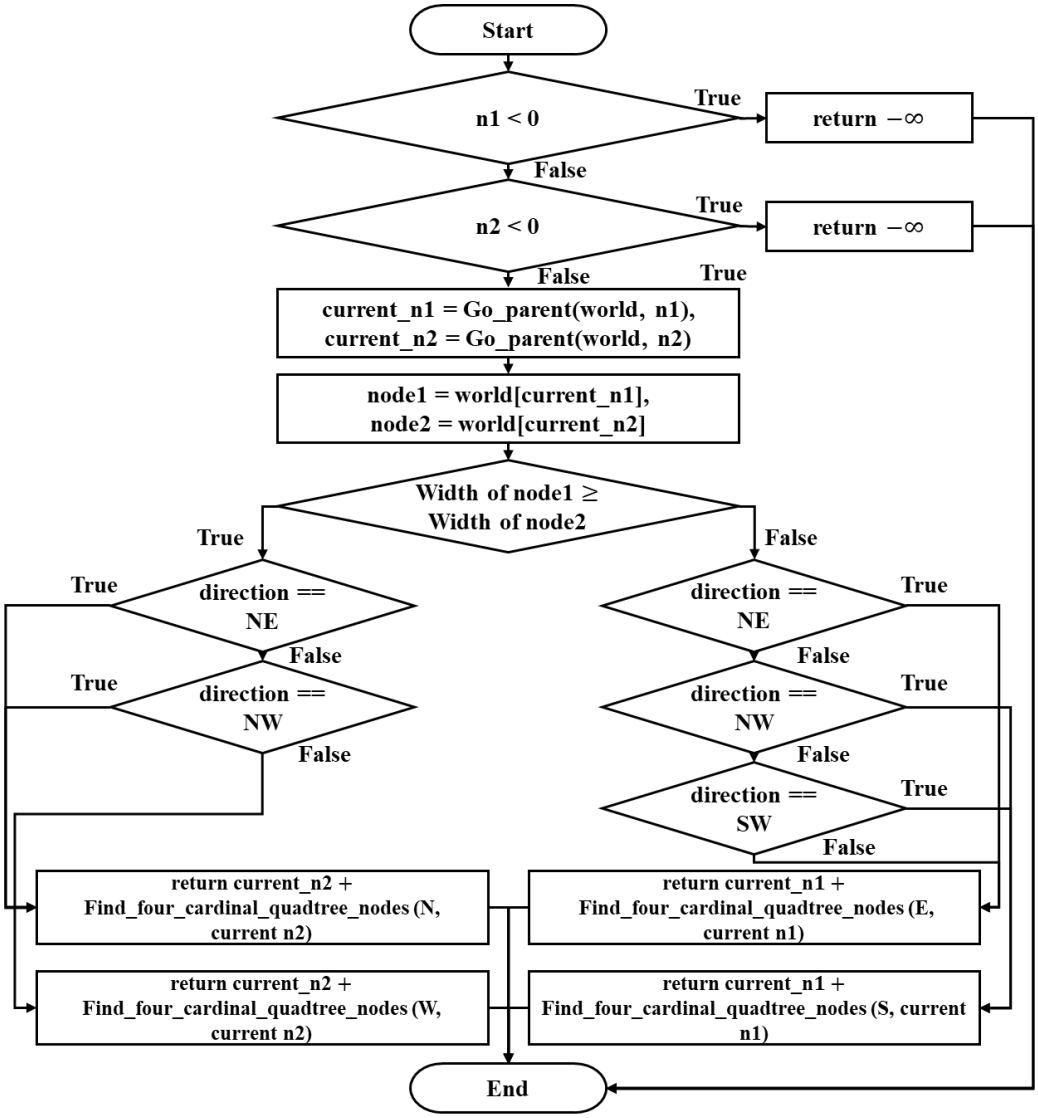
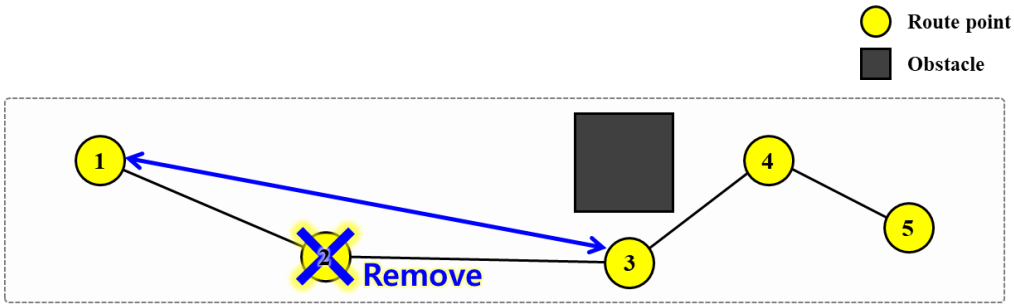


Figure 26. Get_diagonal_n

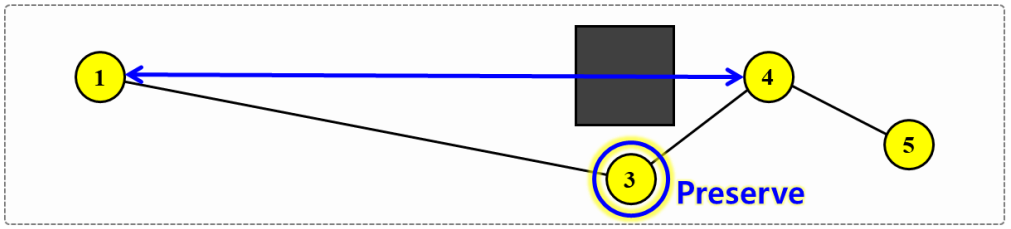
After finding the neighboring quadtree nodes, the search proceeds in the order of the quadtree nodes with the lowest sum of the cost and the heuristic function. For route planning outside the marina at the low-level chart, the euclidean distance between the

centers of the two quadtree nodes was used as a cost and a heuristic function.

To improve the quality of the route planned using A* on the quadtree chart, a smoothing process is required. Route planned on a quadtree is not an optimal route for selecting the center of a node as a route, similar to a regular grid. The smoothing on a route planned on a quadtree chart is as follows. Based on three consecutive route points, as shown in Figure 27 (a), if the line segment connecting the two route points at both ends and the obstacle do not meet, the route point in the middle is removed. If the line segment connecting the two route points at both ends and the obstacle meet, the route point in the middle is preserved, as shown in Figure 27 (b). The process mentioned above is repeated until the last three points of the route are reached.



(a) Example of removing a route point



(b) Example of preserving a route point

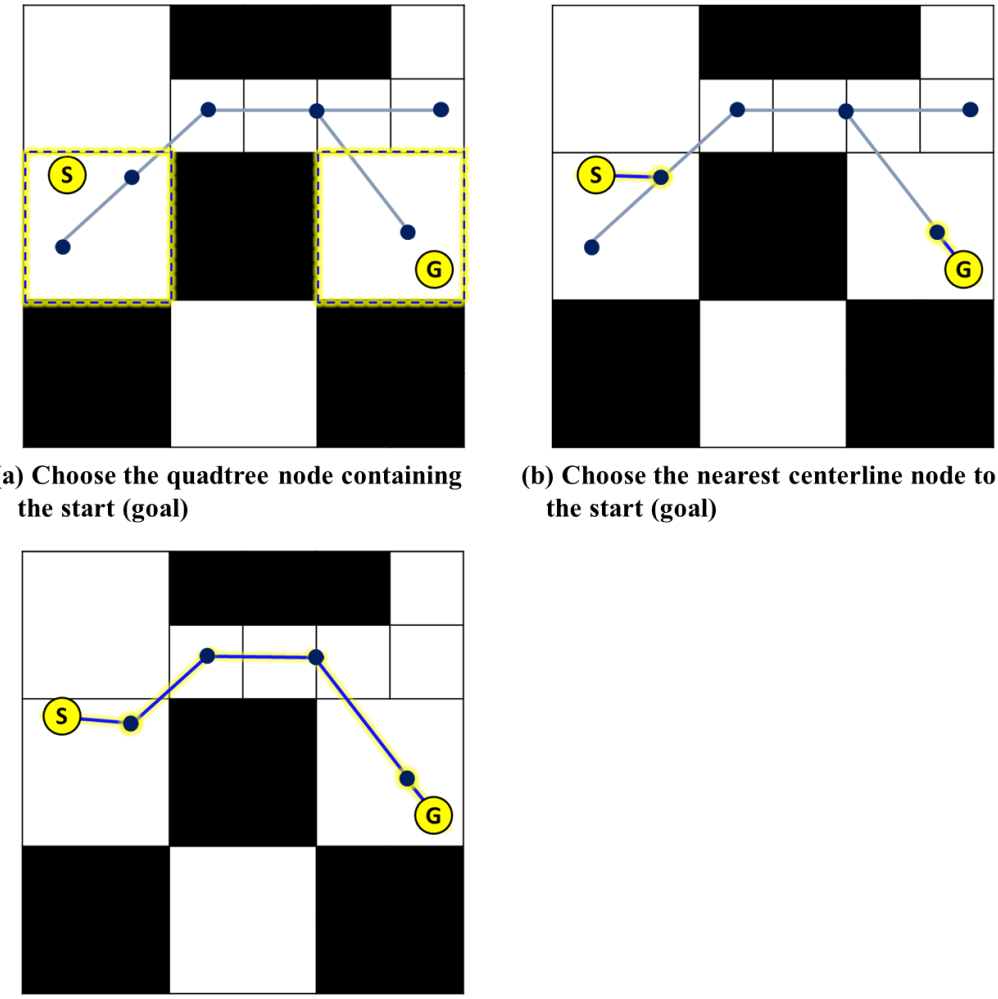
Figure 27. Examples of smoothing method in the outside of the marina at low chart

3.3. Route planning inside the marina at low level-chart

For the route planning inside the marina at low-level chart, the centerline of the marina based on the Voronoi diagram was used as the route planning chart inside the marina. As at 2.3.1, it is necessary to plan a route that maintains a certain distance from the marina obstacles, so the centerline nodes of the marina are selected as candidates for the route. When a departure is inside the marina, a process of connecting to the centerline node of the marina is required.

The quadtree chart is used to find the closest centerline node to the departure. Every quadtree node inside the marina is connected to the centerline nodes inside the quadtree node. If there is no centerline node inside a quadtree node, the closest centerline node among the centerline nodes with no obstacles between the centerline node and the quadtree node is connected to a quadtree node. The closest centerline node among the centerline nodes connected to a quadtree node, including departure, is selected as the start centerline node. The goal centerline node is determined in the same way as the start centerline node. The process connecting the quadtree nodes and the centerline nodes is done when generating a centerline chart. When the start node and the goal node are determined, the

route is planned using the A*.



(a) Choose the quadtree node containing the start (goal)

(b) Choose the nearest centerline node to the start (goal)

(c) Find the route of the centerline chart

Figure 28. Route planning inside of the marina at low level chart

Since the centerline chart generated using the Voronoi diagram is not intended to plan

the shortest route, the route planned on the centerline chart needs an additional smoothing process. The smoothing process is required to maintain a certain distance between the route and obstacles and change route points that are excessively detoured.

The diagram of the smoothing route is in Figure 29. The route planned along the centerline is composed of many nodes, as shown in Figure 29 (a) because of the short distance between the centerline nodes.

First, the number of route points was reduced by utilizing the Douglas-Peucker algorithm. The Douglas-Peucker algorithm is an algorithm that simplifies the polygon by removing the points. If the point between a line segment connecting two points of the polygon is less than ε, it is removed in the Douglas-Peucker algorithm. Snoeyink, n.d [31] The number of executions of subsequent processes is reduced by reducing the number of route points through the Douglas-Peucker algorithm. There are the three cases for modifying the route points.

Figure 29 (c) shows the case where the middle route point is removed among the three consecutive route points. The closest point between the obstacles and the line segment $\overline{AC}$ connecting both ends among the three consecutive route points $\mathrm{A}, \mathrm{B}, \mathrm{C}$ is X_1, and the closest point between the $\overline{AC}$ and X_1 is X_2, The point where the line extending $\overline{X_1X_2}$ and the line segments $\overline{AB}, \overline{BC}$ meet is X_3. The length of the $\overline{X_1X_2}$ is set to d_1, and the length of the$\overline{X_2X_3}$ is set to d_2. If $\frac{d_1}{d_1+d_2}$ is equal or more than α, the route point in the middle is removed. If $\frac{d_1}{d_1+d_2}$ is less than α, as shown in Figure 29 (d), the route point in the middle is replaced to the new point. In this study, α is 0.9. The Figure 29 (e) shows the process replacing a route point. New route point is the point where the line segment $\overline{X_1X_3}$ is divided in the ratio of $(1-\alpha):\alpha$.

Figure 29 (f) and (g) show the process of replacing with a new route point when the line segment connecting both ends of the three consecutive route points overlaps with an obstacle. A new route point cannot be produced by interpolating the $\overline{CE}$ and obstacles because the $\overline{CE}$ is overlapped with the obstacles. Hence, a new route point is decided by interpolating the middle point of the three consecutive route points and the closest point on the obstacles with the line segments $\overline{CD}, \overline{DE}, \mathrm{X}_1$, in β: $(1-\beta)$. In this study, β is 0.65.

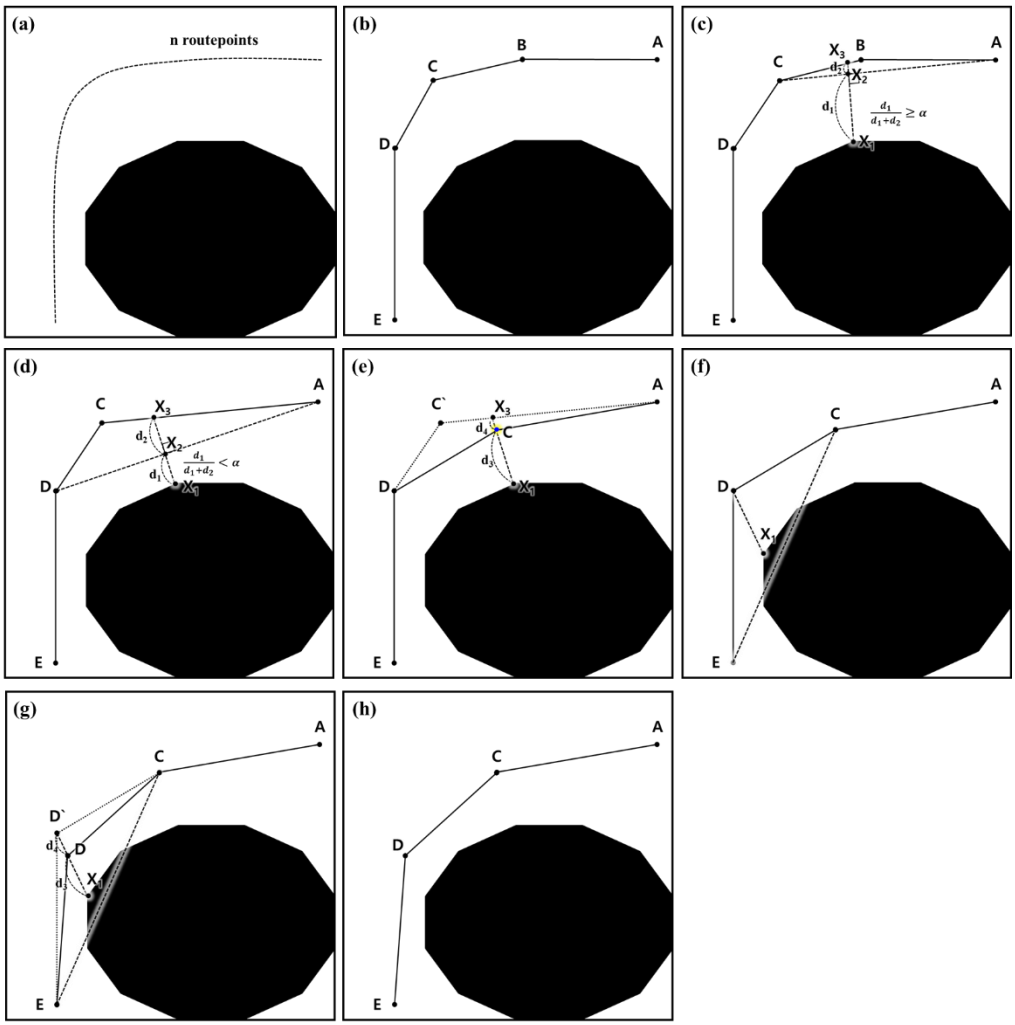


Figure 29. Examples of smoothing method inside of the marina at low chart

3.4. Combining routes inside of the marina and outside of the marina at low-level chart

Through the route planning at the high-level chart, the entrances were determined. When the pair of entrances is inside the marina, the route planning inside the marina is

applied. When the pair of entrances is outside the marina, the route planning outside the marina is applied. After the route planning at all low-level charts is completed, all routes are integrated in order.

An additional smoothing process is not required when the route planned outside the marina and the route planned inside the marina are integrated. However, an additional smoothing process is necessary to improve the route quality when integrating the routes planned outside. Figure 30 is the flowchart of smoothing when integrating the routes outside the marina. If all routes are processed at once, the entire route planning time and memory usage increase due to loading all charts where routes are. Therefore, smoothing is sequentially performed on two adjacent quadtree charts in this study.

The i in Figure 30 is the number corresponding to the chart containing the route. The j and k mean the order of route points of routes belonging to the i th charts and $i+1$ th charts, respectively. It is determined whether there is an obstacle between the j th route point of the i th chart and the k th route point of the $i+1$ th chart. If there is no obstacle, all route points between the two route points are removed. If there is an obstacle, whether there is an obstacle between the $j+1$ th route point of the i th chart and the k th route point of the $i+1$ th chart is checked. When all obstacles are determined for all waypoints of the first chart, the previous process is repeated based on the $k-1$ th waypoint of the second chart. When there are no obstacles between all route points at the i th chart and the k th route point at the $i+1$ th chart, checking the obstacles between route points at the i th chart and the $k-1$ th route point at the $i+1$th chart is repeated. The process above is repeated until the last route is integrated.

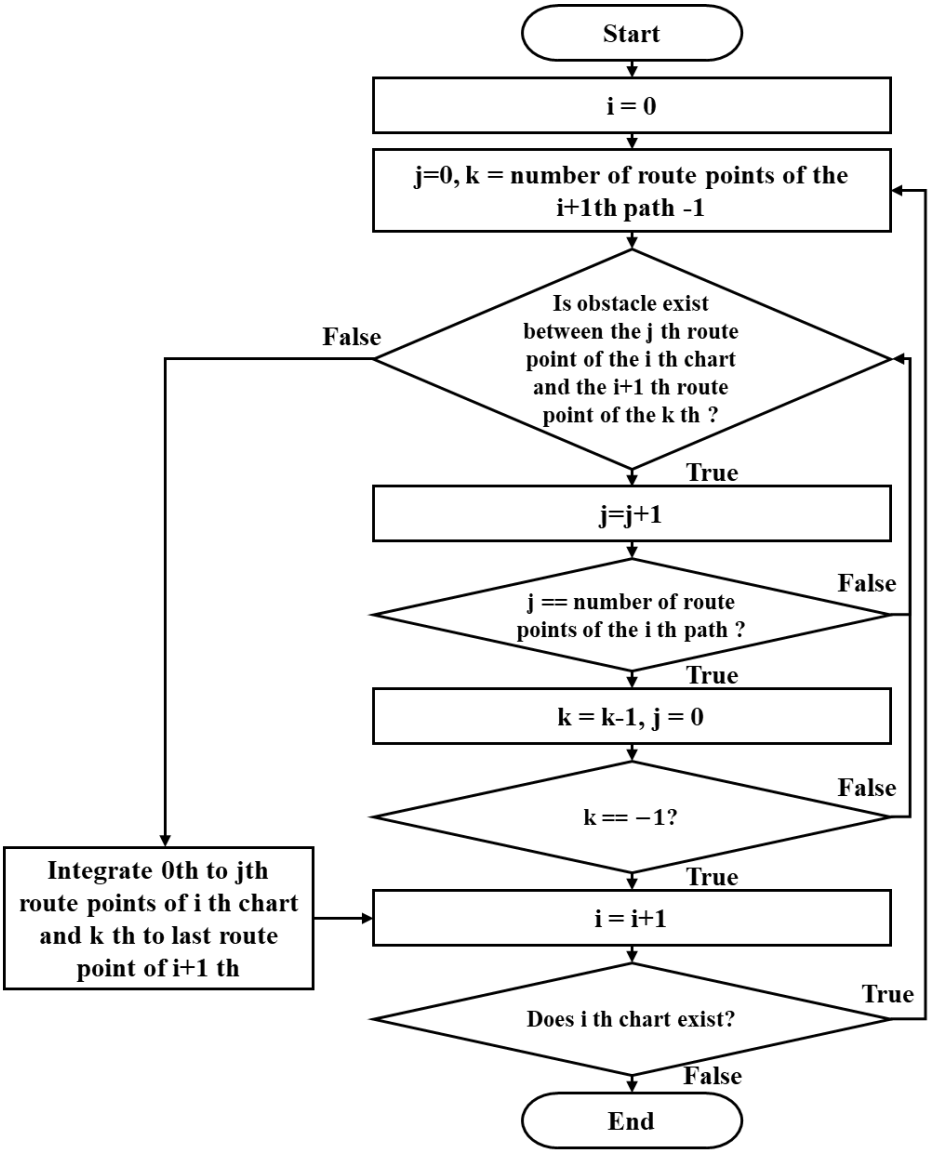


Figure 30. Flowchart of smoothing when integrating the routes outside the marina.

4. Applications

4.1. Implementation results of a high-level chart

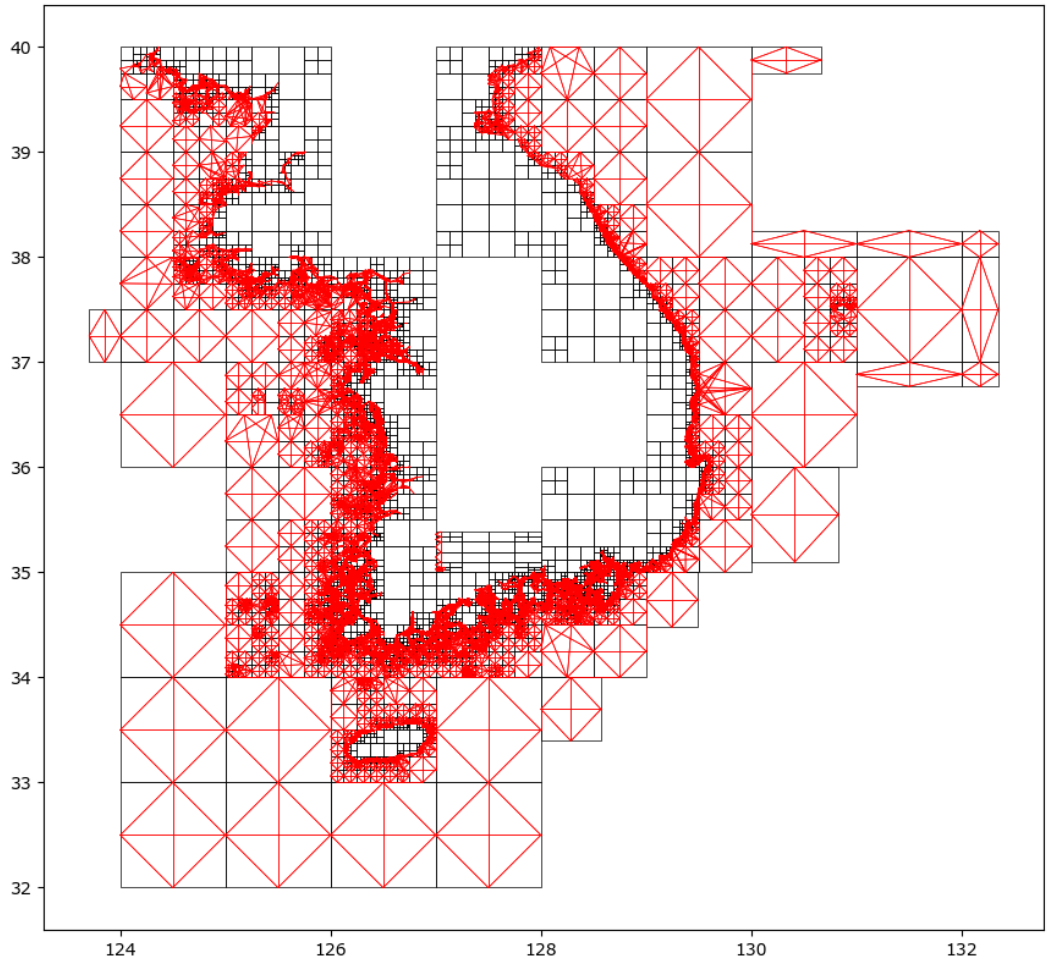


Figure 31. Result of a high-level chart of the Korea

Figure 31 is the result of a high-level chart in Korea. The red lines are the edges connecting the entrances and the square in black is the area of a single low-level chart. As mentioned in 3.1, the entrances are placed at the side of the low-level chart. Since it is divided based on the number of nodes, a coastal area with a large number of nodes has a large number of low-level charts.

4.2. Implementation results of quadtree charts

In this study, the quadtree charts of the 51 coastal areas of Korea and some coastal areas of Miami were generated. Among the objects specified in S-57, 23 objects, such as land area, marine farm, and obstruction, which must be avoided when a ship is sailing, were used to create the route planning charts. Considering that the standard for small ships is 12 m, in the case of a general object, the length of one side of the smallest quadtree node of a chart has a value between 7 and 15 m. However, the objects, such as the pontoon, require a higher resolution. The maximum number of the decomposition for those objects is increased so that the length of one side of the smallest quadtree node is 1 to 2 m for free passage of the small ship.

In addition to the 23 objects reflected as obstacles, the sounding of S-57 was used as water depth information. In the S-57, three main objects express water depth. First, the depth area is an object that expresses the maximum and minimum water depth within the defined area. Second, the depth contour is an object created by connecting points with the same water depth. Finally, sounding is a measured water depth in specific latitude and longitude coordinates. The area with negative water depth is where the seabed is exposed

by the ebb tide. International Hydrographic Organization [25] To apply the water depth on the integrated quadtree chart, the average of the sounding included in the area of the quadtree node is the Water depth of the quadtree node. Only the quadtree nodes whose Info is 1 have the value on the Water_depth.

Figure 32 shows the results of the quadtree charts generated by the proposed method. Figure 32 (a) is the southern coast of Korea, and Figure 32 (b) is an enlarged picture of the highlighted blue box in Figure 32 (a). Figure 32 (c) is the west coast of Korea, and Figure 32 (d) is an enlarged picture of the highlighted blue box in Figure 32 (c). In Figure 32, many objects are expressed in different colors. The dark goldenrod quadtree nodes are the marine farm. The red quadtree nodes are the obstruction. The blue quadtree nodes are the seabed area. The midnight blue nodes are the pontoon, and the black node are the other obstacles. The white quadtree nodes are where ships can pass, and the gray quadtree nodes are the area where the depth of the node is less than 0. The complex coastal areas with many obstacles are well represented with the quadtree in high resolution.

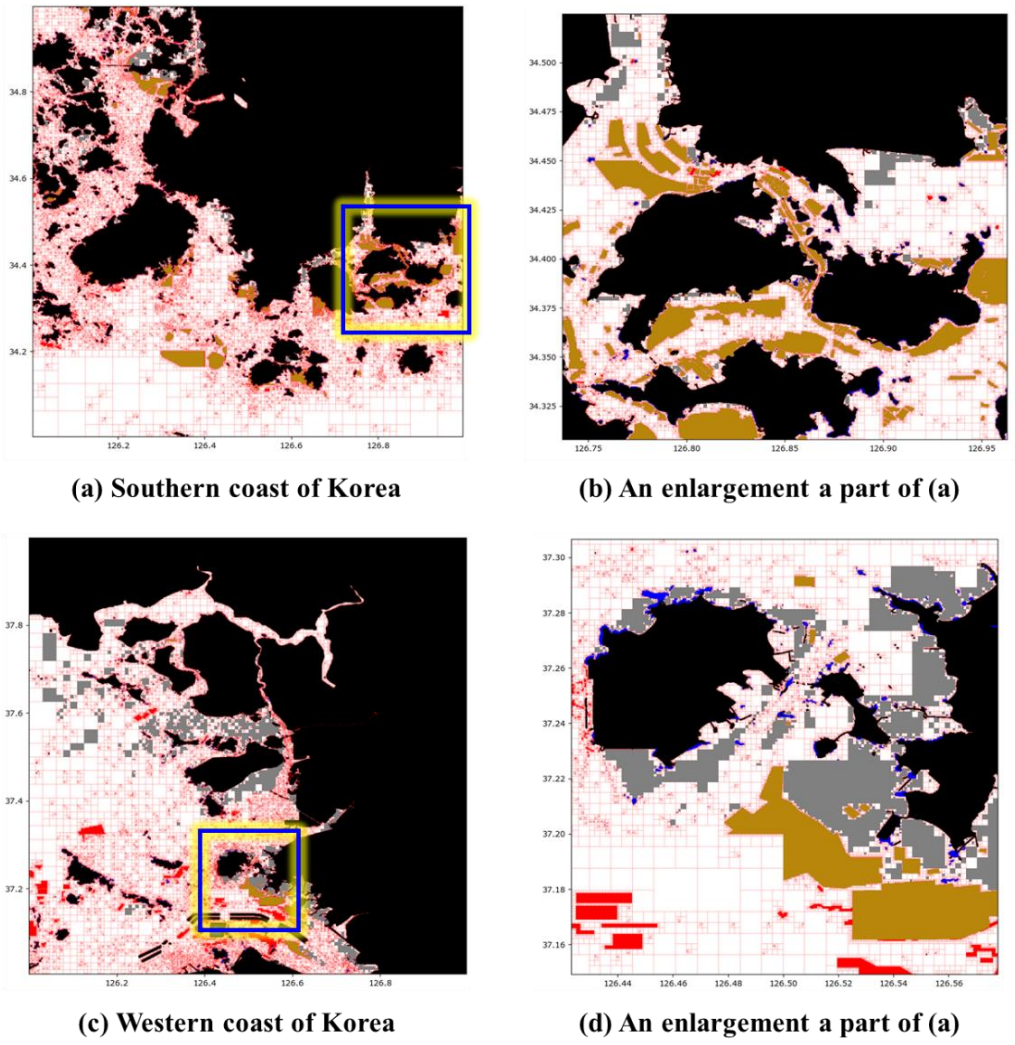
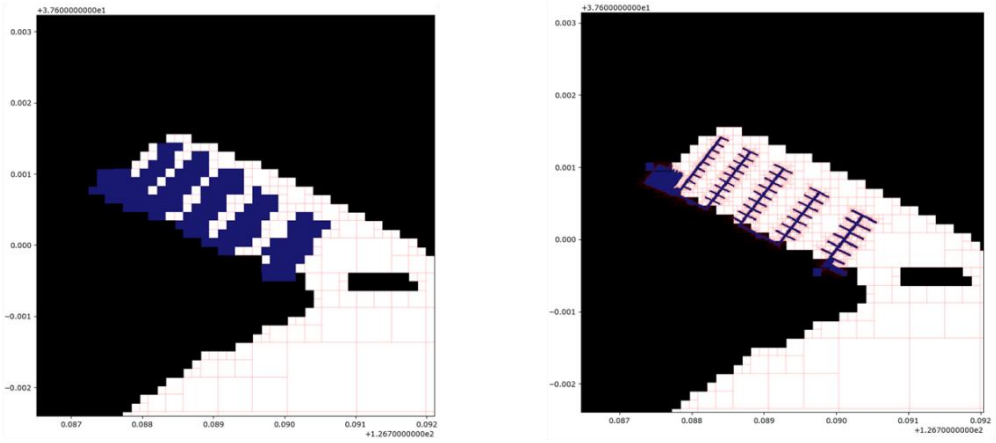


Figure 32. Results of the quadtree charts

Figure 33 is an example of a quadtree chart with different MAX_DEPTH. Comparing the pontoons in midnight blue in Figure 33, Figure 33 (a) shows the pontoon with the same MAX_DEPTH with the MAX_DEPTH of the quadtree node of the normal obstacles, and it seems inappropriate to plan a route inside the pontoon for a small ship. Figure 33 (b)

shows the pontoon with a higher MAX_DEPTH than the MAX_DEPTH of the quadtree node of the normal obstacles. Only the pontoon object is created with a smaller minimum quadtree node size, so the file size of the quadtree chart is slightly increased, and the pontoon is represented much more elaborately than Figure 33 (a).



(a) Pontoon with normal size of minimum node

(b) Pontoon with smaller size of minimum node

Figure 33. Comparison of objects with different sizes of minimum quadtree nodes

The charts represented in the quadtree, regular grid, and regular grid converted to the Compressed Spares Row (CSR) format. Table 3 compares the file size of the charts of the 51 coastal areas of Korea by representation method.

The diagonal length of one grid of the regular grid and the diagonal length of the minimum quadtree node of the normal obstacles are the same. However, the objects requiring a higher resolution were made with a smaller minimum quadtree node, as shown

in Figure 33 (b).

The grids of the regular grid chart have information on whether the vessel can pass or not and the water depth, but the quadtree nodes have information about water depth and various properties of the quadtree node. The CSR format, which efficiently represents the sparse matrix, is used to represent the matrix with the elements of the matrix, row index, and information of the column. The charts represented in CSR format have two sorts of the chart, which is about the topography and the water depth.

The average file size of the regular grid chart is 928.69 MB, and the average file size of the charts with CSR format is 512.79MB by summing the topography and water depth information. The average file size of the quadtree chart is 30.52MB, which is much lighter than the other methods. Therefore, it was confirmed that the quadtree is an efficient method for representing coastal areas.

Table 3. Comparison between the regular grid charts and the quadtree charts

	Regular grid	CSR format			Quadtree
		Topography	Water depth	Sum	
Average file size	928.69 MB	177.70 MB	335.09 MB	512.79MB	30.52MB

4.3. Implementation results of centerline charts

The centerline chart inside the marina for Miami was generated because there are many adequate marinas for modeling. After defining the shape of the marina, the method

mentioned in 2.3 was used to generate the centerline chart inside the marina, and the result is shown in Figure 34. The lines in maroon on the charts are the centerline of the marina. The distance between the centerline nodes is 12 m, which is suitable for route planning of small ships. It can be seen that a centerline is generated only inside the marina. Moreover, two centerlines are generated at the entrance for safe navigation. They prevent collisions between outgoing and incoming ships and guide the direction because they are directed graphs.

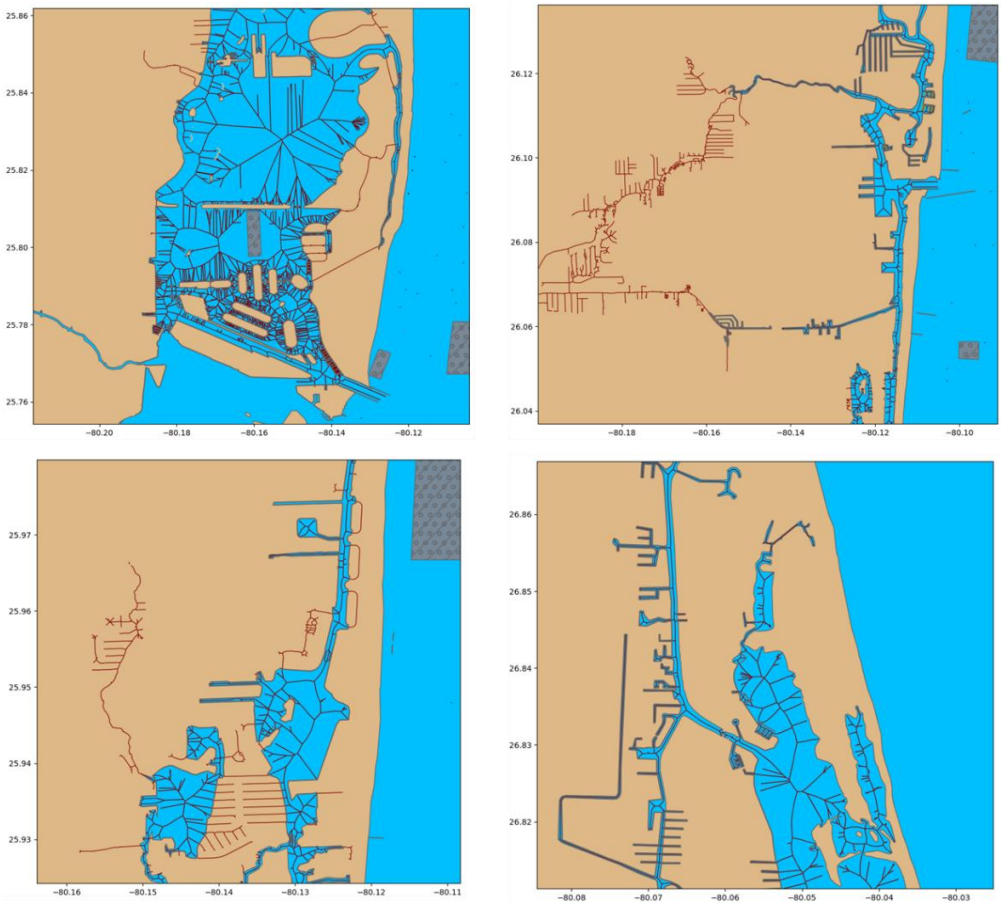


Figure 34. Example of centerlines in marina

4.4. Simulation results

Simulations were conducted in various environments to verify the method presented in this study. The simulation was conducted using Python 3.8 on Windows 10, an Intel Core i7-10700 CPU, and 32.0GB RAM.

In the visualized result, the dark red line is the route, the yellow dot on the route is the starting point, the dark orange dot is the destination point, and the dark blue dots are the route points. The land is in burlywood, the marine farm is in dark blue, the obstruction is in gray, the pontoon is in dark goldenrod, and the other objects are in black.

4.4.1. Comparison between other route planning methods

The method proposed in this study was compared with the following four methods based on the generated route and route planning time, and the results are shown in Table 4. The method proposed in this study was compared with the following four methods based on the generated route and route planning time, and the results are shown in Table 4. The time presented in the table is the route planning time which includes time for reading the chart for route planning, time for planning the route, and time for smoothing. Routes 1, 2, 3, and 4 are the results of routes outside the marina in Korea, and Routes 5, 6, and 7 are the results of routes that pass through the marina on the coast of Miami.

Route planning time was consumed in the order of Dijkstra on regular grid charts, A* on regular grid charts, Dijkstra on quadtree charts, A* on quadtree charts, and the method proposed in this study except for Route 7. There are three reasons for this result. First, this is because A* is generally faster than Dijkstra. Second, the number of total nodes searched

for quadtree charts is smaller than that of regular grid charts. Third, the number of route points to be smoothed on quadtree charts is smaller.

In the case of Route 7, as shown in Figure 38, because it is a very narrow channel, most of the surrounding nodes are nodes of the smallest size. Therefore, there is little difference in the number of searched nodes between route planning on the regular grid chart and the quadtree chart. If the number of searched nodes is similar, the method of searching the neighboring nodes on the quadtree chart is more complicated, so route planning on the quadtree chart took longer than route planning on the regular grid chart in Route 7.

The method presented in this study is much faster than other methods in Routes 1, 2, 3, and 4 where only route planning outside the marina is applied. The route planning time of the proposed method is also shorter than the other algorithms in Route 5 and Route 6, even though routes are planned by combining routes planned outside the marina and inside the marina. For Route 7, the proposed method only uses route planning inside the marina, so there is no big difference with the second fastest method using A* in the regular grid chart.

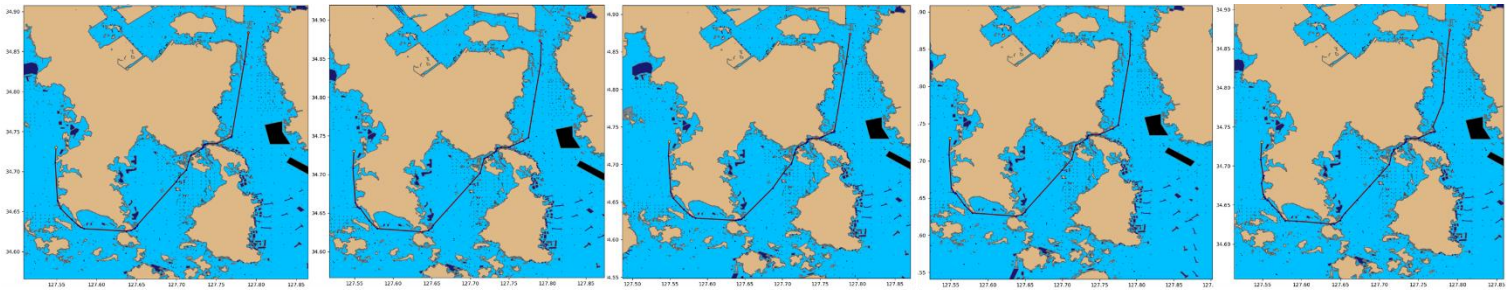
It was confirmed that the length of the route is determined by the representation method of the chart rather than the route planning algorithm. In general, the distance of the route generated from the regular grid chart is slightly shorter than the route generated from the quadtree chart. Due to the nature of the quadtree chart, the center of the quadtree node with a non-uniform size is selected as a route point, so even if the smoothing process is performed, the distance of the entire route planned on the quadtree chart is longer than the route planned on the regular grid charts.

The distance of the route planned by the proposed method is slightly longer than that of the other algorithms. This is because the route is planned hierarchically with high-level and

low-level charts. The entrance in the high-level chart and the quadtree chart in the low-level chart make the route longer. In particular, the route planned by route planning inside the marina is along the centerline chart, which is not for the shortest route, so it is longer than the other routes. However, comparing (a), (b), (c), and (d) in Figure 38, which are the routes planned on the regular grid chart, and (e) in Figure 38, which is the route planned by the proposed method, is the safest route that maintains a certain distance from the obstacles.

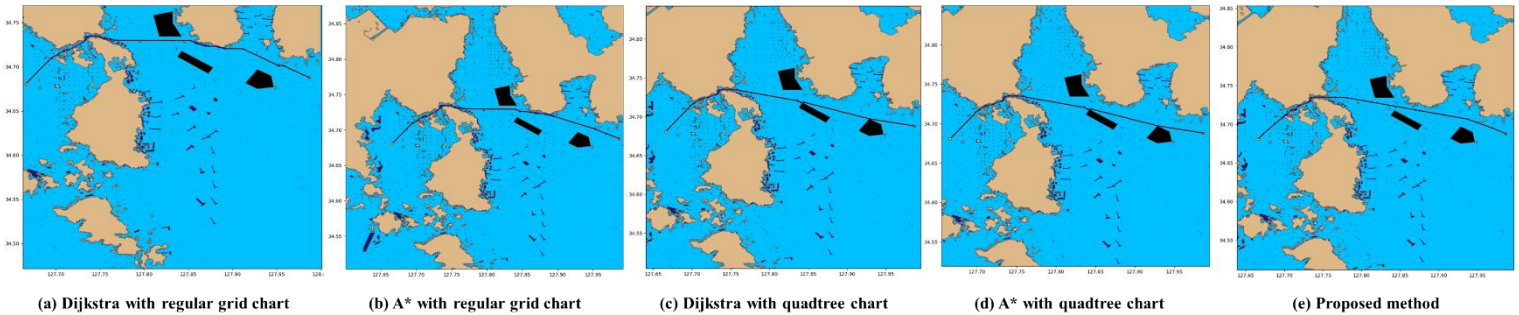
Table 4. Route planning time according to the method

		Regular grid		Quadtree		Proposed method
		Dijkstra	A*	Dijkstra	A*	
Route1	Time (s)	852.16	133.54	36.49	19.07	1.77
	Distance (km)	50.70	50.70	51.04	51.04	51.36
Route2	Time (s)	615.35	49.97	34.76	13.93	1.24
	Distance (km)	33.19	32.95	32.70	32.70	32.92
Route3	Time (s)	222.43	52.56	24.38	12.80	1.24
	Distance (km)	34.64	34.68	34.10	34.10	35.54
Route4	Time (s)	1581.85	123.86	35.58	13.66	2.21
	Distance (km)	47.53	47.47	47.39	47.39	48.33
Route5	Time (s)	42.77	4.61	3.95	2.01	1.02
	Distance (km)	7.55	7.55	7.57	7.57	7.76
Route6	Time (s)	59.43	5.14	3.48	2.36	1.60
	Distance (km)	7.48	7.51	7.49	7.49	7.65
Route7	Time (s)	3.26	1.08	5.01	3.96	0.837
	Distance (km)	3.29	3.29	3.19	3.19	3.27



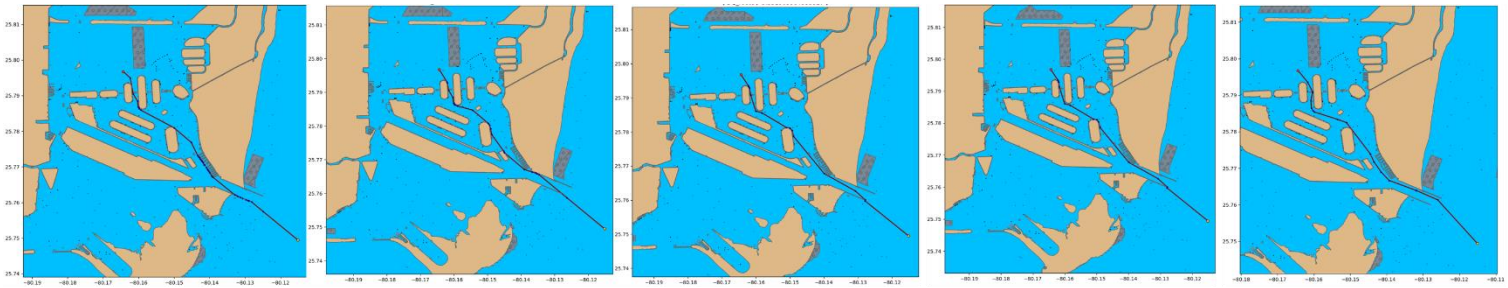
(a) Dijkstra with regular grid chart (b) A* with regular grid chart (c) Dijkstra with quadtree chart (d) A* with quadtree chart (e) Proposed method

Figure 35. Route 1 in Table 4



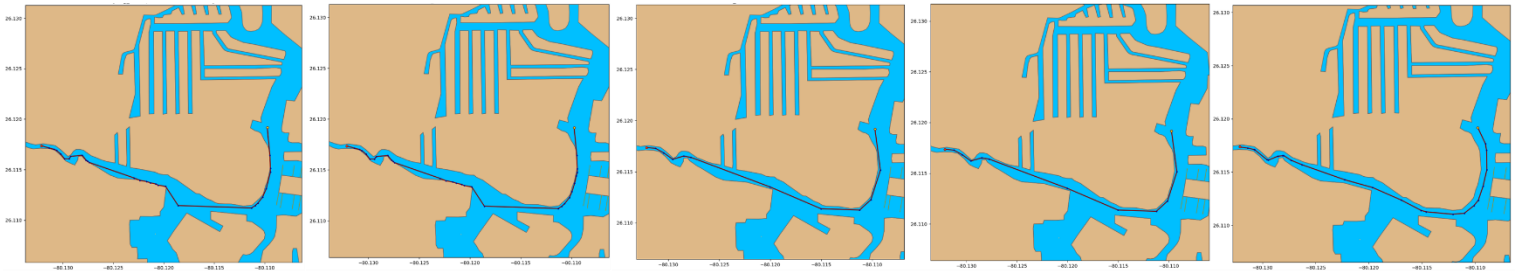
(a) Dijkstra with regular grid chart (b) A* with regular grid chart (c) Dijkstra with quadtree chart (d) A* with quadtree chart (e) Proposed method

Figure 36. Route 2 in Table 4



(a) Dijkstra with regular grid chart (b) A* with regular grid chart (c) Dijkstra with quadtree chart (d) A* with quadtree chart (e) Proposed method

Figure 37. Route 6 in Table 4



(a) Dijkstra with regular grid chart (b) A* with regular grid chart (c) Dijkstra with quadtree chart (d) A* with quadtree chart (e) Proposed method

Figure 38. Route 7 in Table 4

4.4.2. Verification in various areas

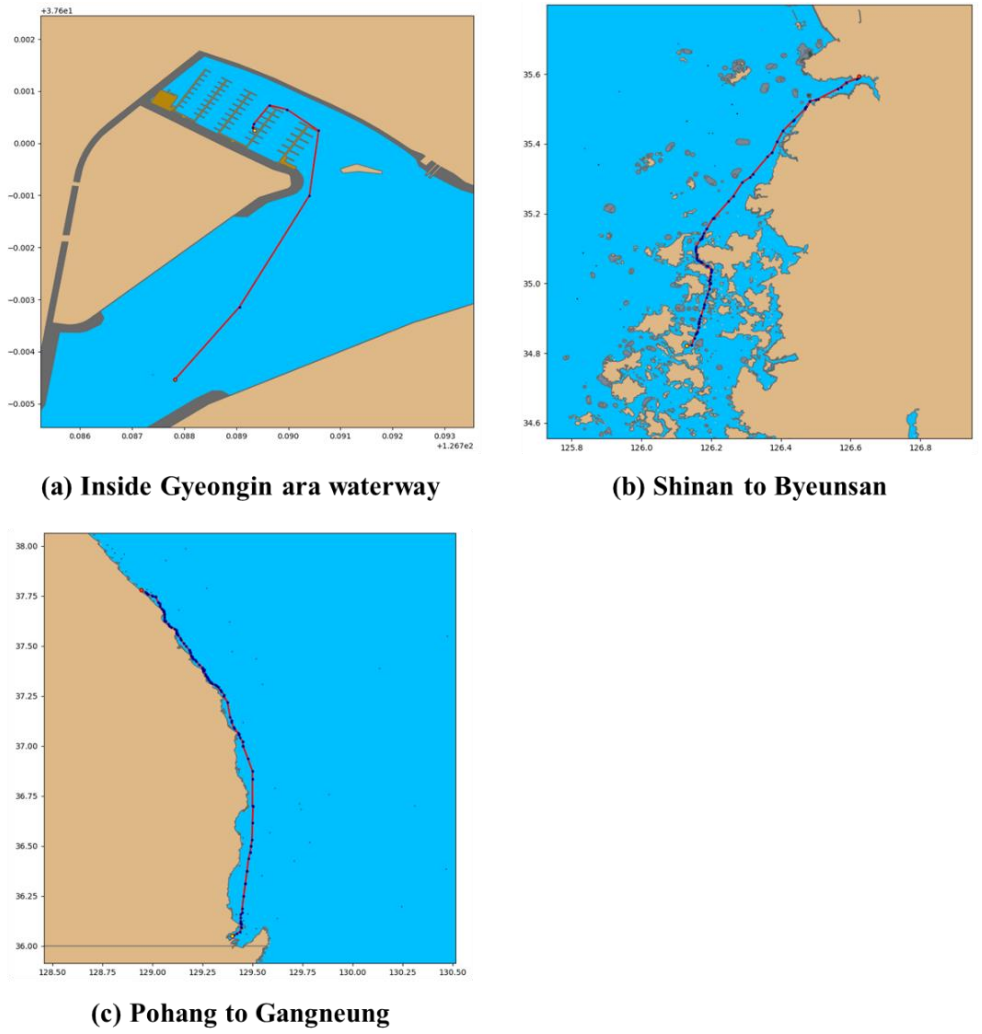
The route planning method proposed in this study was verified in various areas. With the proposed method, the memory required for route planning is kept between 100 and 200 MB, so it can be operated on a computer of a small ship.

Table 5 is a table that summarizes route planning time and distance by area. When only the route planning method outside the marina was used, the routes except for Busan to Incheon, which is an extremely long route, were planned within 3 seconds. Thus, it was verified that most routes were planned in a very short time. The route planning time of the proposed method was affected by the complexity and distance of the area where the route was planned because the complexity and distance are directly connected with the number of searched nodes in the quadtree chart.

Figure 39 (b) is the route from Shinan to Byeonsan, and (c) is the route from Pohang to Gangneung. The route from Pohang to Gangneung is twice as long as the route from Shinan to Byeonsan, but the environment from Shinan to Byeonsan seems more complicated, so the route planning time from Shinan to Byeonsan took a little longer. Figure 39 (a) is a route planned on the Gyeongin Ara Waterway, and it can be seen that the route accurately considers the shape of the pontoon. Also, it is attached to the obstacles because the route planning inside the marina is not applied. It can be seen that the routes by route planning inside the marina have a high route planning time in relation to the route distance, but safe routes that maintain the distance for the obstacles are planned, as shown in Figure 42 and Figure 43.

Table 5. Test results in various areas

Area	Time (s)	Distance (km)	With route planning process inside of marina
Inside Gyeongin ara waterway	0.83	0.76	X
Busan to Incheon	11.23	687.67	X
Around Jangbong	0.86	22.14	X
Around Mokpo	1.02	10.56	X
Pohang to Gangneung	2.52	209.89	X
Around Pyeungtaek	0.89	13.30	X
Shinan to Byeunsan	2.63	106.30	X
Taean to Muan	2.63	176.10	X
Lauderdale beach to Maule Lake	3.61	28.79	O
Around the marina at Fort Lauderdale	1.60	11.83	O
Around the marina at Intracoastal water way	1.30	6.91	O
Inside Fort Lauderdale	2.13	10.17	O



(a) Inside Gyeongin ara waterway

(b) Shinan to Byeunsan

(c) Pohang to Gangneung

Figure 39. Routes in coastal areas in Korea

Figure 40 compares the routes that do not pass through the area where the water depth is less than 2m and the routes that do not consider the water depth. The limit of the water depth is determined by the draft of the ship. The route of Figure 40 (a) around departure is more curved to avoid shallow areas than the route of Figure 40 (b) around departure. In addition, the route of Figure 40 (c) around the western coast is more curved to avoid

shallow areas than the route of Figure 40 (d) around the western coast.

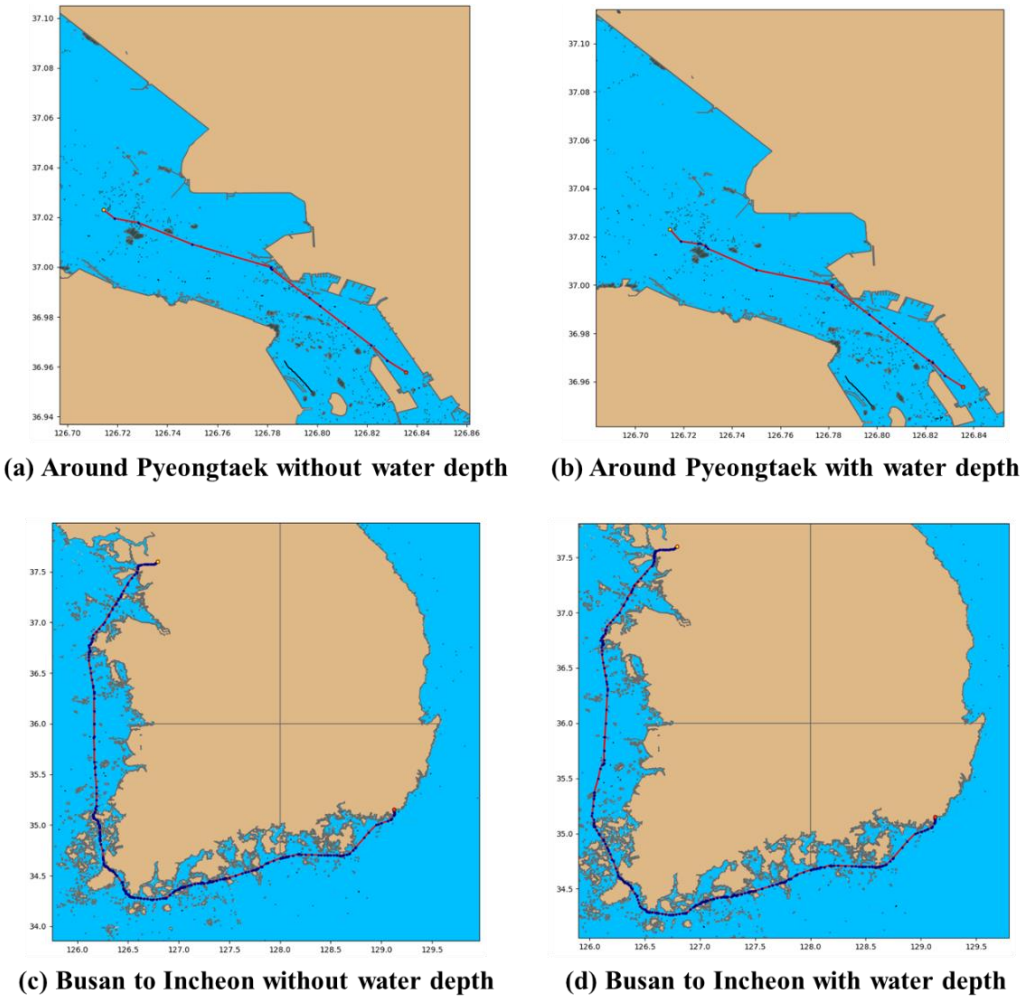
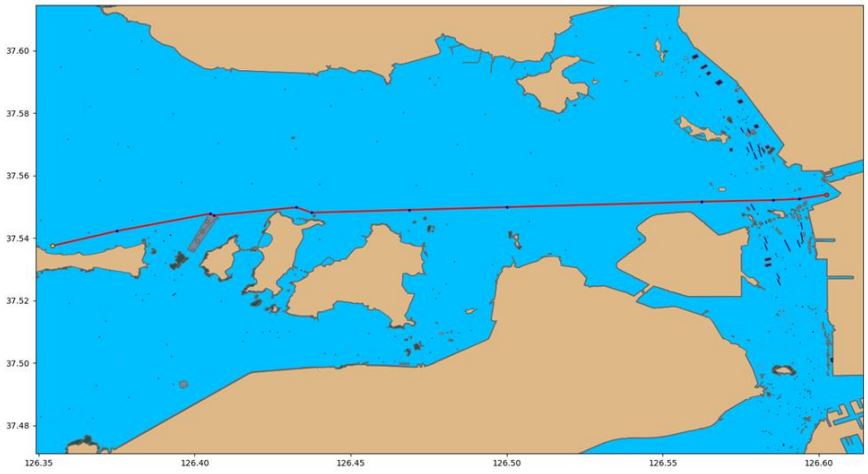
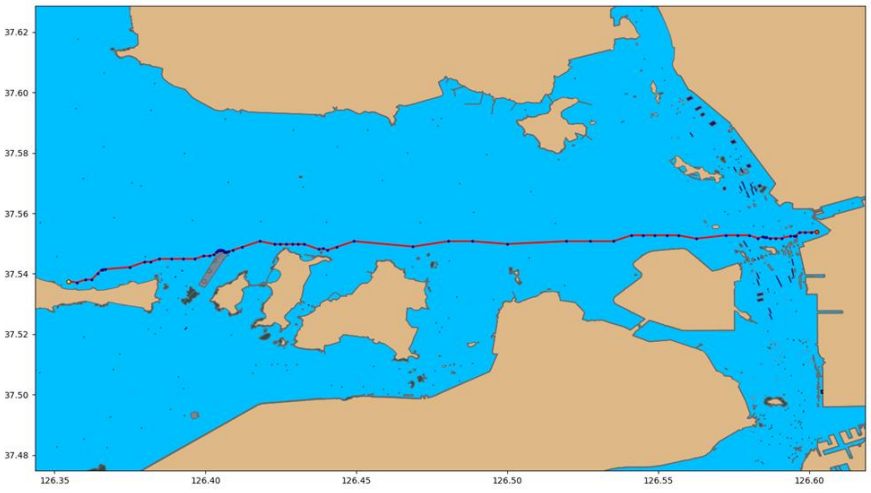


Figure 40. Comparison between routes with and without water depth



(a) Route with smoothing outside the marina

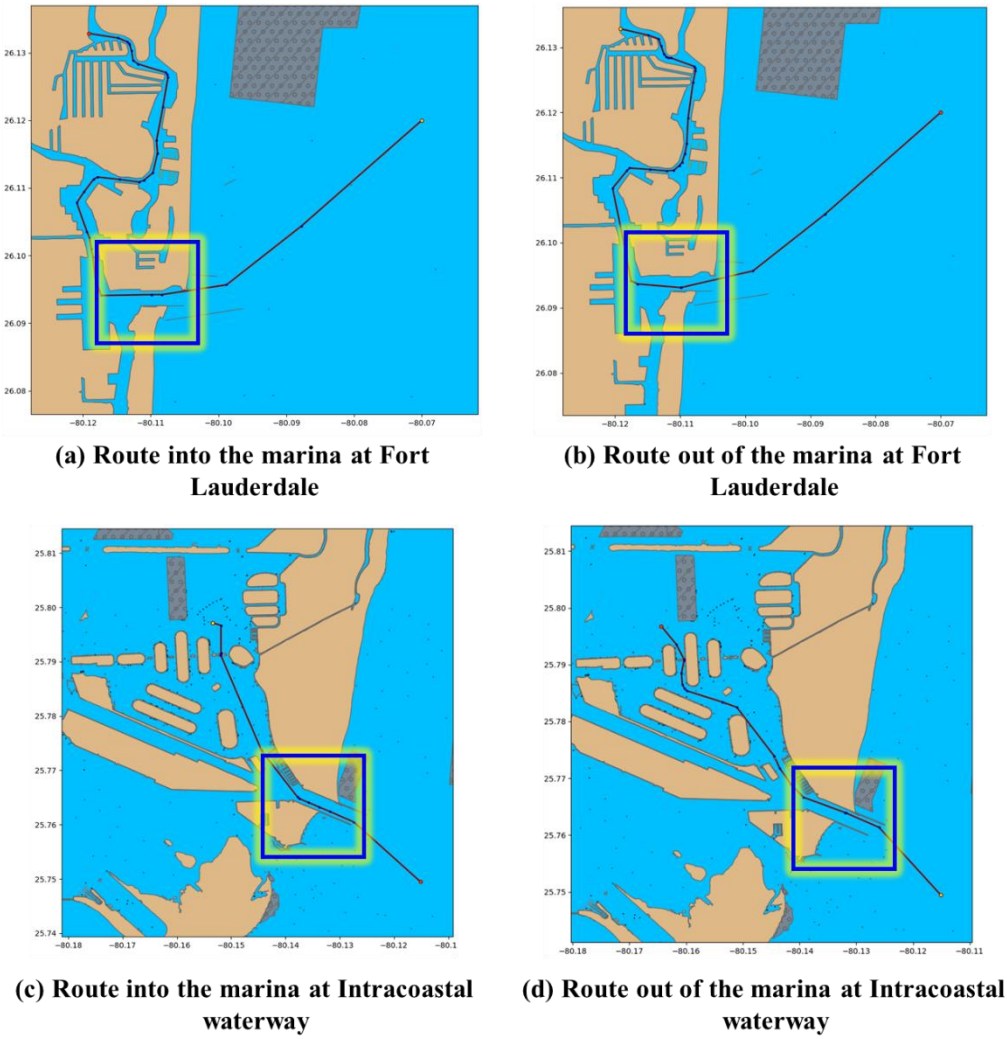


(b) Route without smoothing outside the marina

Figure 41. Comparison between routes with and without smoothing outside the marina

Figure 41 (a) shows a route planned by route planning outside the marina with smoothing. Figure 41 (b) shows a route planned by route planning outside the marina without smoothing. The number of route points in the route with smoothing is much smaller

than the route without smoothing. Also, many bends which is the side effect of grid-based route planning are removed in the route with smoothing.



(a) Route into the marina at Fort Lauderdale

(b) Route out of the marina at Fort Lauderdale

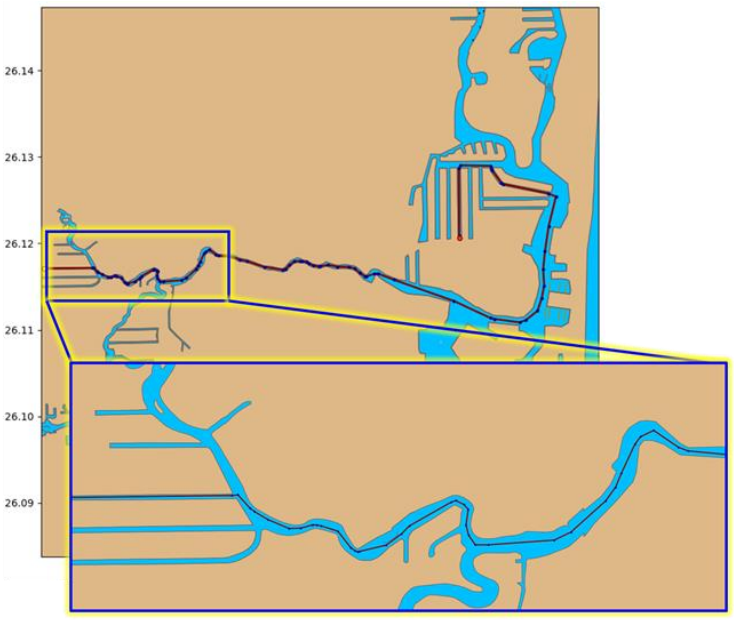
(c) Route into the marina at Intracoastal waterway

(d) Route out of the marina at Intracoastal waterway

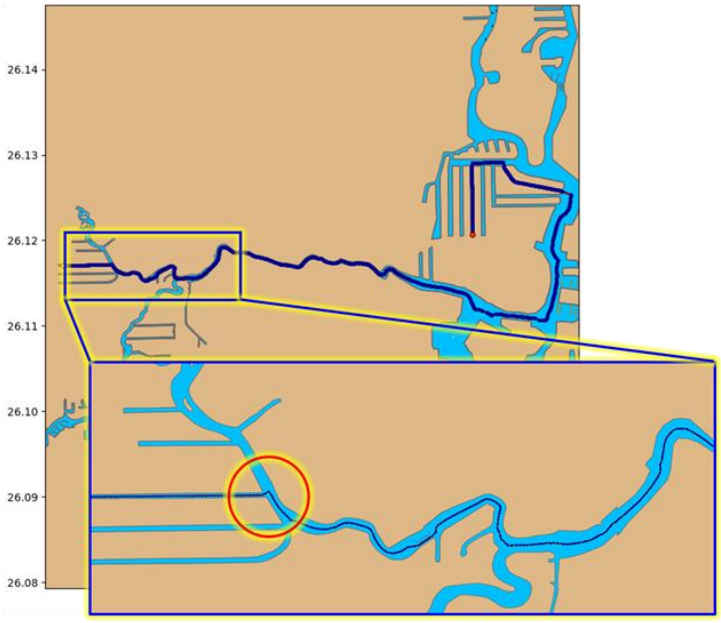
Figure 42. Routes around the marina

Figure 42 visualizes the route passing through the entrance of the marina. The routes outside and inside the marina are well planned by the proposed method, and they comply with the port-to-port rule following the position and the direction when passing through the marina entrance.

Figure 43 (a) shows a route planned by route planning inside the marina with smoothing. It shows that the proposed method can plan a route maintaining a certain distance from surrounding obstacles inside a narrow area. Figure 43 (b) shows a route planned by route planning inside the marina without smoothing. Comparing Figure 43 (a) and Figure 43 (b), the number of route points in Figure 43 (b) is large. Also, a detouring route, like inside the red circle in Figure 43 (b), does not exist in Figure 43 (a) because of the smoothing.



(a) Route with smoothing inside the marina



(b) Route without smoothing inside the marina

Figure 43. Comparison between routes with and without smoothing inside the marina

5. Conclusions and future works

5.1. Conclusions

In this study, a route planning method considering the characteristics of small ships and marinas was proposed for small ships. Even if a route is planned in complex coastal areas, the goal is to plan a route within a time that users can agree. The route planning charts for coastal areas were generated with a small file size using the quadtree representation. In addition, to comply with the rules inside the marina, the coast was separated into inside and outside the marina. The centerline of the marina was generated for the chart of the route planning inside the marina. The centerline of the marina is constructed using the Voronoi diagram of the marina.

Planning a route that covers a wide range on a single chart required much time and memory for route planning, so the HPA* was used. In this study, charts represented in a quadtree were divided to create two levels of hierarchy between charts. When the departure and destination are given, the route planning at the low-level chart is performed after the route planning at the high-level chart.

In the route planning at the high-level chart, a smoothing process was added to provide an optimal route compared to the original HPA*. The route planning at the low-level chart is divided into two methods planning the route outside the marina and planning the route inside the marina to create a safer route when passing through the marina. After the route planning at the low-level chart is completed, three smoothing processes are applied to the route. A smoothing process for the route outside the marina at the low-level chart, the smoothing process for the route inside the marina at the low-level chart, and the smoothing

process for combining the routes outside the marina are performed sequentially.

The route planning chart represented in the quadtree had a smaller file size than the chart represented in the regular grid. It is able to represent various objects with high resolution. In addition, it was verified that the optimal route was planned by applying the proposed route planning method in various environments. As a result of the comparison with four different route planning methods, the route planning time was shorter than other methods. The routes planned inside the marina maintain a specific distance from obstacles and follow the port-to-port rule when entering and exiting the marina.

5.2. Future works

In this study, only the obstacles to be avoided were reflected on the route planning chart when the small ships are sailing. However, the electronic navigational chart created by the S-57 not only provides the objects to avoid but also provides the objects for the safe route. Typically, a recommended route is an object that has been investigated for the safe navigation of a ship. It is required to develop the route planning method that follows the recommended route. In addition, a cardinal buoy is an object that informs the navigator of the direction to avoid a dangerous area.

In addition, a cardinal buoy is an object that informs the navigator of the direction to avoid a dangerous area. While planning a route, the area which is directed by the cardinal buoys should not be allowed to be searched. The traffic separation line or zone is the object that classifies the navigation direction of the ship. There are rules to follow when entering the traffic separation line or zone, so the route planning method that considers the traffic separation line or zone is additionally needed.

The developed route planning charts only have the static information specified in S-57. However, the navigability of the area is greatly influenced by the ocean weather. Therefore, it is necessary to reflect dynamic information that changes in real time, such as water depth according to tides, wind speed, and wave height, to the developed route planning chart. And developing a route planning method using dynamic information is required.

In addition, a safe route for berthing is required for individuals who are not experts. For safe berthing, it is important to maintain a distance from obstacles, but it is also important to consider the maneuverability of the ship. The maneuverability of a ship is affected by the sea conditions and seakeeping performance in the same time. Therefore, it is essential

to study the route planning method for berthing considering the safe distance from surrounding obstacles and the maneuverability of the ship with optimization methods different from this study, even if the route planning time increases

6. Reference

[1] K. Ministry of Oceans and Fisheries, Ship Safety Act, (2020) 4–5.

[2] S. Amarel, On Representations of Problems of Reasoning about Actions, Readings in Artificial Intelligence. (1981) 2–22. https://doi.org/10.1016/b978-0-934613-03-3.50006-4.

[3] H. Samet, The Quadtree and Related Hierarchical Data Structures, ACM Computing Surveys (CSUR). 16 (1984) 187–260. https://doi.org/10.1145/356924.356930.

[4] M.L. Cui, D.D. Harabor, A. Grastien, Compromise-free pathfinding on a navigation mesh, IJCAI International Joint Conference on Artificial Intelligence. 0 (2017) 496–502. https://doi.org/10.24963/ijcai.2017/70.

[5] S. Fortune, VORONOI DIAGRAMS and DELAUNAY TRIANGULATIONS, (1995) 225–265. https://doi.org/10.1142/9789812831699_0007.

[6] E. Welzl, Constructing the visibility graph for n-line segments in O(n2) time, Inf Process Lett. 20 (1985) 167–171. https://doi.org/10.1016/0020-0190(85)90044-4.

[7] S. Ghandi, E. Masehian, Review and taxonomies of assembly and disassembly path planning problems and approaches, CAD Computer Aided Design. 67–68 (2015) 58–86. https://doi.org/10.1016/j.cad.2015.05.001.

[8] E.W. Dijkstra, A note on two problems in connexion with graphs, J Educ Psychol. 19 (1959) 3. https://doi.org/10.1037/h0066879.

[9] B.R. Peter Hart, Nils Nilsson, A Formal Basis for the Heuristic Determination of Minimum Cost Paths, (1968). https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4082128 (accessed January 21, 2023).

[10] A. Botea, M. Martin, J. Schaeffer, C. Tg, Near Optimal Hierarchical Path-Finding, 2004.

[11] A. Nash, K. Daniel, S. Koenig, A. Feiner, Theta*: Any-angle path planning on grids, Proceedings of the National Conference on Artificial Intelligence. 2 (2007) 1177–1183.

[12] D. Harabor, A. Grastien, Online Graph Pruning for Pathfinding on Grid Maps, n.d.

[13] Steven M.LaValle, Rapidly-Exploring Random Trees: A New Tool for Path Planning, (1998).

[14] S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning, International Journal of Robotics Research. 30 (2011) 846–894. https://doi.org/10.1177/0278364911406761.

[15] J.D. Gammell, S.S. Srinivasa, T.D. Barfoot, Informed RRT∗: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic, IEEE International Conference on Intelligent Robots and Systems. (2014) 2997–3004. https://doi.org/10.1109/IROS.2014.6942976.

[16] and T.D.Barfoot. Gammell, Jonathan D., Siddhartha S. Srinivasa, Batch Informed tree, (2015).

[17] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, A. Cosar, A survey on new generation metaheuristic algorithms, Comput Ind Eng. 137 (2019). https://doi.org/10.1016/J.CIE.2019.106040.

[18] M.C. Tsou, H.C. Cheng, An Ant Colony Algorithm for efficient ship routing, Polish Maritime Research. 20 (2013) 28–38. https://doi.org/10.2478/pomr-2013-0032.

[19] R. Vettor, M. Tadros, M. Ventura, C. Guedes Soares, Route planning of a fishing vessel in coastal waters with fuel consumption restraint, in: Proceedings of 3rd International Conference on Maritime Technology and Engineering, MARTECH 2016, CRC Press/Balkema, 2016: pp. 167–174. https://doi.org/10.1201/b21890-24.

[20] S.M. Lee, M. Il Roh, K.S. Kim, H. Jung, J.J. Park, Method for a simultaneous determination

of the path and the speed for ship route planning problems, Ocean Engineering. 157 (2018) 301–312. https://doi.org/10.1016/j.oceaneng.2018.03.068.
[21] C. Liu, Q. Mao, X. Chu, S. Xie, An Improved A-Star Algorithm Considering Water Current, Traffic Separation and Berthing for Vessel Path Planning, Applied Sciences 2019, Vol. 9, Page 1057. 9 (2019) 1057. https://doi.org/10.3390/APP9061057.
[22] P. Han, X. Yang, P. Han, X. Yang, Big data-driven automatic generation of ship route planning in complex maritime environments, Acta Oceanologica Sinica, 2020, Vol. 39, Issue 8, Pages: 113-120. 39 (2020) 113–120. https://doi.org/10.1007/S13131-020-1638-5.
[23] W. Lee, G.H. Choi, T. wan Kim, Visibility graph-based path-planning algorithm with quadtree representation, Applied Ocean Research. 117 (2021) 102887. https://doi.org/10.1016/J.APOR.2021.102887.
[24] C. Liu, Q. Mao, X. Chu, S. Xie, An Improved A-star algorithm considering water current, traffic separation and berthing for vessel path planning, Applied Sciences (Switzerland). 9 (2019). https://doi.org/10.3390/app9061057.
[25] International Hydrographic Organization, S-57 Appendix A IHO Object Catalogue, (2000). https://www.iho.int/iho_pubs/standard/S-57Ed3.1/31ApAch1.pdf.
[26] H. Samet, R.E. Webber, Storing a Collection of Polygons Using Quadtrees, ACM Transactions on Graphics (TOG). 4 (1985) 182–222. https://doi.org/10.1145/282957.282966.
[27] A. Touir, ML-Quadtree: The Design of an Efficient Access Method for Spatial Database Systems, Journal of King Saud University - Computer and Information Sciences. 17 (2004) 45–64. https://doi.org/10.1016/s1319-1578(04)80003-x.
[28] S. Fortune, Voronoi diagrams and Delaunay triangulations, (1995).
[29] S. Younas, C.R. Figley, Development, Implementation and Validation of an Automatic Centerline Extraction Algorithm for Complex 3D Objects, J Med Biol Eng. 39 (2019) 184–204. https://doi.org/10.1007/s40846-018-0402-1.
[30] A. Agrawal, R. Verschueren, … S.D.-J. of C. and, undefined 2018, A rewriting system for convex optimization problems, Taylor & Francis. 5 (2018) 42–60. https://doi.org/10.1080/23307706.2017.1397554.
[31] J.H. and J. Snoeyink, Speeding Up the Douglas-Peucker Line-Simplification Algorithm | Computer Science at UBC, (n.d.). https://www.cs.ubc.ca/tr/1992/tr-92-07 (accessed January 3, 2023).

국문 초록

쿼드트리 해도를 활용한 연안 소형선의 계층적 경로 계획 방법

운송 수단의 경로 계획은 그 목적에 따라 다른 경로 계획용 지도의 자료구조와 경로 계획 알고리즘을 사용한다. 소형선의 경로 계획은 소형선의 작은 크기, 소형선이 주로 운항하는 해역 그리고 개인이 주로 운용한다는 점을 고려하여 적절한 경로 계획용 지도의 자료구조와 경로 계획 알고리즘을 선택하는 것이 필요하다.

일반적으로 전자해도는 International Hydrographic Organization (IHO)에서 규정한 전자해도 표준 형식에 따라 제작되며 전자해도의 객체들은 위경도 좌표로 표현되어 있어 경로 계획에 직접 사용하기엔 어려움이 있다. 본 연구에서는 지형을 일반적인 균일 격자해도에 비해 해도의 총 노드 수를 줄일 수 있는 쿼드트리를 사용하여 경로 계획용 해도를 구현하였다. 쿼드트리 해도는 연안과 같은 복잡하고 작은 장애물들이 산재한 지역을 소형선이 통항할 수 있는 수준의 정교함으로 표현한다.

또한, 소형선은 연안에서도 마리나 내부에서 항해하는 경우가 빈번함으로 마리나 내부 특성을 고려하기 위해 보로노이 다이어그램을 사용하여 제작한 중앙선을 다른 경로 계획용 해도로 선정하였다.

경로 계획 알고리즘으로는 Hierarchical Pathfinding A* (HPA*)를 사용하였다. HPA*는 해도를 계층적으로 분할하고 계층에 따라 순차적으로 경로 계획을 진행하여 경로 계획에 필요한 메모리 사용량과 경로 계획 시간을 크게 단축할 수 있다. 또한, 본 연구에선 경로의 품질을 높이기 위해 다단계 smoothing 방법을 적용하여 사용자 경험을 높이는 경로를 제공하였다.

본 연구는 쿼드트리와 보로노이 다이어그램, HPA*를 활용하여 다양한 해상 객체들을 고려한 빠른 경로 계획 프로그램을 제작하였고 다양한 연안 지역에서 검증하였다.

주요어: Route planning (경로 계획), Quadtree (쿼드트리), Hierarchical pathfinding A*, Voronoi diagram (보로노이 다이어그램), Small ships (소형선)

학번: 2021-26237