공학박사 학위논문

Coastal path planning algorithm using quadtree and visibility graph

쿼드 트리와 가시성 그래프를 이용한 연안 항로 계획 알고리즘

2021 년 8 월

서울대학교 대학원

조선해양공학과

이 원 희

Coastal path planning algorithm using quadtree and visibility graph

지도교수 김 태 완

이 논문을 공학박사 학위논문으로 제출함

2021 년 4 월

서울대학교 대학원

조선해양공학과

이 원 희

이원희의 공학박사 학위논문을 인준함

2021 년 7 월

위 원 장 **노 명 일**

부 위 원 장 **김 태 완**

위 원 **우 종 훈**

위 원 **오 민 재**

위 원 **박 진 형**

요약(국문초록)

국제해사기구(International Maritime Organization)가 e-Navigation 기술을 도입하면서 선박이 해상 센터와 데이터를 교환해야 하는 것이 표준이 되었다. 이를 위해 선박이 항해하는데 필요한 정보들이 정보 교환을 위해 전자화되고 있으며, 이러한 정보에는 항로 정보도 포함되어 있다. 따라서 선박은 항해 계획을 세운 뒤 해상 센터에 전송해야 하며, 해상 센터는 항해 계획에 문제가 없는지 확인하고 이를 다시 선박으로 보내야 한다. 대양을 항해하는 선박은 자동으로 항로를 계산하는 시스템이 구축되어 있기 때문에 항로의 데이터 교환이 용이하지만 연안을 항해하는 선박은 선장 및 항해사가 전문적 지식을 이용하여 수작업으로 항로를 결정하기 때문에 항로 데이터 교환이 어렵다. 항로의 전자화와 함께 수작업으로 항로를 계산하는 과정을 보완하기 위해 연안 내 항로를 자동으로 계산하는 시스템에 대한 요구가 증가하였으며, 이와 관련하여 활발한 연구가 진행되고 있다.

연안 선박은 여객선, 어선, 요트 등이 존재하며 다양한 목적을 갖고 있다. 선박은 각각의 목적을 달성하기 위해 목적지까지의 경로를 연안 내에서 계산해야 한다. 본 연구의 목적은 환경이 복잡한 연안 내에서 안전한

항로를 계산하는 것이다. 변화하는 해양환경을 고려한 경로 계획을 출발 전 혹은 실시간으로 운항 중인 선박에게 제공하여 선박의 안전을 확보해야 한다. 항로를 계획하기 위해 해안선, 수심, 기상, 조위 등의 정보가 필요하다. 이러한 정보를 기반으로 대양을 항해하는 선박의 항로 계획 연구가 많이 개발되었으나, 대부분의 연구는 연료 소모량과 항해 시간을 줄이는데 초점을 맞추고 있다. 연안에는 해안선이 복잡하고 장애물이 많기 때문에 기존의 연구를 그대로 적용하기 어렵다. 또한, 한국의 경우에는 조위 차가 큰 영역이 많기 때문에 항해 시 주변 환경이 자주 변하므로 빠른 시간 내에 항로를 계산해야 하는데 기존 연구는 이에 적합하지 않다.

본 논문에서는 쿼드 트리와 가시성 그래프의 장점을 통합하여 연안 내 복잡한 해양 환경을 위한 새로운 경로 계획 알고리즘을 제안한다. 기존의 균일 격자와 같은 비효율적인 환경 표현 방법을 개선하기 위해 쿼드 트리를 사용하였다. 쿼드 트리 위에서 그래프를 구성하고 쿼드 트리의 최단 경로를 계산한다. 앞서 계산한 항로의 웨이포인트를 기반으로 가시성 그래프를 생성하여 최적 경로를 계산하며, 이 때 Dijkstra 알고리즘을 사용한다. 제안한 알고리즘은 모든 각도에서 항로 계획을 수행할 수 있으며, 해안선으로부터 일정 간격을 유지하여 안전한 항로 계획을 수립한다. 또한, 선박의 출항 시간에 따른 기상 정보를 고려하여 연료 소모량을

계산하였으며, 연료 소모량이 최소가 되는 항로를 계획한다. 연료 소모량을 계산하기 위해 제동 동력을 계산해야 하며, 제동 동력은 선박의 저항과 속도의 곱으로 이루어진 유효 동력으로부터 계산할 수 있다. 선박의 저항은 정수 중 저항, 바람 부가저항, 파랑 부가저항으로 구성되며, 선박의 속력은 조류를 반영하여 이루어진다. 이 때 부가저항과 선박의 속력은 ISO 15016 을 기반으로 하여 계산된다. 이러한 과정을 수행하여 계산한 최소 연료 소모량 항로와 최단 거리 항로를 비교 및 평가하였다. 성능 평가를 위해 본 연구에서 제안한 알고리즘을 대한민국 서해안과 남해안 영역에 대해 적용하였다. 기상 정보의 영향으로 최소 연료 소모량 항로가 최단 거리 항로와 형상이 다른 것을 확인했다. 전반적으로 제안한 알고리즘이 기존 연구 방법에 비해 빠른 시간 안에 최적 항로를 생성할 수 있음을 확인하였다.

주요어: Coastal path planning, Polygon Map Random (PMR) Quadtree, Visibility Graph, Fuel Oil Consumption, Complex Marine Environment

학 번 : 2015-21172

Contents

List of Figure

List of Table

1 Introduction

1.1 Background

국제해사기구(International Maritime Organization)가 e-Navigation을 도입하면서 선박이 해상 센터와 데이터를 교환해야 하는 것이 국제 표준이 되었다. 이를 위해 선박이 항해하는데 필요한 정보들이 정보 교환을 위해 전자화되고 있으며, 이러한 정보에는 항로 정보도 포함되어 있다. IMO의 e-Navigation의 정의는 다음과 같다.

“The harmonized collection, integration, exchange, presentation and analysis of marine information on board and ashore by electronic means to enhance berth to berth navigation and related services for safety and security at sea and protection of the marine environment.” (IMO, 2021)

e-Navigation의 정의는 해상 안전, 보안, 해양 환경 보호를 위해 선상과 해양 센터 사이의 정보를 전자적 수단을 이용하여 조화롭게 수집, 통합, 교환, 표시 및 분석하는 것이다. e-Navigation은 특정 기술에 한정되어 있지 않고 포괄적인 의미를 지니고 있다. 해양 환경 위에서 필요한 모든 정보들을 전자화하고 선박과 해상 센터 사이에서 정보를 교환하는 것이 주 목적이 되며, 현재 많은 기술들이 e-Navigation 국제 표준화에 맞추기 위해 개발되고 있다.

국내에서는 e-Navigation에 맞추기 위해 전자해도 정보 최신화, 해양안전정보 전자화, 선내시스템 정보 전자화, 선박 항로 전자화 등의 노력이 이

루어지고 있다.

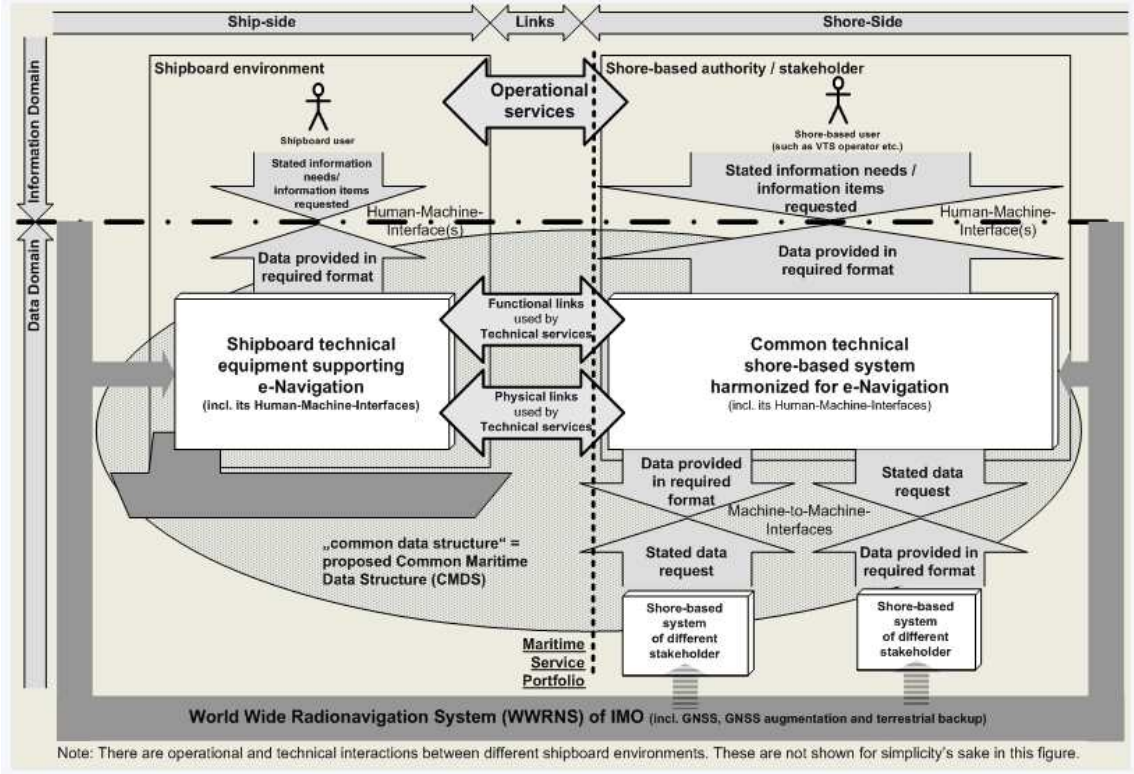

Figure 1-1 Definition of e-Navigation (IMO, 2021)

e-Navigation 에 따르면 선박은 항해 계획을 세운 뒤 해상 센터에 전송해야 하며, 해상 센터는 항해 계획에 문제가 없는지 확인하고 이를 다시 선박으로 보내야 한다. 대양을 항해하는 선박일 경우 weather routing 등의 기법을 이용하여 항로를 자동으로 계산하기 때문에 항로 정보를 교환할 수 있다. 하지만, 연안을 항해하는 선박의 경우 선장 및 항해사가 전문적 지식을 이용하여 수작업으로 항로를 결정하기 때문에 항로의 교환이 어렵다. 수작업으로 항로를 계산하는 과정을 보완하기 위해

연안 내 항로를 자동으로 계산하는 시스템에 대한 요구가 증가하였으며, 이와 관련한 활발한 연구가 진행되고 있다.

연안 선박의 종류는 여객선, 어선, 요트 등이 존재하며, 각 선박은 다양한 목적을 갖고 있다. 선박은 각각의 목적을 달성하기 위해 연안 내 목적지까지의 항로를 계산해야 한다. 연안 내 항로를 계획하기 위해 지형, 수심, 기상, 조위 등의 정보가 필요하다. 대양을 항해하는 선박의 경로 계획은 기상 정보를 이용하여 연료 소모량, 항해 시간을 줄이기 위해 개발되었으며, 대부분의 연구는 연료 소모량과 항해 시간을 줄이는데 초점을 맞추고 있다. 연안에는 해안선이 복잡하고 장애물이 많기 때문에 기존의 연구를 그대로 적용하기 어렵다. 또한, 어선과 요트와 같이 항해 시간이 짧은 선박에 대해서는 빠른 시간 내에 항로를 계산해야 할 필요가 있는데 기존의 대양 항해 연구는 계산 시간이 오래 걸리므로 이에 적합하지 않다.

향후 e-Navigation 표준에 맞추기 위해 연안 내 선박의 항로 계획 자동화가 활발히 이뤄질 것으로 예상된다. 하지만, 현재 연안 내 선박의 항로 계획은 선장 및 항해사의 경험을 바탕으로 이루어지고 있다. 이를 보완하기 위해 자동으로 연안 내 선박의 항로 계획을 수립할 수 있는 시스템 개발이 필요하다.

1.2 Objective and method

본 논문은 빠른 시간 내에 자동으로 연안 선박의 항로 계획을 수립하는

것을 목적으로 한다. 선박의 항로를 결정할 때 다음의 요소인 1) 지형, 2) 수심, 3) 조위, 4) 기상 정보를 고려해야 한다. 연안을 항해하는 선박은 대양을 항해하는 선박의 항로 계획 수립(Weather routing)과는 다르게 항로 계획을 수립할 때 Figure 1-2와 같이 해안선과 섬의 간섭을 확인하는 것이 중요하다. 대양에는 해안선이나 섬과 같은 장애물이 전체 영역의 일부분을 차지하지만, 연안에서는 절반 혹은 그 이상의 영역이 장애물일 수 있다. 따라서 장애물이 많은 복잡한 환경을 최적 항로가 나올 수 있도록 맵을 구성하는 것이 중요하다. 최적 항로의 정의는 여러가지가 있을 수 있다. 본 논문에서는 최단거리 항로와 최소 에너지를 갖는 항로를 최적 항로라 정의하였다. 이와 같은 최적 항로를 수립하기 위해 본 논문에서는 다음의 과정을 통한 알고리즘을 제안한다.

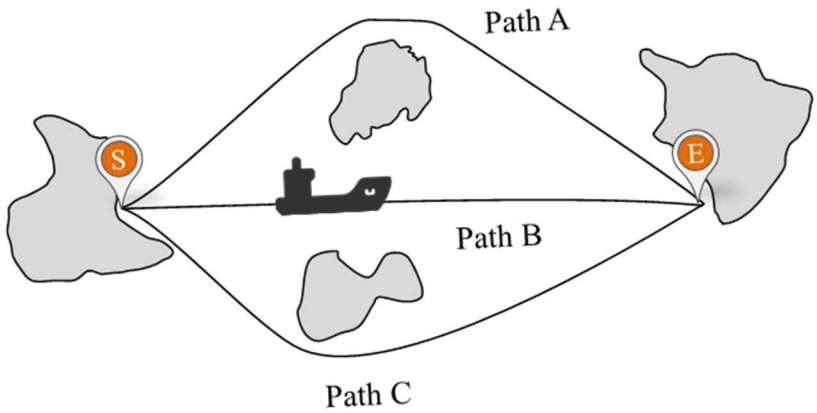

Figure 1-2 Path planning in coastal environment

1) 쿼드 트리를 이용하여 탐색 영역(Configuration space)을 효율적으로 표현한 후 불필요한 영역을 제거한다.

2) 가시성 그래프를 이용하여 남은 영역에서 최적 항로를 계산한다.

3) 기상 정보를 이용하여 선박의 연료 소모량이 최소가 되는 항로를 계산한다.

기존 항로 관련 연구에서는 섬과 같은 장애물이 적은 대양에서의 항로 계획을 수립하였다. 이는 장애물 간섭 제약조건이 적은 환경에서 항로를 수립했기 때문에 연안과 같이 장애물이 많은 복잡한 환경에는 적용하기 어렵다. 본 논문에서는 이러한 기존연구의 한계를 보완하여 해안선이 복잡하고 섬이 많은 영역에 대해 최적 항로를 계산하였다. 또한, 선박의 항로를 수립하는데 필요한 수심, 조위, 기상 정보를 반영하여 효율적인 항로를 계산하였다.

본 논문의 알고리즘의 전체적인 구성과 입력 정보 및 출력 정보는 Figure 1-3과 같다.

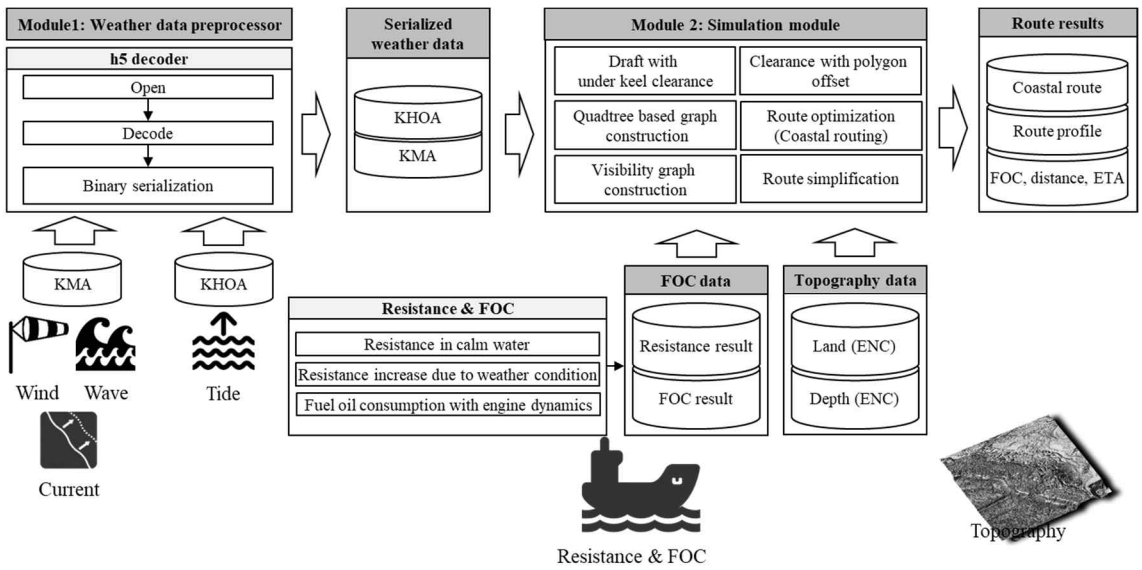

Figure 1-3 Input and output of proposed algorithm

알고리즘의 입력 정보는 환경 정보, 선박 정보, 기상 예측 정보이다. 환

경 정보는 전자해도의 해안선과 섬, 수심 정보를 포함한다. 선박 정보는 선박의 제원과 출발, 도착지, 출발 시간 정보를 포함한다. 기상 정보는 파고, 바람, 해류, 조위 정보를 포함한다. 본 논문에서 파고, 바람, 해류 정보는 기상청 정보에서 가져왔으며, 조위 정보는 해양조사원 정보를 사용하였다.

알고리즘에서는 연안 내 선박의 항로를 수립하는 방법을 제안한다. 빠른 계산 시간 내에 항로를 계산하기 위해서 쿼드 트리와 가시성 그래프의 장점을 통합하여 연안 내 복잡한 해양 환경을 위한 새로운 경로 계획 알고리즘을 제안한다. 복잡한 해양 환경을 표현하기 위해 쿼드 트리를 사용하였으며, 출발지와 도착지에 따라 쿼드 트리 내 필요한 영역을 분할하여 사용하였다. 또한, 항로의 최적화를 위해 가시성 그래프를 이용하였으며, 가시성 그래프의 계산 시간을 줄이기 위해 쿼드 트리 구조를 이용하였다. 수심과 조위 정보를 이용하여 선박이 안전하게 지나갈 수 있는 항로를 계산하였으며, 기상 정보를 이용하여 선박의 연료 소모량을 계산하였다. 연료 소모량을 계산하기 위해 제동 동력을 계산해야 하며, 제동 동력은 저항과 속력의 곱인 유효 동력으로부터 계산될 수 있다. 선박의 저항을 계산하기 위해 정수 중 저항, 바람 부가저항, 파랑 부가저항을 계산하였으며 이 때 ISO 15016을 사용하였다. 이러한 과정을 수행하여 연료 소모량이 최소가 되는 효율적인 항로를 제공할 수 있도록 하였다.

알고리즘의 출력 정보는 선박의 항로 계획 결과이다. 제안한 알고리즘을 검증하기 위해 기존의 가시성 그래프 알고리즘과 비교를 수행했으며, 알고리즘의 성능을 평가하기 위해 확률 기반 경로 탐색 알고리즘과 비교하였다.

그리고 실제로 적용이 가능한지 확인하기 위해 대한민국의 서해안과 남해안 영역에 대해 시뮬레이션을 수행하였다.

1.3 Contributions of this paper

본 논문의 공헌은 다음과 같다.

(1) 연안의 복잡한 해양 환경에 대해 기존의 항로 계획 수립 방법 대비 항로 계획을 자동으로 빠른 시간 내에 수립한다.

(2) 해안선과 섬의 지형 정보의 기하학적인 손실 없이 모든 방향으로 항로를 계획한다.

(3) 연안 선박의 항로 계획 수립에 선박의 제원을 반영하여 효율적이면서 안전한 항로를 탐색한다.

(4) 실시간으로 수심, 기상 정보를 고려하여 항로 계획 수립에 큰 이점을 제공한다.

(5) 기상 정보에 따라 연료 소모량이 최소가 되는 항로를 제공함으로써 효율적인 항로를 계획한다.

1.4 Outline

1장에서는 본 논문의 연구 배경, 목적, 방법, 그리고 기여에 대해 서술하였다. 2장에서는 경로 계획 방법과 관련하여 기존에 수행된 연구를 정리하였으며, 3장에서는 연안에서 항로를 계획하는 알고리즘과 관련된 연구를 정리하였다. 4장에서는 본 논문의 연구에 적용하기 위해 사용된 해안선 정보, 수심 정보, 조위 정보, 기상 정보의 획득 및 해안선 확장 알고리즘을 제시한다. 5장과 6장에서는 복잡한 해양 환경 내에서 항로를 빠르게 계산하기 위해 쿼드 트리와 가시성 그래프를 이용하여 최단 거리를 기반으로 항로를 계산하는 방법을 소개한다. 또한, 수심 정보, 조위 정보, 기상 정보 등의 사용 방법에 대해 설명한다. 7장에서는 ISO 15016에 기반하여 기상 조건에 따른 부가 저항과 선박의 속력을 계산하고 연료 소모량을 계산하는 방법에 대해 설명한다. 8장에서는 연안 항로 계획 알고리즘의 전체 과정에 대해 설명하였으며, Dijkstra 알고리즘으로 최적 항로를 찾는 과정을 설명한다. 9장에서는 제안한 방법을 이용하여 해안선과 섬을 고려한 항로와 수심과 조위 정보를 포함하여 계산한 항로 결과를 제시한다. 또한, 기상 정보를 이용하여 연료 소모량이 최소가 되는 항로와 최단 거리 항로를 비교하였다. 10장은 본 논문의 결론과 향후 연구계획을 제시한다.

2 Theoretical background

일반적으로 경로 계획 방법은 그래프 방법(Combinatorial method), 샘플링 방법(Sampling based method) 및 생물 구조 기반(Bio-inspired method) 방법의 세 가지 방법으로 분류된다. 그래프 기반 방법은 환경을 모델링하여 탐색 공간(c-space)을 그래프로 변환하고 그래프 위에서 경로를 탐색하는 방법이다. 그래프 기반 방법은 탐색 공간 표현 기법과 그래프 탐색 알고리즘으로 구성된다. 탐색 공간 표현 기법에는 균일 격자, 비 균일 격자, 보로노이 다이어그램, 가시성 그래프, 포텐셜 필드 등의 방법이 있다. 그래프 탐색 알고리즘은 깊이 탐색, 너비 탐색, Dijkstra 알고리즘, A* 알고리즘, A* 변형 방법 등이 있다.

그래프 기반 방법 외에도 샘플링 기반 방법이 있다. 샘플링 기반 방법에는 RRT(Rapidly exploring random tree)와 확률 기반 로드맵(Probabilistic Roadmap)가 대표적인 방법이다. 이 방법은 시작점에서 도착점을 정의하고, 시작점에서 트리를 랜덤으로 구성하여 그래프를 생성하는 방법이다. 확률 기반 로드맵 방법은 탐색 공간에서 임의로 점들을 선택하고 환경 내 장애물과 충돌하지 않는다면 이를 연결하여 그래프를 생성하는 방법이다. 출발지와 도착지를 결정한 후 그래프 탐색 알고리즘을 이용하여 경로를 결정한다.

마지막으로 생물학적 구조를 기반으로 경로를 찾는 방법(Bio- inspired method)이 있다. 이 방법에는 유전 알고리즘(Genetic algorithm), 입자 군

집 최적화(Particle swarm optimization), 개미 군집 최적화(Ant colony optimization), 시뮬레이션 어닐링(Simulated annealing), 인공 신경망(Neural network) 등의 방법이 있다.

Figure 2-1 The shortcoming of probabilistic methods

샘플링 기반 방법과 생물 구조 기반 방법들은 확률에 기반하여 경로를 계산하기 때문에 격자나 메시를 이용하여 그래프를 구성하는 방법보다 해의 품질이 좋다. 하지만, 해의 품질을 높이는 만큼 계산 시간이 더 오래 걸린다. 확률에 기반하여 경로를 결정할 때 육지와의 간섭을 판단하는데 많은 연산이 수행된다. 본 논문의 적용 영역은 해안선이 복잡하고 섬이 많기 때문에 이러한 연산의 수행이 기하급수적으로 증가하게 된다. Figure 2-1은 확률 기반 알고리즘의 예시이다. 확률적으로 경로를 생성하면, 생성된 경로가 육지와 겹치는지(Land interference) 확인해야 한다. 경로와 장애물을 하나씩 비교해야 하므로 장애물이 많을수록 계산 시간이 더 증가한다. 최악의 경우, Figure 2-1과 같이 해가 생성되지 않을 수 있다. 경로 생성 범위

를 크게 하면 해를 찾을 수 있으나, 이 경우에도 시간 복잡도가 증가하게 된다.

이러한 이유로 샘플링 기반 방법과 생물 구조 기반 방법은 장애물이 많은 영역에 적용하기가 어렵다. 따라서, 본 논문에서는 환경을 모델링하고 그래프를 구성하여 경로를 계산한다. 2.1절에서는 환경을 모델링하는 방법에 대해 설명한다. 2.2절에서는 경로 탐색을 위해 사용되는 그래프 기반 방법, 샘플링 기반 방법, 생물 구조 기반 방법에 대해 설명한다.

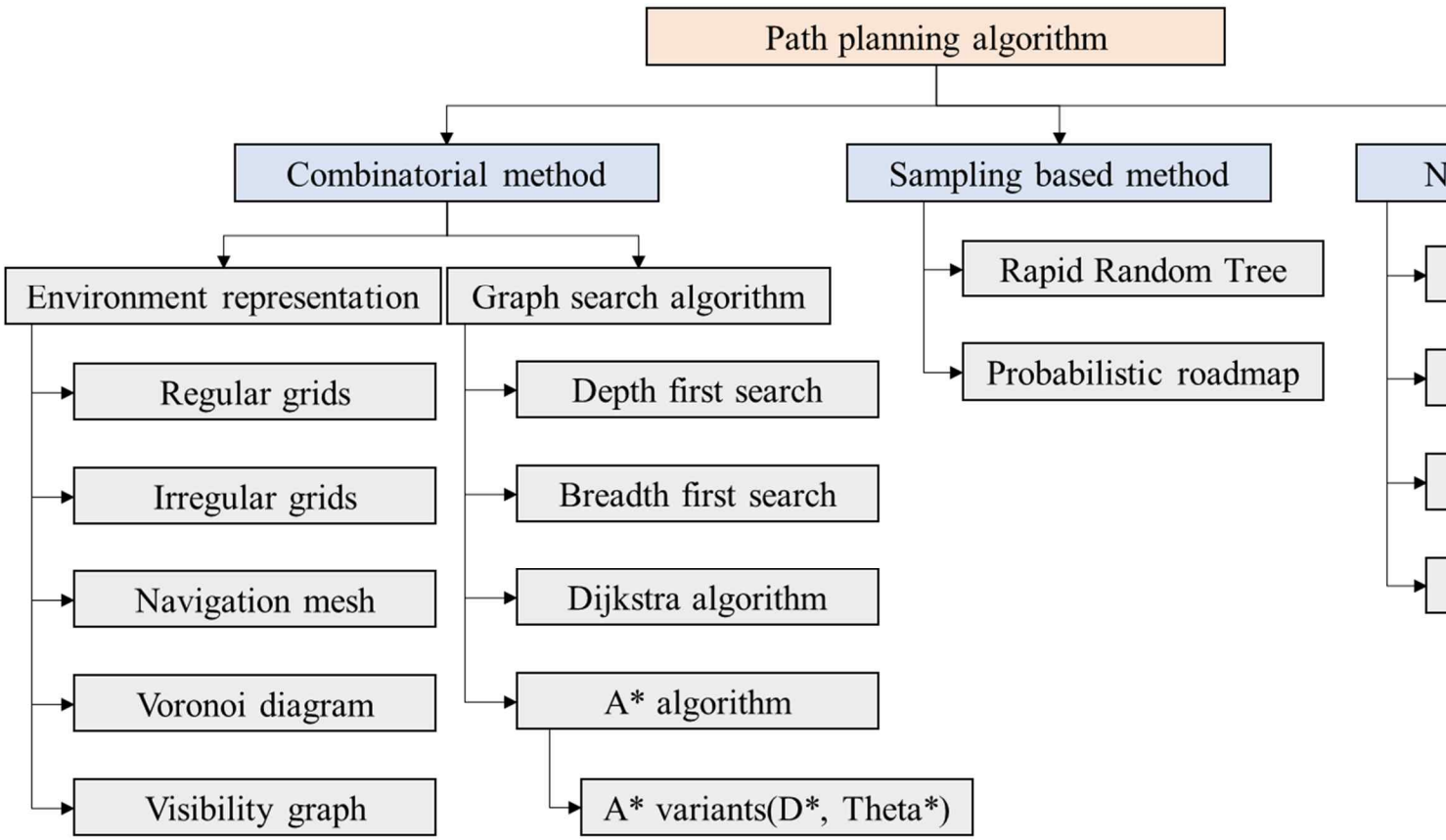

Figure 2-2 Classification of path planning

2.1 Environment modelling

앞에서 언급한 바와 같이 환경 모델링은 경로 계획 알고리즘을 실행하기 위해 필수적으로 진행해야 하는 단계이다. 본 절에서는 경로 탐색 알고리즘에서 사용되는 환경 모델링 방법들을 소개한다. 셀의 사용 여부에 따라 환경 모델링 방법을 크게 두 가지로 분류할 수 있다. 첫 번째 방법은 일정 영역을 셀로 분해하는 방법(Cell decomposition)으로 균일 격자, 비 균일 격자 등의 방법이다. 이 분류 방법에서는 셀이 장애물의 형상(topology)을 표현하기 위해 사용된다. 두 번째 방법은 모델링해야 하는 환경 내 장애물들의 점을 이용하여 가시성 그래프, 메쉬, 다이어그램으로 미리 계산하는 방법이다. 각 방법을 아래에서 자세히 정리하였다.

2.1.1 Regular grids

균일 격자(Regular grids)는 로봇 공학 및 비디오 게임에서 환경 모델링에 가장 많이 사용되는 방법이다. 실제로 균일 격자는 구현의 용이성 및 격자 업데이트의 용이성 등의 장점이 있다. 균일 격자는 격자를 생성하는 영역 내의 장애물의 수와 관계없이 항상 동일한 수의 노드(node)와 간선(edge)를 갖기 때문이다. 균일 격자 내 각각의 노드는 지형의 내부 혹은 외부에 따라 통행이 가능한 영역 혹은 불가능한 영역으로 구분된다. 아래 Figure 2-3은 격자 생성 형태(primitive)인 정삼각형, 정사각형, 정육각형으로 구성된 격자 예시이다. (Add reference)

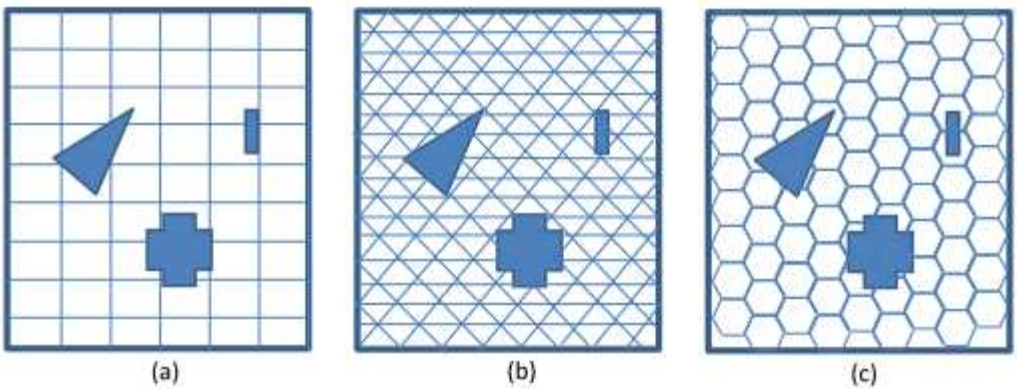

Figure 2-3 Regular grids (a) Square (b) Triangle (c) Hexagon

(Souissi. et al., 2013)

균일 격자는 생성하기 쉽고 노드 수가 일정하기 때문에 다른 방법에 비해 사용하기 용이하지만 다음의 단점을 지니고 있다. 첫 번째로, 셀의 모양과 크기가 지형의 특징을 표현하는데 충분하지 않을 경우, 환경 모델링이 부정확해질 수 있다. 예를 들어, 사각형으로 균일 격자를 구성하면, 원형 장애물을 정확하게 표현하기 어렵다. 두 번째로, 균일 격자를 생성하기 위해 많은 메모리 소모량과 계산 시간이 필요하다. 균일 격자로 복잡한 형상을 표현하려면 격자의 크기를 세분화해야 하기 때문에 필요한 셀의 수가 많아지므로 메모리 소모량이 크다. 복잡한 형상이 많아 셀의 수가 많을 경우에 경로 탐색 알고리즘을 실행하면 메모리 소모량은 커지므로 시스템의 성능이 낮아질 수 있다 (Souissi et al., 2013).

2.1.2 Irregular grids

균일 격자의 단점을 보완하기 위해 쿼드 트리를 사용하는 비 균일 격자(Irregular grids) 방법이 소개되었다. 쿼드 트리를 사용하는 방법은 셀의 크기가 동일하지 않기 때문에 격자 배치가 불규칙하다. 가장 많이 사용되는 쿼드 트리는 이미지 처리, 지리적 정보 시스템(Geographic Information System), 2차원 충돌 감지 등의 다양한 영역에서 사용되었다. Figure 2-4에서 볼 수 있듯 쿼드 트리 표현 방법은 타겟 영역을 4개의 사각형으로 세분화하고 장애물이 있는 각 사각형에 대해 해당 과정을 반복 수행하는 방법이다.

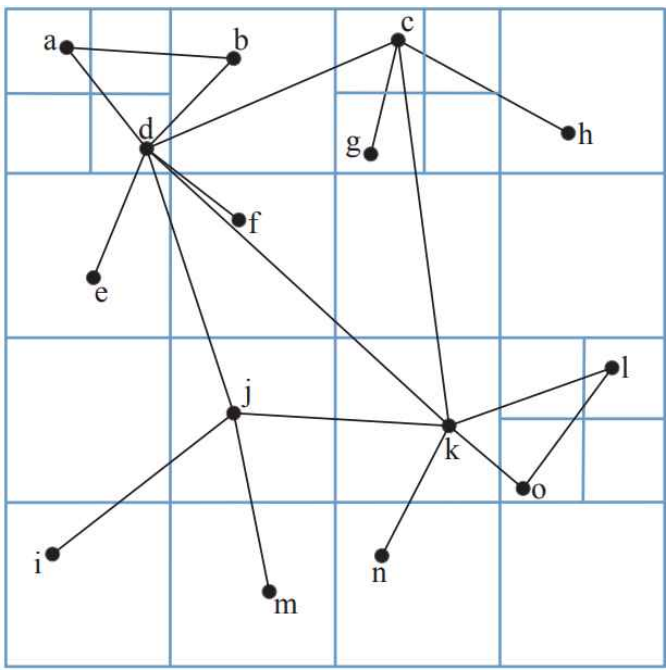

Figure 2-4 Quadtree partitioning and network example

(Huang et al., 2014)

따라서 쿼드 트리를 사용하면 장애물이 없는 영역에는 크기가 큰 셀들이

적게 생성되기 때문에 노드 수가 적으며, 장애물이 많은 영역에는 크기가 작은 셀들이 많이 생성되므로 노드 수가 많다. 따라서 쿼드 트리를 사용하는 비 균일 격자 방법은 균일 격자 방법에 비해 메모리 소비를 크게 줄일 수 있다. 쿼드 트리를 기반으로 그래프를 구성하면 그래프의 크기가 작이지기 때문에 경로 탐색 알고리즘에서 그래프를 탐색하는 시간이 감소한다. 쿼드 트리의 주요 단점은 노드가 불규칙하게 배치되기 때문에 쿼드 트리를 기반으로 경로를 계산하면 경로의 품질이 낮아질 수 있다. 또 다른 단점은 타겟 영역 내에 장애물이 너무 많으면 쿼드 트리의 분해 과정이 계속 수행되기 때문에 균일 격자와 비슷한 성능이 나오게 된다.

2.1.3 Navigation mesh

탐색 메시(Navigation mesh)는 타겟 영역의 이동 가능한 영역을 표현하기 위해 사용된다. 탐색 메시는 주어진 환경에서 이동 가능한 영역을 나타낼 수 있는 모든 유형의 다각형 메시를 나타내는데 광범위하게 사용되었다. 볼록한 다각형(convex polygon)인 경우, 탐색 메시를 구성할 수 있다. 가장 널리 쓰이는 방법은 삼각형을 이용하는 방법으로, 탐색 메시는 주어진 환경을 삼각화(Triangulation)하여 생성할 수 있다(Kallmann, 2010). 삼각화 과정에서 모든 삼각형들의 내각이 최대한 균등하게 설정(angle-optimal)되도록 들로네 삼각분할(Delaunay Triangulation)을 수행한다. 이 과정에서 삼각형 내부 최소 각도가 최대가 되도록 설정하여 기하학적으로

얇은 삼각형(skinny triangle)이 나오지 않도록 설정한다. Figure 2-5는 주어진 환경에 대해 탐색 메시를 생성하는 예시이다.

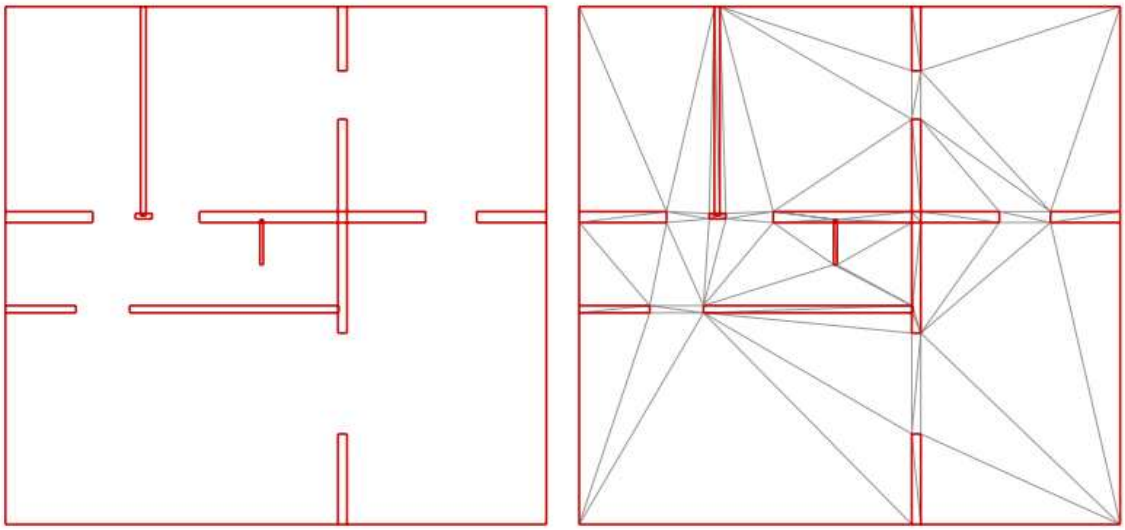

Figure 2-5 The example of navigation mesh in environment

(Kallmann, 2010)

탐색 메시는 장애물이 많은 영역을 표현하는데 탐색 메시를 구성하는 노드 수가 장애물의 점 수에 따라 선형으로 증가하기 때문에 적은 수의 노드와 간선으로 구성할 수 있다. 탐색 메시는 장애물의 점을 이용하여 생성하기 때문에 영역 내 장애물 형상 정보가 손실되지 않는다. 하지만, 탐색 메시는 장애물에 기반하여 생성되기 때문에 이를 기반으로 경로를 찾으면 최적 경로가 아닌 경로가 나올 수 있다.

2.1.4 Voronoi diagram

보로노이 다이어그램(Voronoi diagram)은 주어진 환경에서 장애물과 동일한 거리의 선을 연결하여 생성된다. 보로노이 다이어그램은 장애물과 일정 거리를 두고 생성되기 때문에 안전한 경로를 탐색할 수 있다. 보로노이 다이어그램은 다각형의 점이 n개 존재할 때 $O(nbg\ n)$ 시간 내에 생성될 수 있다. 보로노이 다이어그램을 이용하여 경로의 웨이 포인트가 최소가 되도록 최적 경로를 찾는 연구가 많이 진행되었다(Bhattacharya and Gavrilova, 2008). 하지만, 보로노이 다이어그램 위에서 계산한 경로는 최적 경로가 아닐 가능성이 높다.

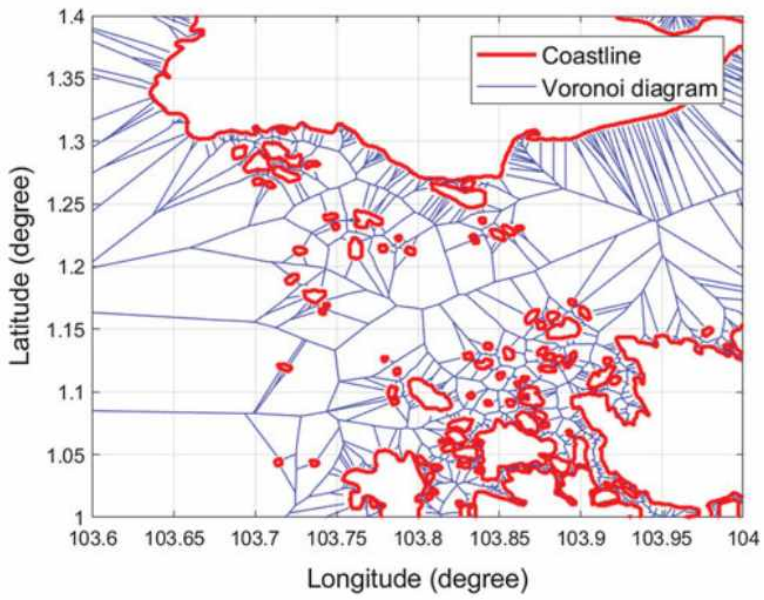

Figure 2-6 The example of Voronoi diagram-based partition of an obstacle map (Niu et al., 2019)

이러한 단점을 보완하기 위해 보로노이 다이어그램에서 계산한 경로를 가공하는 연구들이 많이 제안되었다. 코너 반복 제거 방법(Iterative

corner refinement), 가시성 그래프(Visibility graph) 등의 방법을 이용하여 보로노이 다이어그램을 보완하는 방법이 제안되었다 (Magid et al., 2017; Niu et al., 2019).

2.1.5 Visibility graph

가시성 그래프(Visibility graph)는 시작 노드, 도착 노드를 포함한 장애물의 모든 점을 연결하여 그래프를 생성한다. 가시성 그래프는 가능한 모든 경로를 생성하기 때문에 최적 경로를 찾을 가능성이 매우 높다. 그리고 최적 경로는 직선이거나 장애물과 밀접하게 계산된다. Dijkstra 알고리즘이나 A* 알고리즘을 이용하면 가시성 그래프 위에서 최적 경로를 찾을 수 있다. 이러한 이유로 경로 탐색 문제에서 가시성 그래프는 환경 모델링에서 자주 사용되었다.

장애물의 모든 점에 대해 연결하고 그 간선이 통항 가능한지 판단하기 위해 연결한 모든 간선과 장애물이 충돌하는지 확인해야 한다. 모든 점을 연결하는데 필요한 계산 시간이 $O(n^2)$, 간선이 장애물과 충돌하는지 판단하는데 필요한 계산 시간이 $O(n)$이므로 가시성 그래프를 구성하는데 필요한 계산 시간은 $O(n^3)$이다. 실제로 적용하기엔 많은 시간이 걸리기 때문에 이를 단축시키기 위한 다양한 방법이 제안되었으며(Ghosh and Mount, 1991; Overmars and Welzl, 1988; Welzl, 1985), 트리 구조를 이용하여 빠른 가시성 그래프 생성 알고리즘의 계산 시간은 $O(n^2 bg\ n)$이다.

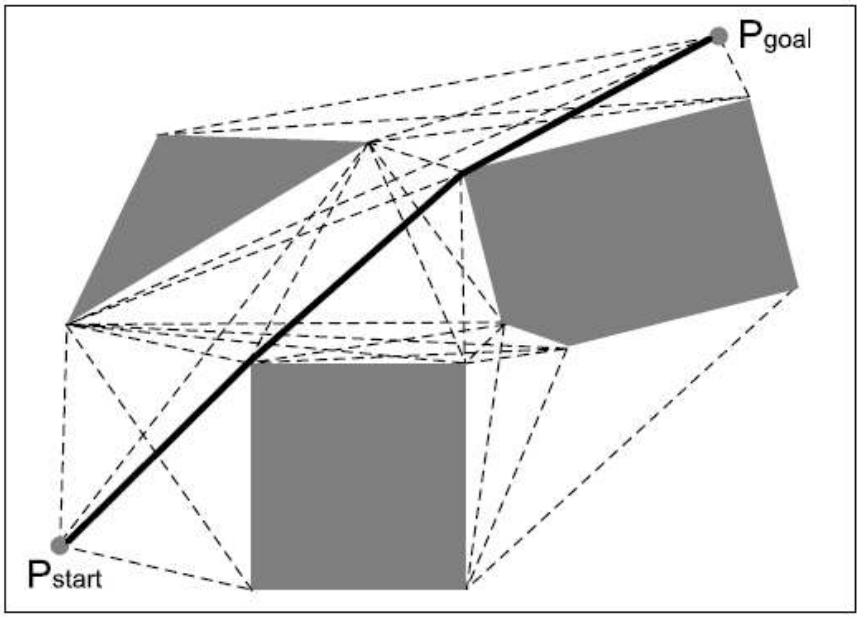

Figure 2-7 An example of visibility graph with start and goal node

(Kim et al., 2011)

가시성 그래프 생성 알고리즘이 개선되었으나, 여전히 계산 시간이 많이 필요해서 사용하기 어렵다. 장애물이 적은 2차원으로 구성된 환경에서 사용해야 하며, 환경 중 일부가 바뀌면 그래프를 업데이트하는데 많은 시간이 소모되기 때문에 현실적으로 사용하기 어렵다.

2.1.6 Comparison of environment modeling

위와 같이 기존에 사용되는 다섯 가지 환경 모델링 방법에 대해 알아보았다. 탐색 메시와 쿼드 트리는 장애물이 많고 가변적인 환경에서 사용하기에 용이하다. 탐색 메시는 장애물을 정확하게 표현하면서 메모리 소모량이

적어 탐색 메시 내 경로를 빠른 시간 내에 탐색할 수 있다. 하지만, 탐색 메시를 구성하는 방법은 쿼드 트리에 비해 복잡하다. 또한, 주어진 환경에 따라 탐색 메시가 결정되므로 장애물이 없는 영역에 대해 탐색 메시가 불완전하게 생성될 수 있다. 다섯 가지 방법의 최적 경로, 계산 시간, 실시간 구성 여부, 메모리 소모량, 완전성 여부를 다음 표에 나타냈다.

다음 표에서 주목해야 할 부분은 경로의 최적화 여부(optimality)이다. 가시성 그래프를 제외한 균일 격자, 비균일 격자, 보로노이 다어이그램, 탐색 메시 위에서는 최적화된 경로를 찾을 수 없다. 환경을 모델링하는데 필요한 계산 시간은 가시성 그래프가 $O(n^3)$으로 가장 높았으며, 그 다음으로 균일 격자가 $O(n^2)$, 나머지 방법들이 $O(nbgn$)이다.

본 논문에서는 장애물이 많은 복잡한 해양환경에서 최적 경로를 찾아야 한다. 장애물이 많은 해양환경에 적용하려면 각 방법으로 환경을 모델링하면 쿼드 트리는 계층적 특성(Hierarchical nature)으로 인해 장애물이 많은 지형에 적용 가능하다. 쿼드 트리 위에서 경로 탐색을 하게 되면 최적 경로와 다른 경로가 생성될 가능성이 높은데 이를 가시성 그래프를 이용하여 보완하였다. 본 연구는 쿼드 트리의 장점과 가시성 그래프의 장점을 통합하여 최적 경로를 탐색하였다.

Table 2-1 A comparison of environmental modeling methods

	Regular grids	Irregular grids (Quadtree)	Navigation mesh	Voronoi diagram	Visibility graph	The proposed method
Construction component	Same polygon	Different size of square	Convex polygon mesh	Voronoi diagram of obstacles	Link between all obstacle points	Quadtree+ Visibility
Complex environment representation	No full coverage	Full coverage	Full coverage	Full coverage	Full coverage	Full coverage
Memory consumption	High	Low	Low	Medium	Very high	Low
Computation time	-	$O(nbgn)$	$O(nbgn)$	$O(nbgn)$	$O(n^3)$	$O(k^2bgn)$ (k is small)
Path optimality	Not optimal	Not optimal	Not optimal	Not optimal	Near optimal	Near optimal

2.2 Path planning methods

2.1절에서 환경을 모델링하는 방법에 대해 설명하였다. 2.2절에서는 기존에 개발된 그래프 기반 알고리즘, 샘플링 기반 알고리즘, 생물 구조 기반 알고리즘에 대해 설명한다.

2.2.1 Graph based methods

그래프에서 경로를 탐색하는 알고리즘은 Dijkstra 알고리즘과 A* 알고리즘이 대표적이다. 두 알고리즘은 그래프의 형상에 관계없이 최적해를 찾는 방법이며, Figure 2-8은 격자 기반 그래프에서 Dijkstra 알고리즘과 A* 알고리즘의 예시를 보여준다. Dijkstra 알고리즘은 그래프 내 모든 비용을 계산하기 때문에 전역 최적해를 찾을 수 있다는 장점이 있지만, 시간 복잡도가 $O(Ebg\ V)$로 간선(E)의 수가 증가할수록 시간이 오래 걸린다(Dijkstra, 1959). A* 알고리즘은 휴리스틱 함수를 사용하여 모든 노드를 탐색하지 않고 목적지에 가까운 노드만 탐색한다(Hart et al., 1968).

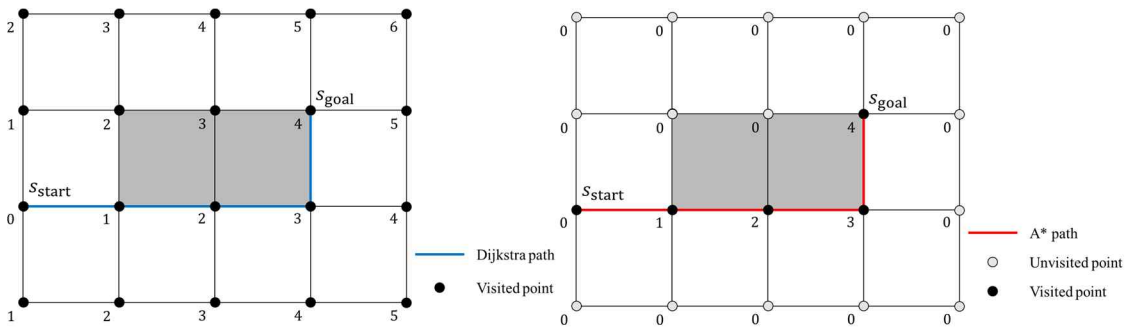

Figure 2-8 The example of the Dijkstra algorithm and the A* algorithm

격자 위에서 경로를 탐색하게 되면 경로 생성 시 각도가 4방향일 경우 90도, 8방향일 경우 45도로 제한된다. 제한된 각도 때문에 불필요한 점들이 생성되며, 이는 격자가 아닌 공간 위에서는 실제 최적 해가 아니다. 이러한 단점을 극복하기 위해 각도에 제한이 없는 D*, Field D* 알고리즘 등이 개발되었다(Ferguson and Stentz, 2006).

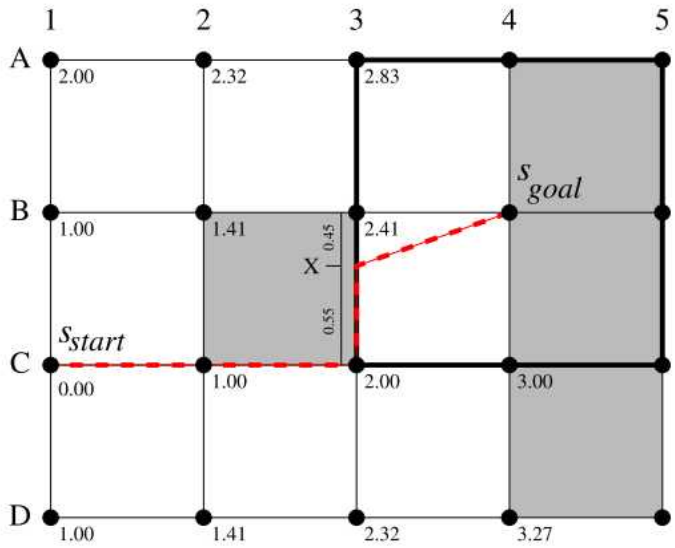

Figure 2-9 Field D* algorithm (Daniel et al., 2010)

Figure 2-9는 Field D* 알고리즘의 결과를 보여준다. 기존 격자에서 90도로 꺾이는 점이 사라지긴 했지만 여전히 경로 내에 불필요한 점들이 존재한다. 이러한 점들 때문에 Field D* 알고리즘은 격자 위에서 최적 경로를 찾을 수 없다.

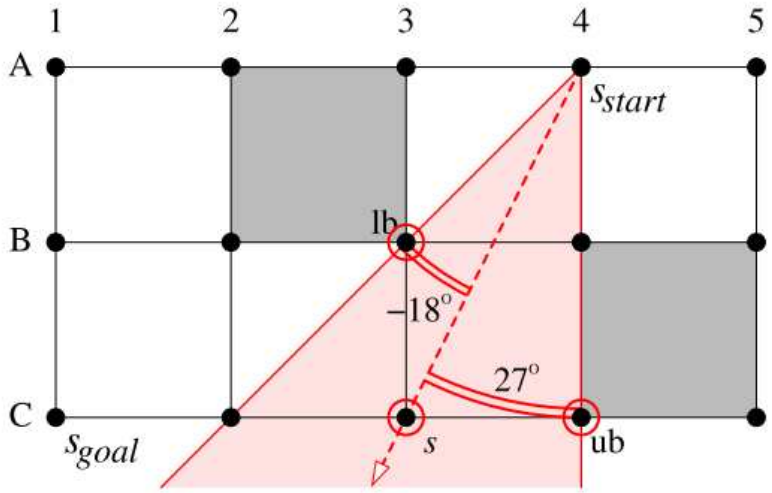

Figure 2-10 AP Theta* algorithm (Daniel et al., 2010)

Filed D* 알고리즘의 단점을 극복하기 위해 Theta* 알고리즘이 제안되었다. Theta* 알고리즘은 선택된 점에서 양 방향으로 가장 멀리 떨어진 점과 간선을 연결하여 경로를 찾는 알고리즘으로 가시성 그래프와 유사하다. 양 끝점만을 고려하기 때문에 경로가 실제 최적 경로와 다를 수 있다. 이러한 단점을 해결하기 위해 AP Theta* 알고리즘이 제안되었다. AP Theta* 알고리즘의 예시는 Figure 2-10에 나타냈다. Figure 2-10에서 볼 수 있듯 AP Theta*는 보이는 모든 점들과 연결하기 때문에 격자 위에서 최적 경로를 찾을 수 있다. 다만, 연결하는 점들이 늘어나기 때문에 시간은 기존 Theta* 알고리즘보다 더 걸린다.

2.2.2 Sampling based methods

샘플링 기반 알고리즘은 확률 로드맵(Probabilistic roadmap)과 RRT 방

법이 있다. 영역에 임의로 장애물이 존재할 때 Figure 2-11과 같이 샘플링을 수행하여 영역 위에 점들을 설정한다. 각 점에서 가까운 점들을 찾고 이들이 장애물과 겹치는지 확인한 후 겹치지 않는다면 그 점을 연결한다. 이 과정을 확률 로드맵 시작점과 출발점을 설정한 후 생성한 로드맵을 이용하여 경로를 계산한다. 이 방법은 복잡한 형상의 장애물이 존재하여 균일 격자로 표현하기 어려울 때 사용된다. 하지만, 충분한 수의 샘플링이 이루어지지 않는다면 경로를 찾을 수 없다. 샘플링의 수가 많아질수록 계산 시간이 증가하기 때문에 장애물이 많은 환경에서는 적용하기 어렵다. 이러한 단점을 극복하기 위해 RRT가 개발되었다.

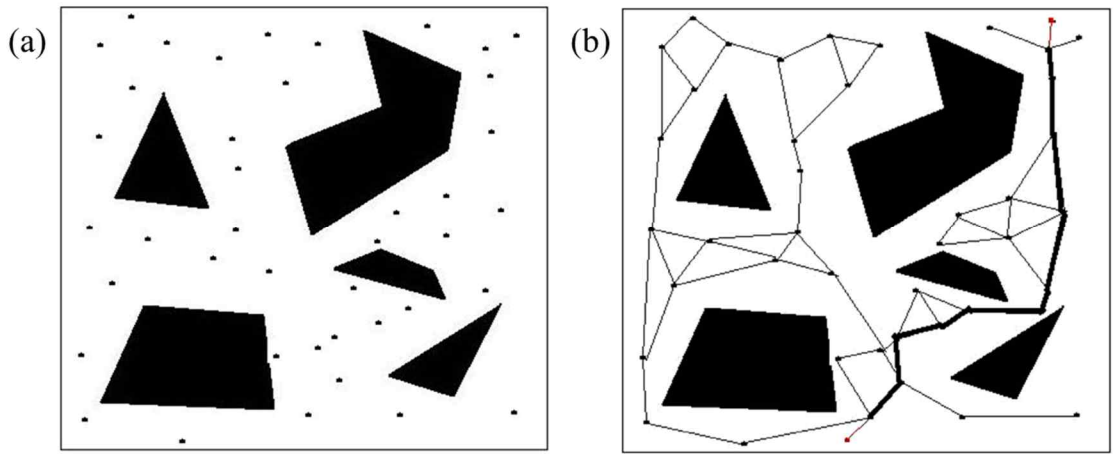

Figure 2-11 Probabilistic roadmap(Nieuwenhuisen and Overmars, 2004)

RRT(Rapidly exploring tree)와 이를 변형한 RRT* 알고리즘은 샘플링을 기반으로 트리 구조를 활용하는 알고리즘이다. (Karaman and Frazzoli, 2011a). RRT와 RRT*의 과정을 아래에 나타냈다. RRT는 아래 Figure 2-12의 (a), (b)만 수행하는 방법이며, RRT*는 (a)~(f)까지 수행하는 방

법이다. 임의의 영역에 대해 시작점으로부터 랜덤하게 점을 생성하고 이를 기반으로 트리를 구성한다. 랜덤하게 생성한 점을 가장 가까운 노드와 연결하여 트리 구조를 업데이트하며, 트리에 추가되는 노드가 도착지에 도달할 때까지 위 과정을 반복한다. RRT*는 RRT 알고리즘과 노드 추가 방식은 동일하다. 다만, RRT와 차이점은 상위 노드의 재선정과 트리의 재구성이다.

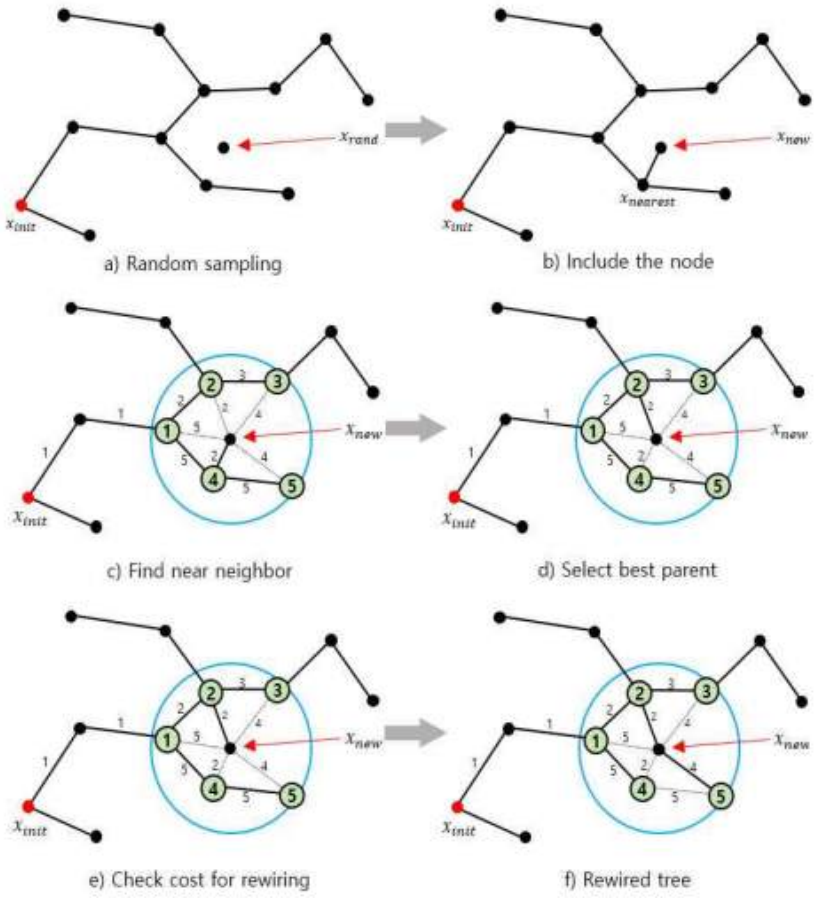

Figure 2-12 The example of RRT and RRT*

(Tak et al., 2019)

RRT에서는 Figure 2-12(b)에서 볼 수 있듯, 가장 가까운 노드가 상위

노드가 된다. RRT*에서는 Figure 2-12(c)에서 볼 수 있듯, 일정 반경에 있는 노드를 찾고 그 노드까지의 비용을 모두 계산한다. 만약, Figure 2-12(f)와 같이 하위 노드로 연결했을 때 비용이 더 작으면 기존 상위 노드와의 연결을 끊고 하위로 연결하여 트리를 재구성한다.

RRT*는 충분하게 많은 수의 샘플링을 수행하면 최적해로 수렴한다는 것이 증명되었다. RRT*는 고차원에서 최적 경로를 계산하는데 많이 적용되고 있으며 그 효용성이 입증되었다. RRT의 변형 버전은 RRT*의 변형 버전이며, 다양한 RRT* 변형 버전이 제안되었다(Jaillet et al., 2010; Karaman and Frazzoli, 2011b; Noreen et al., 2016).

2.2.3 Evolutionary methods

경로 최적화에 사용되는 진화 알고리즘 중 대표적인 알고리즘은 유전 알고리즘, 입자 군집 최적화(Particle swarm optimization), 개미 군집 최적화(Ant colony optimization) 등의 방법이 있다. 유전 알고리즘은 경로를 유전자로 표시하고 유전 연산을 통해 세대를 반복하여 더 좋은 최적 경로를 찾는 방법이다. 먼저, 초기 해 집합을 임의로 선정하고 이에 대해 목적함수를 평가하고 목적함수 값이 높은 해를 선택하여 저장한다. 현재 세대수가 설정한 최대 세대 수보다 작을 경우 유전 연산(교차, 돌연변이)을 수행하여 다음 세대의 해 집합을 생성한다. 이 과정을 반복하여 최적 해를 찾는 과정이 유전 알고리즘이다.

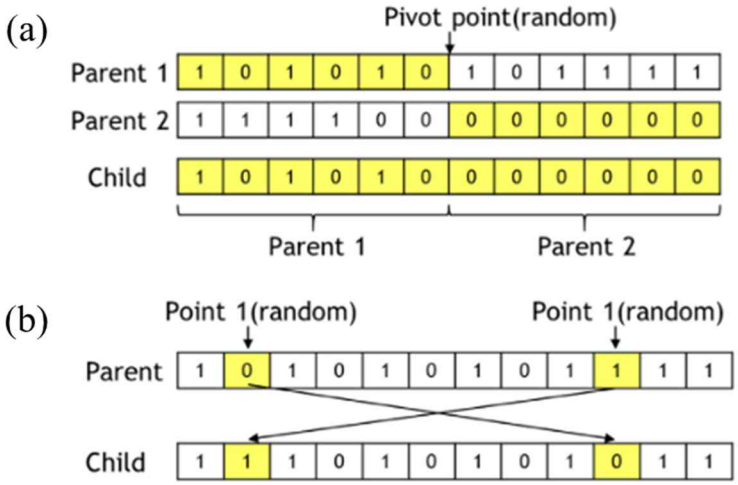

Figure 2-13 Crossover and mutation in genetic algorithm (Kim and Kim, 2017)

다음으로, 경로를 찾기 위해 사용되는 진화 알고리즘은 입자 군집 최적화 (Particle swarm optimization) 방법이 있다. PSO는 전역 최적해를 찾기 위해 사용되며 반복적인 계산을 통해 최적화를 수행한다. Figure 2-14는 PSO의 예시를 보여주며, 두 점이 최적해에 도달하는 과정을 보여준다. 이와 같이 PSO는 해가 될 수 있는 공간에서 무작위로 초기 해를 생성하고 설정한 식에 따라 최적 해를 찾는 방법이다. 유전 알고리즘에서 유전 연산을 수행하지 않으면 PSO가 되며, 진화론적 알고리즘 중 빠른 시간 내에 최적 해를 찾을 수 있다는 장점이 있다. 다만, 해결해야 할 문제가 수식으로 표현하기 복잡하면 PSO는 사용하기 어렵다.

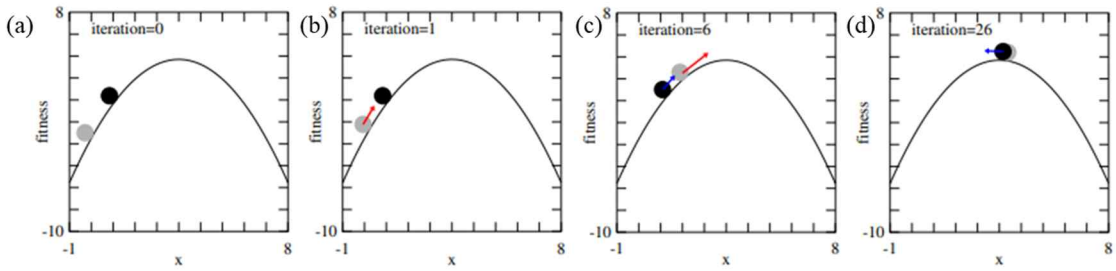

Figure 2-14 An example of particle swarm optimization (Chopard and Tomassini, 2018)

마지막으로, 개미 군집 최적화 방법(Ant colony optimization)이 있다. ACO는 그래프 위에서 최적 경로를 찾을 때 자주 사용하는 방법으로 초기에 임의로 해를 계산한다. 이 경로의 목적함수 값에 따라 값이 좋으면 그 경로의 가중치를 낮추고 값이 나쁘면 경로의 가중치를 높여 다음 단계에서 경로를 계산할 때 목적함수 값이 좋은 경로가 나올 수 있게 설정한다. 이 방법이 개미가 다른 개미들을 위해 페로몬을 생성하는 과정과 유사하여 개미 군집 최적화라 불린다.

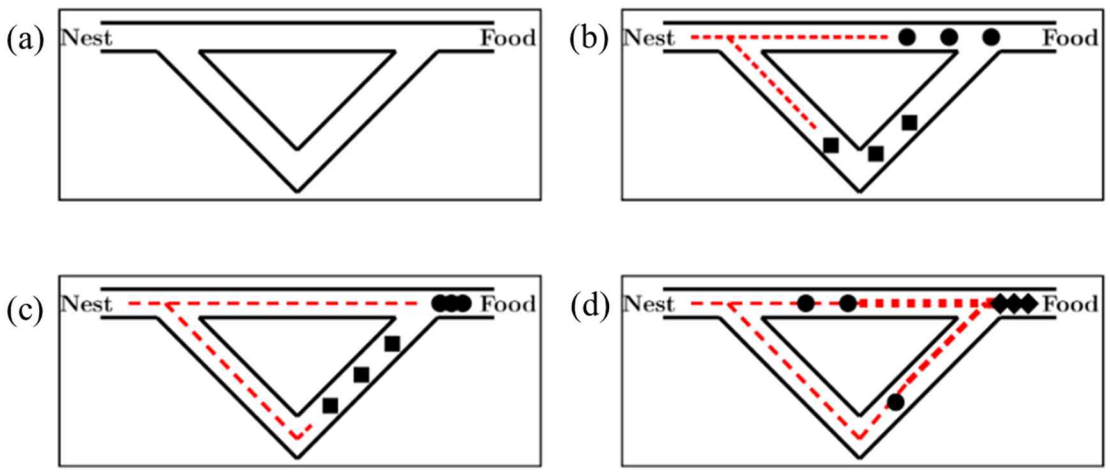

Figure 2-15 An example of ant colony optimization (Blum, 2005)

2.2.4 Comparison of path planning methods

2.2절에서는 경로 최적화 방법에 대해 알아보았다. 샘플링 기반 알고리즘과 진화 알고리즘들은 그래프를 특정 형상에 제한되지 않고 생성하여 최적 해를 찾을 수 있다는 장점이 있으나 시간이 오래 걸린다는 단점이 있다. 알고리즘의 계산 시간을 증가시키는 데에 육지와의 간섭을 판단하는 부분이 많은 역할을 한다. 본 논문의 적용 대상인 연안에서는 해안선이 복잡하고 장애물이 많기 때문에 알고리즘의 계산 시간이 크게 증가하며, 이러한 이유로 샘플링 기반 알고리즘과 진화 알고리즘들은 빠른 시간 내에 경로를 계산할 수 없다.

그래프 기반 알고리즘은 Dijkstra 알고리즘과 A* 알고리즘으로 나뉜다. Dijkstra 알고리즘은 전역 최적해를 계산하고, A* 알고리즘은 지역 최적해를 계산하므로 알고리즘의 적용 환경이 매우 중요하다. 지역 최적해와 전역 최적해가 비슷하다면 계산 성능 때문에 A* 알고리즘을 사용하는 것이 좋다. 하지만, 지역 최적해와 전역 최적해의 차이가 크다면 시간이 오래 걸려도 Dijkstra 알고리즘을 사용해야 한다. 연안 환경에서는 장애물이 많아 휴리스틱 함수를 거리로 설정하고 A* 알고리즘을 수행하면 Dijkstra 알고리즘의 결과와 다를 가능성이 높다. 따라서, Dijkstra 알고리즘을 수행하여 최적 항로를 계산한다.

3 Related work

경로 계획 문제의 개념과 각 방법들은 앞서 2장에서 설명하였다. 본 연구는 쿼드 트리와 가시성 그래프를 이용하여 연안 내 최적 항로를 결정하였다. 본 절에서는 연안 내 최적 항로에 사용된 방법에 따라 세 가지 기존 연구에 대해 분석하고, 본 연구의 방법과 이를 비교하고자 한다.

3.1 Genetic algorithm with navigation mesh

Tsou (2010)는 GIS(Geographic Information System)을 이용하여 연안에서의 최소 거리를 갖는 최적 항로를 계산하였다. 장애물을 탐색 메시와 비슷하게 다각형으로 변환하고 다각형의 점들을 이용하여 항로를 계산하였다. 항로 계산 방법은 유전 알고리즘을 사용하였으며, 유전 알고리즘의 해는 Figure 3-1과 같이 장애물을 둘러싸고 있는 점들이다. 이러한 점들을 무작위로 조합하여 초기 해를 생성한 후 최적 경로가 될 때까지 세대를 반복하였다.

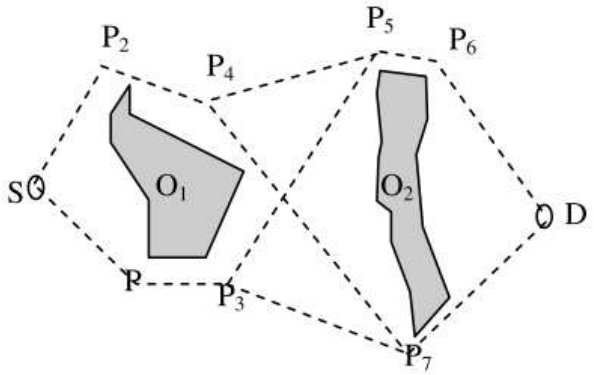

Figure 3-1 Bilateral multi-route generation (Tsou, 2010)

Figure 3-2는 유전 알고리즘을 이용하여 계산한 항로 예시이다. 마지막 세대에서 최적 해와 그렇지 않은 해를 나타낸 것이며, 육지와 장애물을 매우 단순화하여 알고리즘을 수행한 것을 확인하였다.

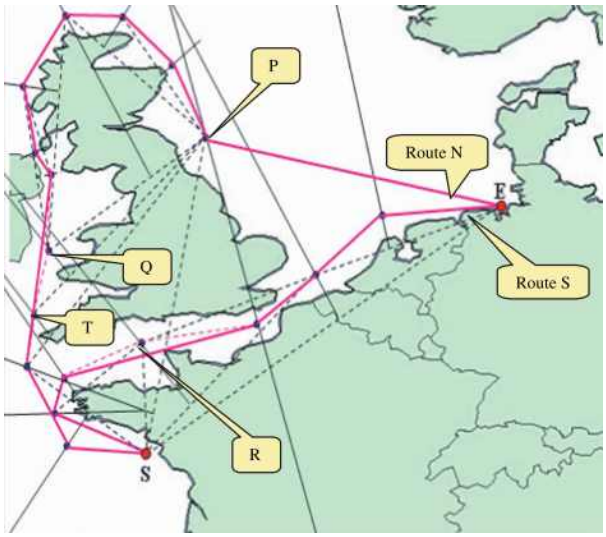

Figure 3-2 An example of multi-route between two points (Tsou, 2010)

3.2 Dijkstra algorithm with Voronoi diagram

Niu et al. (2019)은 싱가포르 해역에 대한 육지에 대해 보로노이 다이어그램을 구성하였다. 해안선과 섬의 점들을 이용하여 보로노이 다이어그램을 구성할 수 있으며 보로노이 다이어그램에 기반하여 Dijkstra 알고리즘을 수행하였다. 보로노이 다이어그램은 장애물들의 중앙에서 생성되기 때문에 보로노이로 다이어그램에 기반하여 Dijkstra 알고리즘을 수행하면 최단 거리 항로가 나올 수 없다. 보로노이 다이어그램에 의해 계산한 항로는 Figure 3-3과 같다.

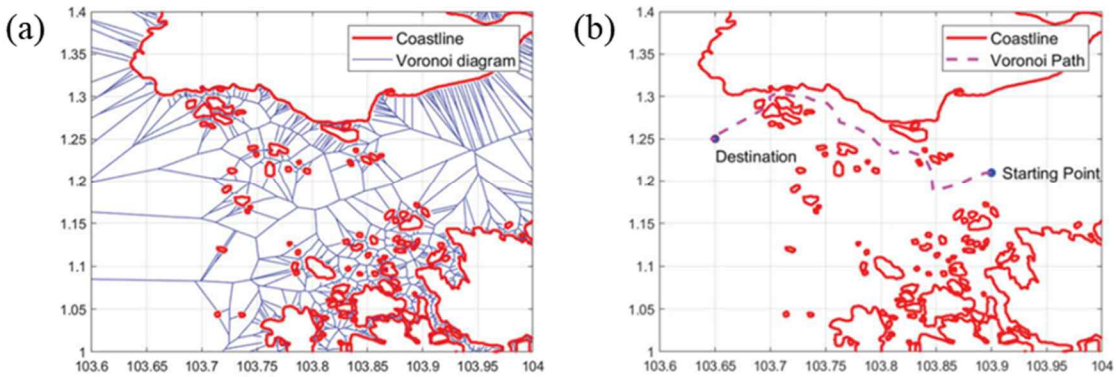

Figure 3-3 Voronoi diagram and Voronoi path (Niu et al., 2019)

Figure 3-3에서 볼 수 있듯, 보로노이 다이어그램에서 계산한 항로는 최적 경로가 아니다. Niu et al. (2019)은 보로노이 다이어그램으로부터 계산한 경로에서 불필요한 점을 제거하기 위해 가시성 그래프를 이용하였다.

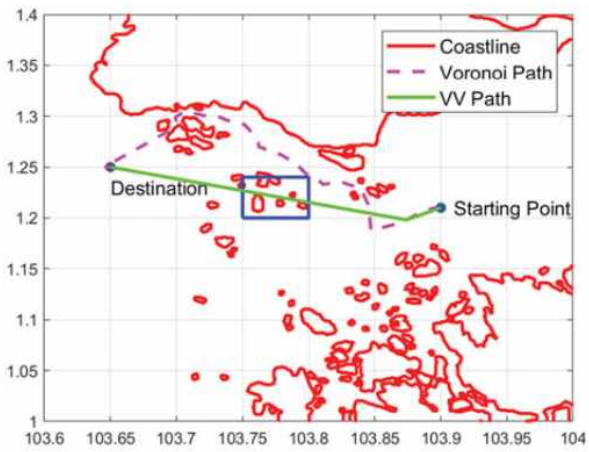

Figure 3-4 Path refinement with visibility graph (Niu et al., 2019)

3.3 A* algorithm with quadtree

Shah and Gupta (2020)는 미국의 복잡한 지형이 존재하는 영역에 대한

최단 거리 항로를 계산하는 방법을 제안하였다. Shah and Gupta (2020)는 복잡한 지형을 쿼드 트리를 이용하여 표현하였다. 항로를 탐색하기 위해 A* 알고리즘을 사용하였으며, 새로운 휴리스틱 함수를 설정하였다.

Figure 3-5 Quadtree representation with complex shaped obstacle (Shah and Gupta, 2020)

육지를 표현하는 쿼드 트리를 하나의 장애물로 합쳐서 장애물을 단순화하였으며, 이러한 장애물 뒤에 있는 점들을 제거하여 항로를 계산하는데 불필요한 연산을 줄였다. 또한, A* 알고리즘을 적용할 때, 도착점까지의 직선과 장애물까지의 거리를 기반으로 휴리스틱 함수를 설정했다. 휴리스틱 함수는 최적 경로는 출발지부터 도착지까지의 직선과 교차하는 장애물의 한 점(corner)를 지난다는 가정 하에 설정되었다.

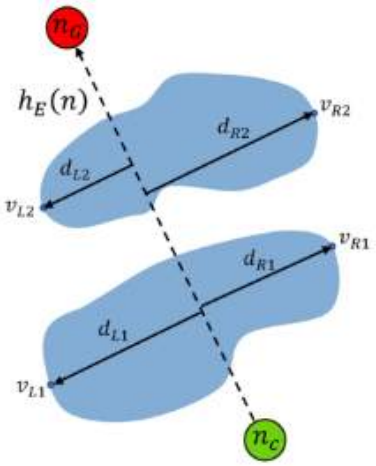

Figure 3-6 Designed heuristic function (Shah and Gupta, 2020)

하지만, 이 방법은 항로의 웨이 포인트가 쿼드 트리로 표현한 육지 위에만 설정된다는 단점을 지니고 있다. 또한, 영역에 비해 장애물의 밀집도가 높을 때에는 시간이 오래 걸린다는 단점이 있다.

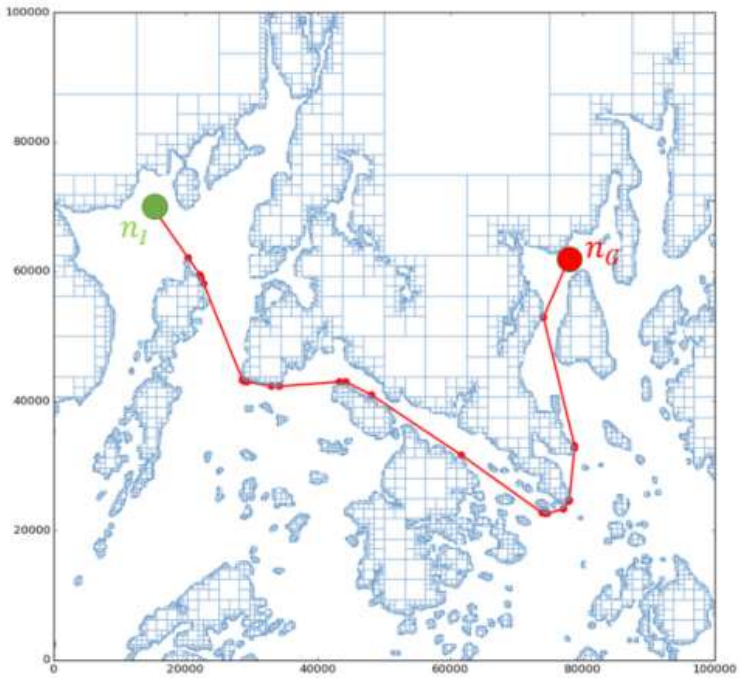

Figure 3-7 Computed path on a real-world scenario (Shah and Gupta, 2020)

3.4 Comparison of related works with this thesis

위와 같이 기존에 수행된 세 가지 연구에 대해 알아보았다. 기존의 연안에서 항로를 계획하는 연구들은 섬과 장애물의 복잡한 형상 때문에 최단거리만 계산하는 것을 알 수 있었다.

기존의 연구와 본 연구의 항로 계산 방법과 시간을 비교하여 정리하면 아래 Table 3-1, Table 3-2와 같다. 본 연구의 항로 계획 방법은 시간에 상관없이 일정한 육지, 수심과 시간에 따라 변하는 기상 및 조위 정보를 사용하였으며, 최적화 과정에서 목적함수는 항해 시 필요한 연료 소모량으로 설정하였다. 계산 시간을 빠르게 하기 위해 쿼드 트리 기반 가시성 그래프를 사용하였으며, 다른 연구와 비교했을 때 빠른 시간 안에 최적 항로를 찾아낼 수 있도록 알고리즘을 설계하였다. 또한, 다른 연구에서는 기상 정보를 조위 혹은 조류 정보만 사용하였으나 본 연구에서는 바람, 파랑, 조류, 조위 정보를 사용하여 선박의 항해 시 연료 소모량이 최소가 되는 항로를 탐색하였다.

Table 3-1 A comparison of related work (Weather routing)

Item		Related work			This thesis (coastal routing)
		1. Lee et al. (2018)	2. Li et al. (2018)	3. Ma et al. (2020)	
Ocean and weather condition source	Wind	Considered	Considered	Not considered	Considered
	Wave	Considered	Considered	Not considered	Considered
	Current	Not considered	Not considered	Not considered	Considered
	Tide	Not considered	Not considered	Considered	Considered
Environment representation		Weather routing	Weather routing	Weather routing	Quadtree Visibility graph
Weight determination		Regression model (Weather data and ship data)	Route length and voyage time	Integration of fuel cost and time cost	Weather increase resistance → Engine power increase → FOC Increase

Optimization	Objective function	Fuel efficiency	Voyage time	Fuel and time cost	Fuel efficiency
	Method	Genetic algorithm	Ant colony optimization	Genetic algorithm	Dijkstra algorithm
Computation time		More than 10 min	Not specified (more than 5 minutes due to ant colony optimization)	Not specified (more than 5 minutes due to genetic algorithm)	About 20 s
Notes		RPM−power, RPM−UFOC, and speed−power curve generated	Consideration of dangerous area	Consideration of Emission Control Area (ECA)	Visibility path using waypoints of quadtree path

Table 3-2 A comparison of related works (USV path planning)

Item		Related works			This thesis (coastal routing)
		1. Tsou (2010)	2. Niu (2019)	3. Shah (2020)	
Ocean and weather condition source	Wind	Not considered	Not considered	Not considered	KMA
	Wave	Not considered	Not considered	Not considered	KMA
	Current	Not considered	Tidetech	Not considered	KMA
	Tide	Not considered	Not considered	NOAA	KHOA
Environment representation		Similar to navigation mesh	Voronoi diagram Visibility graph	Quadtree	Quadtree Visibility graph
Weight determination		Sum of the distance, navigable areas with threat, and benefit navigable areas	Current Speed change → Drag force change → Energy change	Distance	Weather increase resistance → Engine power increase → FOC Increase

Optimization	Objective function	Distance & safety	Energy	Distance	Fuel efficiency
	Method	Genetic algorithm	Dijkstra algorithm	A* algorithm	Dijkstra algorithm
Computation time	Map	Not real time	Not real time	Real time	Real time
	Path	more than 1 min	About 1 min	more than 20 s	about 20 s
Notes		Minimum bounding rectangle with obstacle avoidance	Visibility path using waypoints of Voronoi path	Quadtree decomposition in obstacles	Visibility path using waypoints of quadtree path

4 Data acquisition and preprocess

4.1 Topography data

연안 항로의 탐색에서 이동 가능한 경로인지 판단하기 위해 지형 정보가 필요하다. 본 연구에서는 이를 위해 전자해도 정보를 추출하여 사용하였다. 4.1.1절에서 전자해도 정보에 대해 자세하게 설명하며, 4.1.2절에서 전자해도 정보를 전 처리하는 방법에 대해 설명한다.

4.1.1 Topography and depth

육지 및 해안선 정보를 얻기 위해 국립해양조사원의 전자해도 정보를 사용하였다. 전자해도 정보는 해안선, 수심, 위험물 등의 다양한 정보를 포함하고 있으며, 전자해도 표시시스템(ECDIS)이라고 부른다. 이 정보는 국제수로기구(IHO)의 표준규격인 S-57에 따라 제작된다.

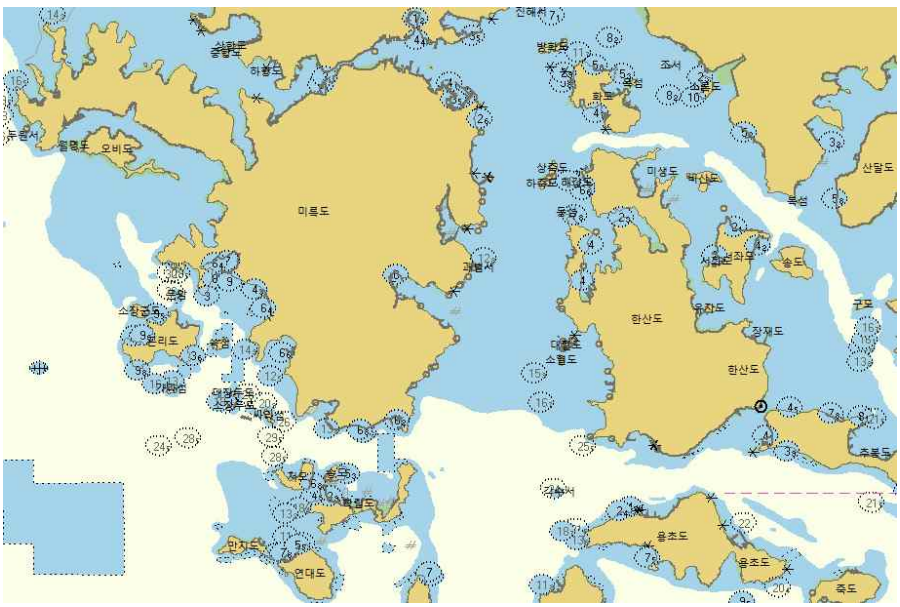

Figure 4-1 The example of electronic navigational chart

전자해도는 항해목적별로 다음과 같이 구분된다. 레벨이 낮을수록 축척이 낮기 때문에 정보의 품질이 떨어지며 메모리 사용량이 적고, 레벨이 높을수록 축척이 높아 정보의 품질이 올라가며 메모리 사용량이 많아진다. 모든 영역에 대해 레벨 별로 정보가 있는 것이 아니라 일정 영역 별로 정보가 존재한다. 예를 들어, 항구 근처에서는 레벨 6의 항박도가 존재하며, 항만 근처에는 레벨 5의 항만도가 존재한다. 따라서, 경로의 정확도를 높이기 위해 높은 품질의 전자해도 정보를 사용해야 하므로 각 레벨 별로 존재하는 해안선 데이터를 전부 사용한다.

Table 4-1 Classification of electronic navigational chart in voyage objective

ECN level	Chart name	Scale
1	총도 (Overview chart)	Larger than 1,500,000
2	함양도 (General Chart)	350,000 ~ 1,499,999
3	해안도 (Coastal Chart)	90,000 ~ 349,999
4	항만접근도 (Approach Chart)	30,000 ~ 89,999
5	항만도 (Harbor Chart)	7,500 ~ 29,999
6	항박도 (Berthing Chart)	Smaller than 7,500

항만도나 항박도에는 지형에 대한 자세한 정보가 있다. 1°×1° 영역 안에 최소 10,000개 이상의 점이 있으며, Figure 4-2는 경도 구간 (127.7°, 128.7°), 위도 구간 (34.5°, 35.5°) 구간에 있는 해안선과 섬을 표현한 것

이다. 이 영역 내에서는 내륙을 포함하여 391개의 섬이 있으며, 이들을 표현하기 위해 54,625개의 점이 존재한다.

복잡한 해양 환경에서 항로를 결정하는데 영향을 주는 가장 중요한 요소는 섬이나 해안선과 같이 고해상도 공간 데이터 (High-solution spatial data)와 간섭을 확인하는 과정이다. 복잡한 해안선이 존재하는 환경 내에서 경로를 계산하기 위해서는 지형을 자세히 이산화(discretization)해야 한다.

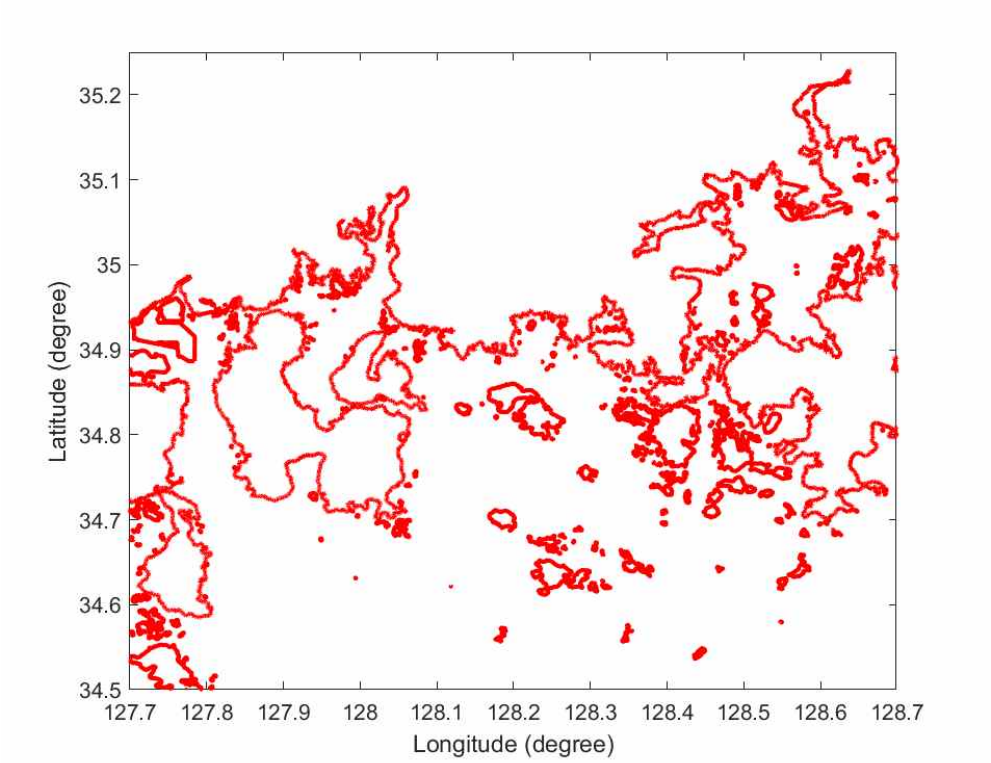

Figure 4-2 South Korea coastline and islands

지형을 자세히 표현하는 가장 좋은 방법은 가시성 그래프를 적용하는 방법이다. 고해상도 공간 데이터에 가시성 그래프 (Visibility graph)를 적용하면 최적 경로를 계산할 수 있지만 가시성 그래프는 계산 시간 및 메모리

비용이 매우 높다. 가시성 그래프를 생성하기 위해서는 가시성 선분(Visibility line)이 섬이나 해안선과 교차하는지 여부를 판별해야 한다. 가시성 그래프의 시간 복잡도는 $O(n^3)$이며 여기서 n은 가시성 그래프를 표현하기 위한 점 개수이다. 따라서, 가시성 그래프의 생성 시간은 n의 세제곱에 비례하므로 계산 시간 때문에 가시성 그래프를 직접 사용하는 것은 적합하지 않다.

4.1.2 Coastline expansion algorithm

경로의 품질을 향상시키기 위해 고해상도의 전자해도를 사용하였다. 그러나 고해상도의 전자해도 정보는 다른 지도의 정보와 차이가 존재할 수 있다. 아래 Figure 4-3은 전자해도 정보와 구글 어스의 정보를 비교한 예시이며, 빨간색 선은 ENC 정보에서 얻은 해안선이다. 구글 지도에서 얻은 섬 데이터는 ENC 정보와 비교하여 다른 형상을 나타낸다. 두 데이터에서 해안선 사이의 최대 거리는 약 100m이다. ENC 데이터 부정확성으로 인한 해안선과의 충돌을 피하면서 경로 계획을 수행해야 한다. 복잡한 환경에서 경로가 해안선에서 멀리 떨어져 있도록 여유 거리를 설정해야 한다.

Figure 4-3 Comparison ENC data (red line) and satellite image from Google Maps

해안선에서 여유 거리를 설정하는 방법은 해안선을 일부 확장하여 수행할 수 있다. 섬을 포함하여 해안선은 다각형(Polygon)으로 구성되어 있다. 다각형을 확장하는 방법은 여러 가지가 있으나 가장 안정적인 방법은 오프셋(Offset)을 적용하는 방법이 있다. 이차원에서 오프셋은 원을 기반으로 Minkowski 합을 이용한다. Minkoswki 합은 아래와 같이 정의되며, 다각형의 점들은 시계반대방향으로 설정된다.

$$A \oplus B = \{a + b \mid a \in A, b \in B\} \tag{4-1}$$

$$P = (p_0, \ldots, p_{n-1}) \quad in \quad counterclockwise \tag{4-2}$$

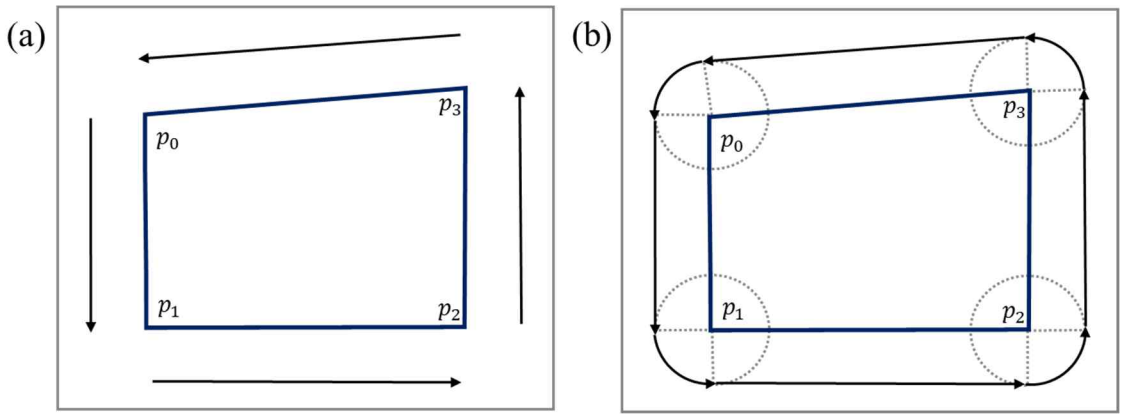

Figure 4-4 Polygon offset with convex polygon

다각형이 볼록(Convex)하다면 오프셋은 다각형의 간선 (edge)을 다각형으로부터 일정 거리만큼 평행 이동하는 것으로 계산된다. 다각형의 점들이 위 식과 같이 시계 반대 방향으로 주어져 있을 때 오프셋은 다각형 간선의 오른쪽 방향에 존재한다. 따라서, Figure 4-4(a)와 같이 끊어진 오프셋 간선들을 계산할 수 있다. 인접한 오프셋 간선들($p_{i-1}p_i, p_ip_{i+1}$)은 반경 r을 갖는 원호에 의해 연결된다. 이 때 원호의 중앙점은 p_i가 된다. 이 때 계산 시간은 다각형의 점 개수에 비례한다.

하지만, 대부분의 해안선과 섬은 볼록하지 않다. 다각형이 볼록하지 않다면(non-convex), 볼록한 다각형으로 아래 식과 같이 쪼갠 후 오프셋을 구하고 이들을 합쳐서 오프셋을 계산한다.

$$P = \cup_{i=1}^{m} P_i \quad if\ P_1, \quad \dots, P_m\ is\ sub-polygon \tag{4-3}$$

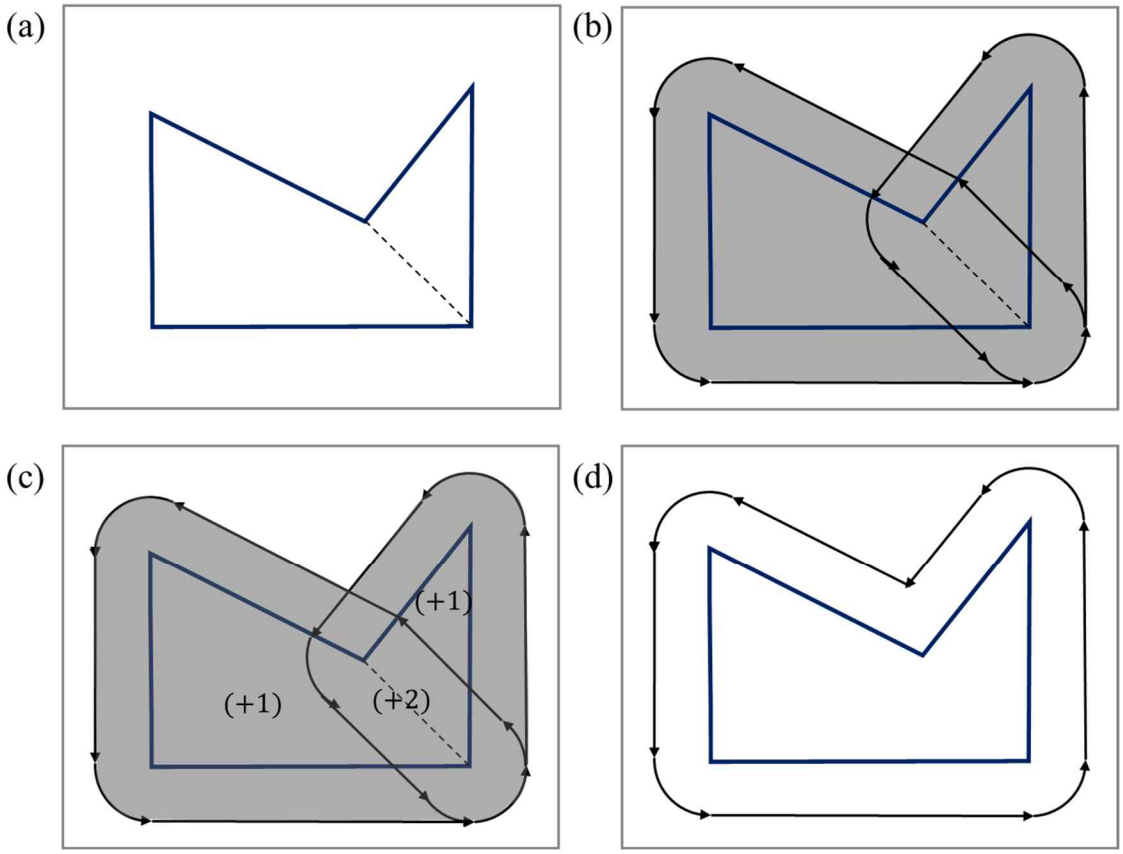

Figure 4-5 Polygon offset with non-convex polygon

아래 Figure 4-5(a)에서 볼록하지 않은 다각형은 점선을 추가하여 두 개의 볼록한 다각형으로 나뉜다. Figure 4-5(b)와 같이 나뉜 다각형(sub polygon)에 대해 오프셋을 계산할 수 있으며 이는 볼록한 다각형의 오프셋을 구하는 방법과 동일하다. 이후 다각형을 합칠 때는 감김 수(winding number)를 이용하여 감김 수가 0보다 큰 영역은 하나로 합친다. 감김 수는 Figure 4-6에서 볼 수 있듯 평면에서 폐곡선과 그 폐곡선이 지나지 않는 점이 있을 때 폐곡선이 점 주위를 시계반대 방향으로 감기는 횟수를 의미한다(Chen and McMains, 2005). Figure 4-5(c)에서 점선을 추가하여 다각형이 겹치는 부분이 있는데 그 부분은 곡선이 시계 반대방향으로 두

번 감기므로 +2로 설정된다. 나머지 부분은 +1로 설정되며, 다각형의 외부는 0으로 설정된다. 감김 수가 0보다 큰 영역에 대해 다각형을 합치면(union) Figure 4-5(d)와 같이 볼록하지 않은 다각형에 대한 오프셋을 계산할 수 있다.

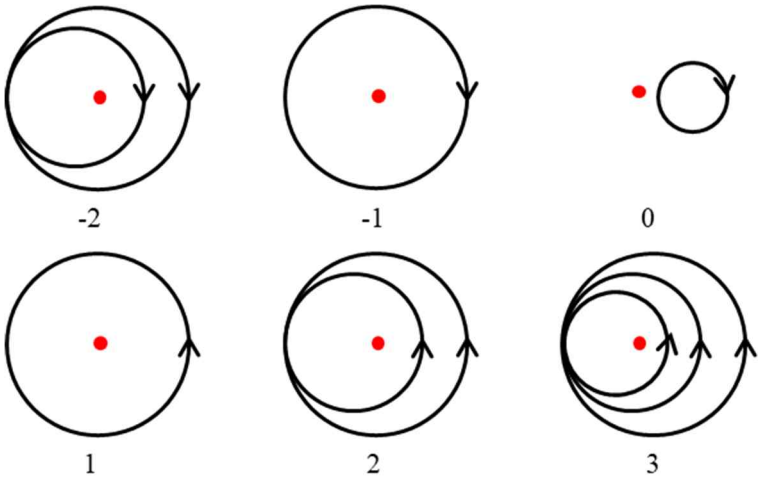

Figure 4-6 Example of winding number

오프셋을 구하기 위해 Clipper 라이브러리의 ClipperOffset 클래스를 사용하였다. 해안선을 확장하는 알고리즘은 시간 복잡도 $O(n)$이므로 점의 개수가 많아도 적용 가능하다. Figure 4-7은 한국 섬의 일부분에 대해 오프셋을 적용한 결과이다. 이 때 확장한 거리는 100m이다.

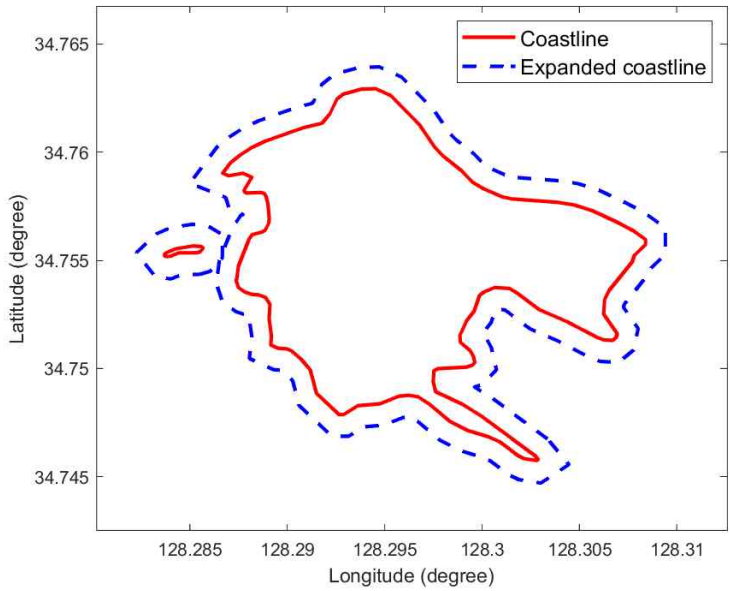

Figure 4-7 Illustration of coastline expansion algorithm output

4.2 Weather data

해상에서 항해하기 위해서는 해당 지역의 해상 정보와 기상 정보를 모두 고려해야 한다. 기상 정보에는 바람 속도, 바람 방향, 기온 등이 있으며, 해상 정보에는 파랑 방향, 파랑 높이, 조류 방향, 조류 속도, 염도 등이 해당된다. 이러한 정보를 아울러서 기상 해양학적 자료(metocean data)라고 한다. 본 연구와 같이 연안 항로에 직접적인 영향을 주는 정보는 바람, 파랑, 조류 정보이다.

본 논문의 시뮬레이션에 적용된 기상 해양학적 자료의 출처는 아래 Table 4와 같다. 바람, 파랑, 해류, 조위 정보가 사용되었다. 연안 항로를 계획하기 위해서 중기 예보가 필요하며, 총 4가지 자료가 사용되었다. 한국

기상청(Korea Meteorological Agency)에서 바람, 파랑, 조류 정보를 사용했으며, 한국 해양조사원(Korea Hydrographic and Oceanographic Agency) 조위 정보를 사용하였다.

Table 4-2 Source of ocean data

Item	Wind	Wave	Current	Tide
Forecast (mid-range)	KMA	KMA	KMA	KHOA

다음 4.2.1항부터 4.2.4항까지는 각각의 자료에 대해 설명한다.

4.2.1 Wave data (KMA)

파랑 정보는 한국 기상청 (Korea Meteorological Agency)에서 제공하는 지역 모델(Regional date assimilation and prediction system, RDAPS)의 정보를 사용하였다. 파랑 정보는 격자 정보로 이루어져 있으며, 각 데이터의 주요 제원은 아래 Table 4-3에 나타냈다. 한국 기상청에서 제공하는 영역은 한국의 영해를 포함하고 있기 때문에 한국 내에서 항로를 계산하는데 충분한 정보를 제공한다.

Table 4-3 Wave forecasting data from KMA

Item	Value	Note
Resolution	0.00833° (1 km)	1212 × 682 (longitudinal × latitudinal)
Cover range	122.0°E to 132.1°E 33.0° N to 38.68° N	Korea region
Medium-range forecast		
Total forecasting hours	72 hours (3 days)	3 hours × 24 records
Model cycle	8 times / day	
Time resolution	3 hours	

파랑 정보는 평균 파랑 높이, 최대 파랑 높이 (m), 파랑 방향, 파랑 주기 (s) 등의 정보를 포함하고 있으며, 본 논문에서는 아래 Figure 4-8, Figure 4-9과 같은 파랑 정보를 사용하였다.

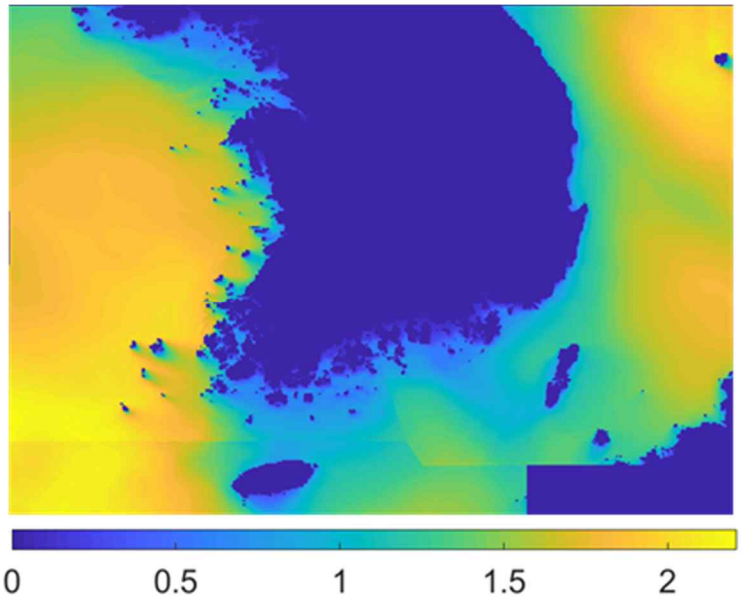

Figure 4-8 Wave height information (KMA)

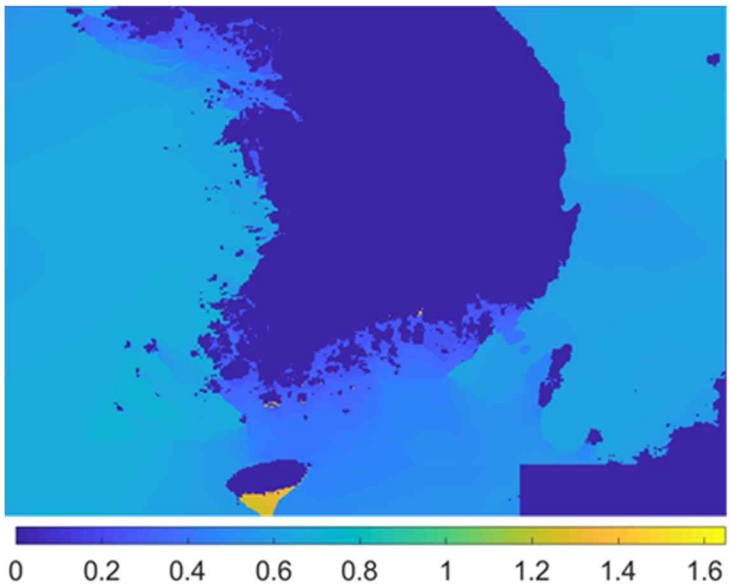

Figure 4-9 Wave period information (KMA)

4.2.2 Wind data (KMA)

바람 정보는 한국 기상청 (Korea Meteorological Agency)에서 제공하는

지역 모델(Regional date assimilation and prediction system, RDAPS)의 정보를 사용하였다. 바람 정보는 격자 정보로 이루어져 있으며, 각 데이터의 주요 제원은 아래 Table 4-4에 나타냈다.

Table 4-4 Wind forecasting data from KMA

Item	Value	Note
Resolution	0.00833° (1 km)	1212 × 682 (longitudinal × latitudinal)
Cover range	122.0°E to 132.1°E 33.0° N to 38.68° N	Korea region
Medium-range forecast		
Total forecasting hours	72 hours (3 days)	3 hours × 24 records
Model cycle	8 times / day	
Time resolution	3 hours	

바람 정보는 바람 방향과 바람 속도(m/s)의 정보를 포함하고 있으며, 본 논문에서는 아래 Figure 4-10와 같은 바람 정보를 사용하였다.

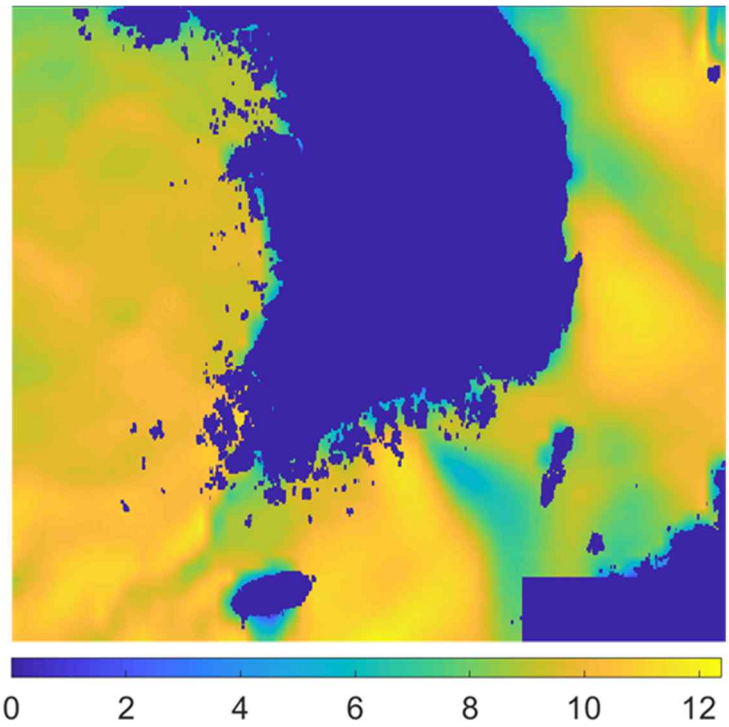

Figure 4-10 Wind speed information (KMA)

4.2.3 Current data (KMA)

조류 정보는 한국 기상청 (Korea Meteorological Agency)에서 제공하는 지역 모델(Regional date assimilation and prediction system, RDAPS)의 정보를 사용하였다. 조류 정보는 격자 정보로 이루어져 있으며, 각 데이터의 주요 제원은 아래 Table 4-5에 나타냈다. 조류 정보는 조류 방향과 조류 속도(m/s)의 정보를 포함하고 있으며, 본 논문에서는 아래 Figure 4-11과 같은 조류 정보를 사용하였다.

Table 4-5 Current forecasting data from KMA

Item	Value	Note
Resolution	0.00833° (1 km)	1212 × 682 (longitudinal × latitudinal)
Cover range	122.0°E to 132.1°E 33.0° N to 38.68° N	Korea region
Medium-range forecast		
Total forecasting hours	72 hours (3 days)	3 hours × 24 records
Model cycle	8 times / day	
Time resolution	3 hours	

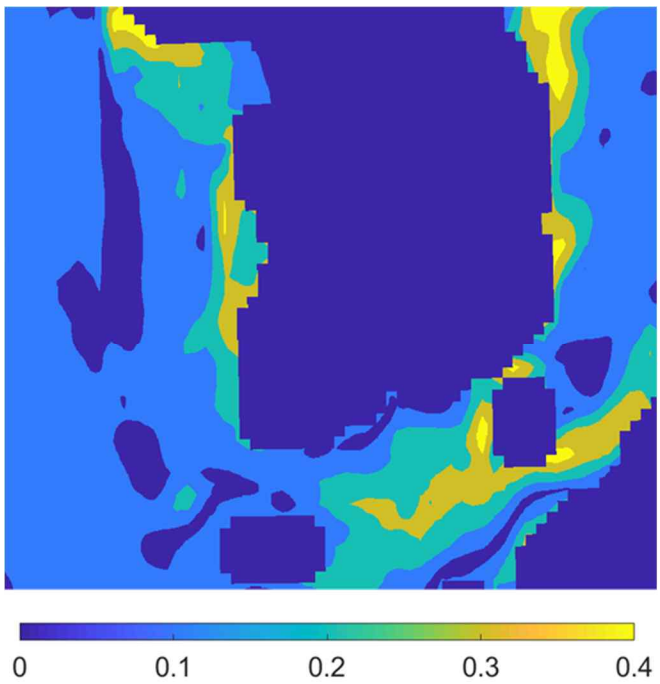

Figure 4-11 Current speed information (KMA)

4.2.4 Tide data (KHOA)

조위 정보는 한국 해양조사원 (Korea Hydrographic and Oceanographic Agency)에서 제공하는 지역 모델(Regional date assimilation and prediction system, RDAPS)의 정보를 사용하였다. 조위 정보는 격자 정보로 이루어져 있으며, 각 데이터의 주요 제원은 아래 Table 4-6에 나타냈다. 조위 정보는 격자로 구성되어 있으며, 해당 격자의 수심 변화 값을 포함하고 있다. 본 논문에서는 아래 Table 4-6과 같은 수심 정보를 사용하였다.

Table 4-6 Tide forecasting data from KMA

Item	Value	Note
Resolution	0.01° (1.2 km)	501 × 601 (longitudinal × latitudinal)
Cover range	125.0°E to 130.0°E 33.0° N to 39.0° N	Korea region
Medium-range forecast		
Total forecasting hours	72 hours (3 days)	3 hours × 24 records
Model cycle	8 times / day	
Time resolution	3 hours	

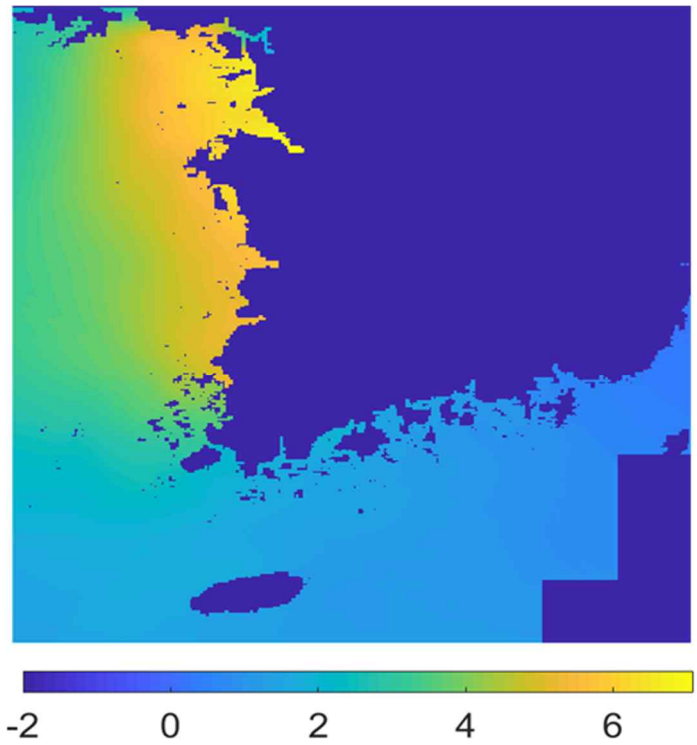

Figure 4-12 Tide information (KHOA)

5 Quadtree based graph

5.1 Quadtrees

쿼드 트리는 일정 영역을 반복하여 분할하면서 생성되는 공간 데이터 구조이며, 각각의 영역은 트리 형태로 저장된다. 즉, 각각의 분할된 영역은 동일한 크기와 모양의 분할된 하위 영역으로 나뉘며, 분할된 영역은 동일한 수의 하위 영역을 생성한다. 각 쿼드 트리 블록(Quadtree block) 혹은 셀(Cell)은 일반적으로 최상위 노드(root)로부터 분할된 수만큼 1/2의 거듭제곱 길이를 갖는다. 예를 들어 최상위 노드의 한 변이 16이면 3번 분할된 쿼드 트리 블록의 한 변은 2가 된다. 쿼드 트리 블록은 종료 조건(termination criteria)을 만족할 때까지 동일한 2차원의 하위 블록으로 나눌 수 있으며, 블록의 하위 블록은 각 좌표 축을 따라 블록을 1/4로 나눈 것이다. Figure 5-1은 2차원 공간 내의 단순한 쿼드 트리 분할 과정을 보여준다. 일반적으로, 쿼드 트리에서 하위 노드가 없는 노드가 리프 노드이며 리프 노드가 없는 모든 노드에 대해 하위 구조가 존재한다. Figure 5-1(b)에서 리프 노드는 숫자로 표현하였으며, 리프 노드가 아닌 노드(non-leaf node)는 알파벳으로 표현하였다. 이러한 표현 방식에서는 쿼드 트리 블록과 쿼드 트리 노드라는 용어를 같은 의미로 사용한다. 쿼드 트리 블록에서 분할 레벨(partition level)이라는 용어를 사용하게 되며, 분할 레벨은 트리 구조에서 블록의 분할 횟수를 의미한다. 일반적으로 최대 분할

레벨 (트리 구조에서 높이에 대한 제한) 또는 쿼드 트리 블록에 대한 최소 크기를 설정한다. Figure 5-1에 표시된 2차원 쿼드 트리의 경우, 나침반 방향을 사용하여 분할로 인한 4개의 영역 중 하나인 특정 사분면을 표현한다. 예를 들어 Figure 5-1에서 “1”로 표시된 블록은 최상위 노드(root)의 남서(NW) 사분면이다.

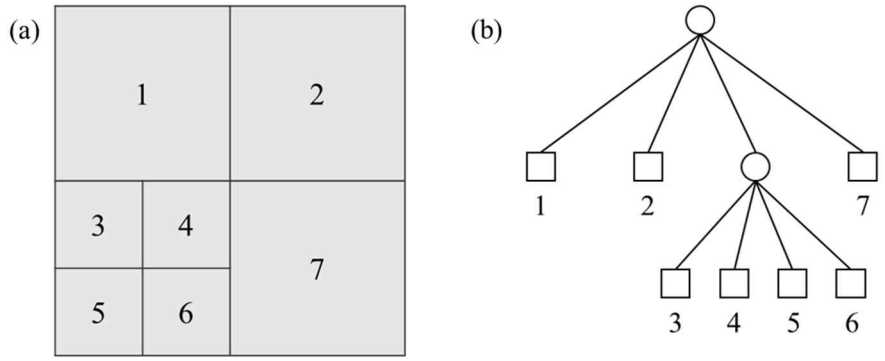

Figure 5-1 (a) The block partitioning and (b) tree structure of a simple quadtree, where leaf blocks are labeled with numbers and non-leaf blocks with letters

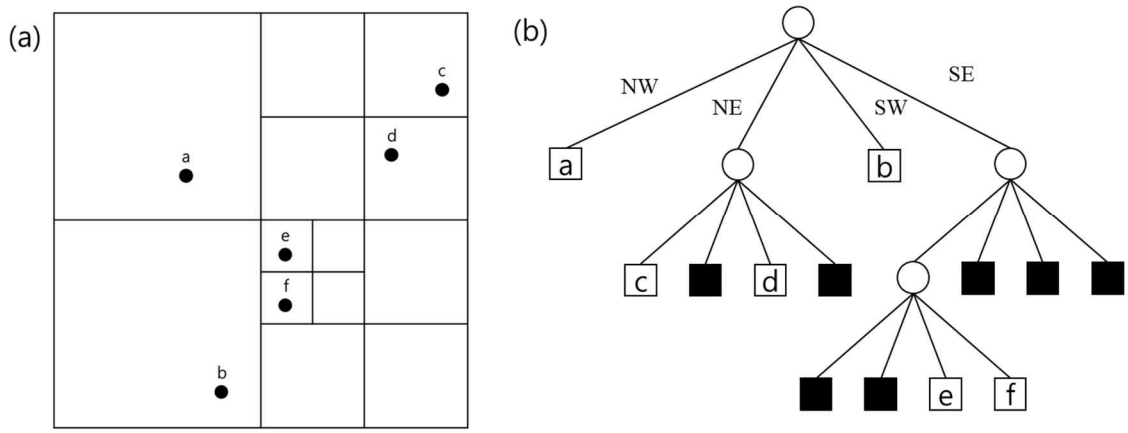

Figure 5-2 (a) Example of PR quadtree partitioning and (b) tree structure of a PR quadtree

노드 분할, 데이터 유형 및 기타 세부 사항의 설정 방법에 따라 다양한 쿼드 트리 방법이 존재한다. 가장 많이 쓰이는 방법 중 하나는 PR(Point Region) 쿼드 트리이며(Samet, 1989), 이 방법은 점 데이터의 수에 따라 쿼드 트리 블록을 분할하는 방법이다. 점은 리프 노드에 저장되며, 분할 규칙은 리프 블록에 두 개 이상의 포인트가 포함된 경우 분할해야 한다. 즉, 각 리프 블록은 하나의 점 이상을 포함하거나 하나도 포함하지 않는다. 분할 조건은 고정 최대 점 c를 설정하고 점 개수가 c 이상일 경우 리프 블록을 분할한다.

PR 쿼드 트리가 점에 기반하여 블록을 분할한다면, PM (Polygon map) 쿼드 트리는 선에 기반하여 블록을 분할한다. 다각형과 같이 연결 관계가 있는 정보는 연결 관계 정보를 저장하기 위해 선 단위로 저장해야 한다. PM 쿼드 트리는 Figure 5-3과 같이 구성된다. PR 쿼드 트리에서 블록을 분할하는 점 개수를 정했듯, PM 쿼드 트리에서도 분할 횟수(splitting threshold)를 정해야 한다. PM 쿼드 트리는 분할 방법에 따라 PM1, PM2, PM3 쿼드 트리가 존재한다. PM1 쿼드 트리가 제일 엄격한 기준으로 블록을 분할하며, PM3는 PR 쿼드 트리와 분할 기준은 동일하며 데이터 저장 구조만 다르다. PM1 쿼드 트리는 1) 하나의 점과 하나 이상의 간선, 2) 최대 하나의 간선으로 기준을 설정하여 블록을 분할한다. PM2 쿼드 트리는 하나의 동일한 점과 하나 이상의 간선으로 기준을 설정하여 블록을 분할한다. PM3 쿼드 트리는 최대 하나의 점으로 기준을 설정하여 블록을 분할한다. Figure 5-3은 PR, PM1, PM2, PM3 쿼드 트리 예시를 보여준다.

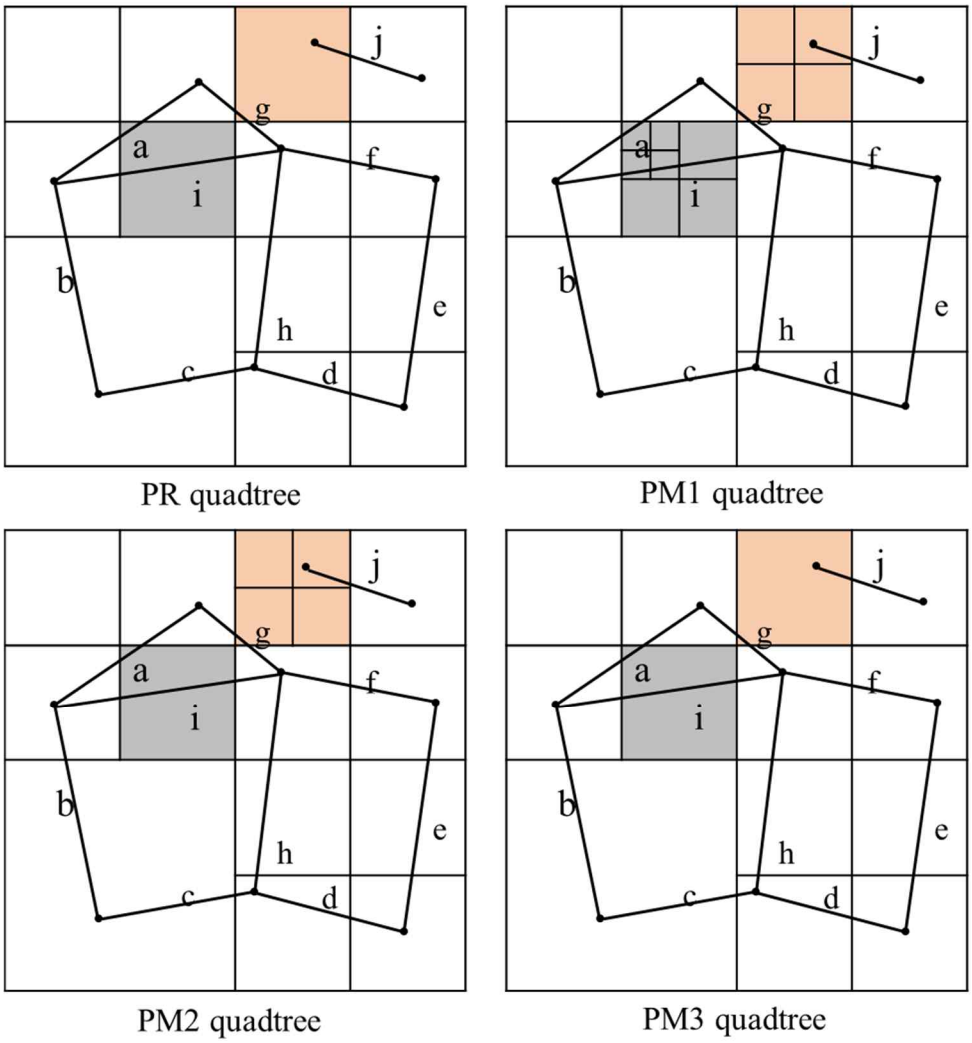

Figure 5-3 An example of PR, PM1, PM2, PM3 quadtree

Figure 5-3에서 회색 블록이 분할된 쿼드 트리는 PM1 쿼드 트리가 유일하다. 회색 블록이 분할되기 전에 가지고 있는 간선은 a와 i 두 개로 PM1 쿼드 트리의 분할 기준을 만족하여 분할된 것을 확인하였다. 주황색 영역은 PM1, PM2 쿼드 트리 모두 분할된 것을 볼 수 있다. 분할되기 전 주황색 블록에는 g와 j가 존재한다. g와 j는 끝점이 다르기 때문에 PM1 쿼드 트리뿐만 아니라 PM2 쿼드 트리의 분할 기준도 만족하여 분할된 것을 확인하였다.

간선이 추가될 때마다 분할이 일어나는 쿼드 트리는 PM1 쿼드 트리가 유일하다. PM1 쿼드 트리를 사용하면 상세하게 트리를 구성할 수 있지만, PM1 쿼드 트리는 다른 쿼드 트리와 달리 많은 메모리를 필요로 한다. 특히, 깊이가 낮게 설정된 쿼드 트리에서 간선이 추가되었을 때 블록과 간선의 교차가 계속 일어나면서 쿼드 트리의 깊이가 크게 증가할 수 있다. 본 논문에서 적용하고자 하는 연안에서는 해안선이 복잡하고 장애물이 많기 때문에 PM1 쿼드 트리를 사용하기 어렵다.

이러한 PM1 쿼드 트리의 단점을 보완한 방법으로 PMR(Polygon Map Random) 쿼드 트리가 있다. PMR 쿼드 트리는 임의의 공간 객체를 저장하기 위한 트리 구조이다. Figure 5-4에서 선 집합에 대한 PMR 쿼드 트리 예시를 보여주며, 공간 분할과 이에 대한 결과 트리 구조를 모두 보여준다. PMR 쿼드 트리는 블록을 분할하여 리프 블록에 객체를 저장하며, 이는 선분이 둘 이상의 리프 블록에 저장될 수 있다. 하위 블록이 없는 블록을 리프 블록이라고 한다. 이를 클리핑(Clipping)이라고 하며, 선분과 교차하는 리프 블록의 영역에 선분을 클리핑하였다고 볼 수 있다. 선분을 포함하는 리프 블록과 교차하는 선분 부분을 $q-object$ 라고 하며, 객체가 선분이므로 여기서는 $q-edges$ 라 부른다. 예를 들어, Figure 5-4의 선분 a는 3개의 리프 블록과 교차할 때 Figure 5-4에 표시된 트리 구조의 리프 노드에서 a는 3개의 쿼드 트리 블록에 저장되어 있다.

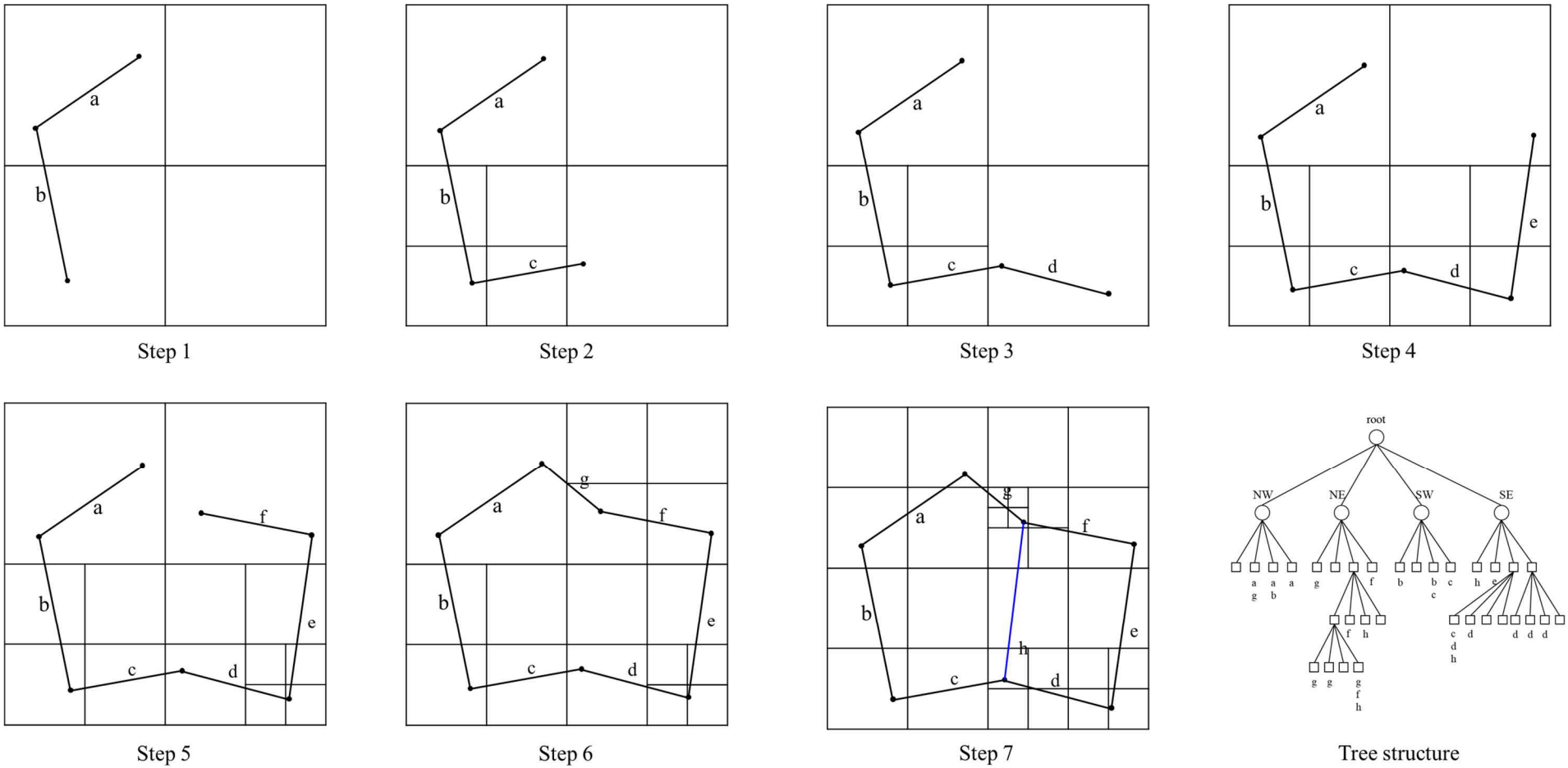

Figure 5-4 A PMR quadtree division for line segments with a splitting threshold of 2 and depth 3 and a tree access structure

PMR 쿼드 트리의 핵심은 분할 횟수로 쿼드 트리 블록이 분할되는 조건을 설정하는 것이다. PMR 쿼드 트리는 이를 위해 사전에 설정한 분할 횟수(Splitting threshold) t를 사용한다. 객체 o의 삽입으로 인해 리프 블록 b의 객체 수가 t를 초과하고 b의 최대 분할 수준이 최대 분할 수준이 아닌 경우 b가 분할되고 b (o 포함)의 객체가 새로 생성된 하위 노드에 삽입된다. 이러한 하위 블록은 t개 이상의 객체를 포함하더라도 현재 상태에서는 더 이상 분할되지 않는다. 따라서 깊이(Depth) D의 리프 노드는 최대 $t + D$ 객체를 포함할 수 있으며 루트는 깊이 0으로 설정된다. 최대 깊이에서 리프 노드의 객체 수에는 제한이 없으며, 과도한 분할을 방지하기 위해 새로 형성된 리프 블록을 즉시 분할하지 않는다. PMR 쿼드 트리의 이러한 속성은 객체가 삽입되는 순서가 트리의 결과에 영향을 준다는 것을 의미하며, 결국 결과가 확률적으로 발생한다는 것을 의미한다. 예를 들어, Figure 5-4에서 선분 h가 선분 g 이후가 아니라 선분 e 다음에 삽입되고 다음에 g가 삽입되면, 루트의 NE 사분면의 SE 사분면의 분할이 추가로 일어난다. 루트의 NE 사분면의 SE 사분면에 e가 있는 상태에서 f가 삽입되면 분할 임계 값인 2를 초과했기 때문에 분할이 일어난다. 하지만, 이러한 경우는 전체 쿼드 트리를 분할하는데 극히 일부분이므로 쿼드 트리를 생성하는데 크게 영향을 주지 않는다.

PM 쿼드 트리에 비해 PMR 쿼드 트리의 장점은 메모리 소모량이 훨씬 적다는 점이다. 앞서 설명했듯, PM 쿼드 트리는 간선이 2개 이상 존재하면 설정한 값에 도달할 때까지 쿼드 트리가 계속 분할된다. 장애물이 적을 때

는 쿼드 트리가 세밀하게 쪼개지므로 PMR 쿼드 트리보다 유용할 수 있다. 하지만, 본 논문의 적용 영역은 해안선이 복잡하고 섬이 많은 연안이므로 PM 쿼드 트리를 적용하면 많은 메모리 소모량이 필요하다. 따라서 PM 쿼드 트리보다 메모리가 훨씬 적게 필요한 PMR 쿼드 트리를 사용하기로 한다.

5.2 PMR quadtree insertion and split algorithm

대부분의 계층적 데이터 구조에 대한 삽입 알고리즘과 마찬가지로 PMR 쿼드 트리 삽입 알고리즘은 쿼드 트리의 하향식 순회(Top-down traversal) 방식으로 진행된다. 즉, 최상위 노드에서 객체를 삽입한 후, 객체와 교차하는 리프 노드들을 찾고 모든 리프 노드에 객체를 추가한다. 객체와 리프 노드가 교차하는지 판별하는 과정이 시간이 많이 필요한 부분이다. 이러한 과정은 객체를 삽입하는 데 필요한 계산 시간은 리프 노드의 깊이에 거의 비례한다. 이러한 계산 시간을 줄이기 위해 쿼드 트리 분할이 반복적으로 이루어진다는 점을 이용한다. 반복적인 쿼드 트리 분할을 이용하여 탐색 과정을 짧은 시간 내에 수행할 수 있으며, 이 경우 계산 시간이 객체와 교차하는 리프 노드 수에 거의 비례한다. 이 과정에서 핵심적인 부분은 객체의 형상만을 기반으로 객체를 최소한으로 둘러싸는 쿼드 트리 블록을 계산할 수 있다는 점이다.

Figure 5-5(a)에서 볼 수 있듯, 객체를 최소한으로 둘러싸는 쿼드 트리

블록은 쿼드 트리 구조를 이용하여 찾아낼 수 있다. 여기서 점선은 기존에 생성된 쿼드 트리 블록을 의미한다. 이 과정을 이용하면 쿼드 트리 분할 과정을 최상위 노드가 아닌 하위 노드 수준에서 시작할 수 있다. 찾아낸 쿼드 트리 블록이 리프 노드가 아닐 경우를 Figure 5-5(b)에, 리프 노드일 경우를 Figure 5-5(c)에 나타냈다. 일반적으로 객체는 기존의 쿼드 트리 블록에 의해 완전히 둘러싸이는 경우가 많으므로 객체를 둘러싸는 쿼드 트리 블록은 기존 리프 노드 내부에 있거나 기존 리프 노드와 일치한다.

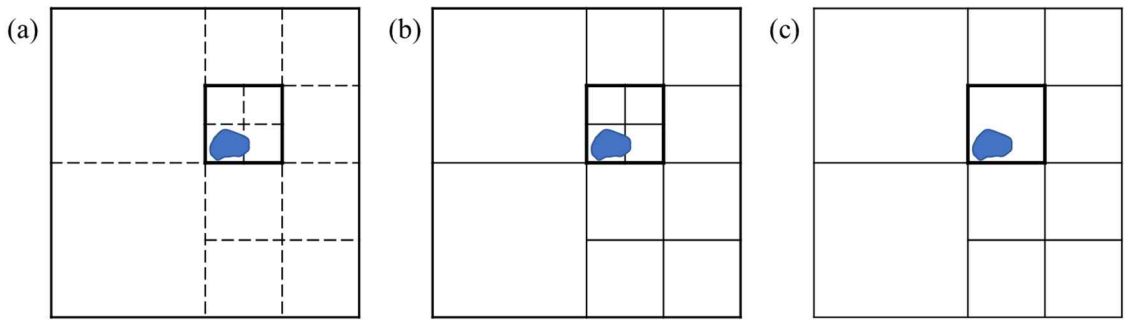

Figure 5-5 Computation of the minimum bounding block in case of leaf node and non-leaf node.

PMR 쿼드 트리에 선 집합을 삽입하는 알고리즘을 Figure 5-6, Figure 5-7, Figure 5-8에 나타냈다. 함수 *InsertObject* 는 *ComputeEnclosingBlock* 및 *FindEnclosingNode* 함수를 사용하여 객체를 포함하는 작은 노드를 찾고 해당 노드에서 *Insert* 함수를 호출한다. 이 과정은 삽입을 최상위 노드가 아닌 중간 노드에서부터 시작하기 위해 수행된다. 최악의 경우는 객체가 최상위 노드의 경계에 걸쳐 있어 최소한으로 둘러싸는 쿼드 트리 노드가 쿼드 트리의 최상위 노드가 되는 경우이다.

Algorithm 1 Edge insertion.

```
1: procedure InsertObject(edge)
2:     enclosingBlock ← ComputeEnclosingBlock(edge)
3:     node ← FindEnclosingNode(enclosingBlock)
4:     Insert(node, edge)
5:return
6:end procedure
```

Figure 5-6 Edge insertion of PMR quadtree

Algorithm 2 Edge insertion in leaf node.

```
1: procedure Insert(n_i^lv, object)
2:     if n_i^lv is leaf node then
3:         n_i^lv ← add edge to leaf node
4:         if (edge count in leaf node > threshold) then
5:             Split(n_i^lv)
6:         end if
7:     else
8:         foreach childNode of node do
9:             if (Intersects(n_i^lv.children, object)) then
10:                Insert(n_i^lv.children, object)
11:            end if
12:        end for
13:    end if
14:return
15:end procedure
```

Figure 5-7 Edge insertion in leaf node of PMR quadtree

Algorithm 3 Split a node.

```
1: procedure Split(node_i^d)
2:     edgeList ← node_i^d.edges
3:     if node_i^d.children ≠ ∅
4:         return null
5:     end if
6:     if d ≥ maxDepth
7:         return null
8:     end if
9:     node_i^d.children ← {node_i^d.NW, node_i^d.NE, node_i^d.SW, node_i^d.SE}
10:    foreach dir in {NW, NE, SW, SE}
11:        node_i^d.children[dir].bound is set with node_i^d.bound
12:    end for
13:    foreach edge in edgeList do
14:        foreach dir in {NW, NE, SW, SE}
15:            if intersection(edge, node_i^d.children[dir])
16:                node_i^d.children[dir].edges ← edge
17:            end if
18:        end for
19:    end for
20: end procedure
```

Figure 5-8 Split algorithm in leaf node of PMR quadtree

객체를 포함하는 가장 작은 쿼드 트리 블록을 찾는 과정은 *ComputeBoudningBlock* 함수와 *FindBoundingNode* 함수를 이용하여 진행된다. 첫 번째 함수 *ComputeEnclosingBlock* 는 객체의 형상을 이용하여 최소로 둘러싸는 쿼드 트리 블록을 계산하며, 두 번째 함수 *FindBoundingNode* 는 실제 쿼드 트리 노드를 찾기 위해 쿼드 트리 구조를 이용한다. *Split* 함수는 리프 노드를 분할하며, 해당 노드를 리프 노드가 아닌 노드로 설정한 후 객체를 하위 노드에 다시 삽입한다. 이 때 분할 임계 값을 초과하더라도 자식 노드가 분할되지 않는지 확인하면서 진행한다. *Split* 함수의 첫 번째 반복문은 하위 노드 중 그 블록에 완전히 포함된 선을 찾고 이 선을 최소로 둘러싸는 쿼드 트리 블록을 사용한다. 이후 두 번째 루프는 *objList* 에 남아있는 객

체, 즉 둘 이상의 하위 노드와 교차하거나 노드 자체에 완전히 둘러싸여 있지 않은 개체를 다시 삽입한다.

PMR 쿼드 트리는 주어진 환경 M을 그래프 형태인 $Q=(N,E)$로 변환하기 위해 사용된다. N은 쿼드 트리 노드의 집합, E는 간선의 집합이다. 쿼드 트리 노드는 다음과 같이 표현된다.

$$node_i^d \in N \qquad (5-1)$$

where

i: the index of the node

d: the dpeth of this node

쿼드 트리에서 최대 깊이 (maximum depth)는 $maxDepth$로 정의하며 다음 관계를 가진다. 첫 번째 쿼드 트리는 ($d=0$) 오직 하나의 노드($node_1^0$)를 가지며 이 노드를 트리 구조의 최상위 노드(root)라고 부른다. 최상위 노드는 $node^0$이라 표기한다.

$$0 \le d \le maxDepth \qquad (5-2)$$

where

$maxDepth$: the maximum depth

이러한 표현을 이용하여 쿼드 트리 노드 $node_i^d$가 가져야 하는 특성 ($node_i^d.property$)은 다음과 같다.

a) 상위 노드 ($node_i^d.parent$)는 쿼드 트리 구조에서 이전 깊이의 노드를

의미한다. 상위 노드가 존재하지 않으면 그 노드는 트리 구조의 최상위 노드(root)이다.

$$node_{i}^{d}.parent = node_{j}^{d-1} \quad (5-3)$$

$$node_{i}^{d}.parent = null \rightarrow node_{i}^{d} = node^{0} \quad (5-4)$$

where

$node_{i}^{d}$: a node of index i in the depth d

$node_{j}^{d-1}$: a node of index j in the depth $d-1$

$node^{0} = node^{R}$: a root of quadtree node

b) 하위 노드($node_{i}^{d}.children$)는 상위 노드의 위치를 4방향(북서, 북동, 남동, 남서)으로 분할하여 저장된다. 탐색한 노드가 리프 노드일 경우, 하위 노드는 존재하지 않는다.

$$node_{i}^{d}.children = \begin{cases} \{node_{NW}^{d+1}, node_{NE}^{d+1}, node_{SE}^{d+1}, node_{SW}^{d+1}\} & if\ node_{i}^{d}\ is\ not\ a\ leaf\ node \\ \emptyset & if\ node_{i}^{d}\ is\ a\ leaf\ node \end{cases} \quad (5-5)$$

where

$node_{i}^{d}.children$: a childeren of node of index i in the depth d

$node_{NW}^{d+1}$: a north west children of the node

$node_{NE}^{d+1}$: a north east children of the node

$node_{SE}^{d+1}$: a south east children of the node

$node_{SW}^{d+1}$: a south west children of the node

c) 쿼드 트리 경계($node_i^d.bound$)는 쿼드 트리가 차지하고 있는 영역을 의미한다. n_i^d의 경계는 상위 노드 n_j^{d-1}의 경계의 1/4 크기이다.

$$size\ (node_i^d.bound) = \left(\frac{1}{4}\right) \cdot size\ (node_j^{d-1}.bound) \quad (5-6)$$

where

$node_i^d.bound$: a bound of the node of index i in the depth d

$node_j^{d-1}$: a bound of the node of index j in the depth $d-1$

d) 현재 노드가 리프 노드인지 판별하여 ($node_i^d.isLeaf$) 저장한다. 만약, 현재 노드의 하위 노드가 존재하지 않는다면 n_i^d는 리프 노드이다.

$$node_i^d.isLeaf = \begin{cases} true & if\ node_i^d.children = \emptyset \\ false & if\ node_i^d.children \neq \emptyset \end{cases} \quad (5-7)$$

where

$node_i^d.isLeaf$: a boolean proerty of leaf node node $node_i^d$

$node_i^d.children$: a children of leaf node $node_i^d$

e) 현재 노드가 통항 가능한 지역($node_i^d.passable$)인지 판단하고 저장한다. 쿼드 트리 노드가 통항 가능한지 판단하기 위한 방법은 여러 가지가 있다. 첫 번째 방법은 쿼드 트리 노드의 경계 안에 장애물의 일부분이 겹치면 통항 불가능이라 판단하는 방법이다. 장애물과 겹칠 때 쿼드 트리 블록이 막혀 있다고 정의한다. 이 방법은 매우 간단하지만 크기가 큰 쿼드 트리 노드

의 일부분에 장애물이 있어 그 노드가 통항 불가능하게 되면, 최적 경로가 있음에도 경로를 구할 수 없는 경우가 존재할 수 있다. 두 번째 방법은 쿼드 트리 노드의 경계 안에 장애물의 밀집도를 계산하여 통항 여부를 판단하는 방법이다. 이 방법은 쿼드 트리 노드 통항 여부를 밀집도에 따라 결정하기 때문에 합리적인 방법이지만, 계산 시간이 많고 통항 여부를 판별하는 밀집도 값을 계산해야 하기 때문에 적용하기 어렵다. 세 번째 방법은 쿼드 트리 노드의 가운데 점을 기준으로 통항 여부를 판단하는 방법이 있다. Figure 5-9(a)와 같이 가운데 점이 장애물 안에 있으면, 가운데 점은 통항 불가능(passable = false)이 되며, Figure 5-9(b)와 같이 가운데 점이 장애물 밖에 있으면 통항 가능(passable = true)이 된다. 리프 노드가 경로 탐색에 사용되기 때문에 리프 노드만 통항 가능 여부를 판별한다.

$$node_i^d.passable = \begin{cases} true & if\ isBlocked(node_i^d) = false \\ false & if\ isBlocked(node_i^d) = true \end{cases} \quad (5-8)$$

where

$isBlocked(n_i^p)$: a boolean proerty whether current node n_i^p is block or not

$node_i^p.isLeaf$: a boolean proerty of current node n_i^p

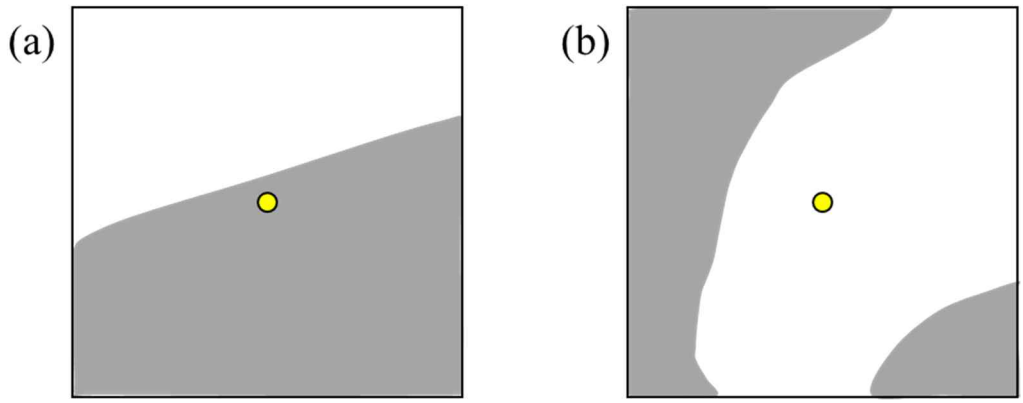

Figure 5-9 Example of passable property((a) blocked, (b) unblocked)

g) 현재 노드에 포함된 간선($n_i^b.edges$)을 저장한다. PMR 쿼드 트리를 사용하면 장애물의 간선이 분할되어 저장된다. 통항 가능 여부와 마찬가지로 리프 노드가 경로 탐색에 사용되기 때문에 리프 노드만 통항 가능 여부를 판별한다.

$$node_i^d.edges = \begin{cases} \{e_1, e_2, \ldots, e_k\} & if \quad node_i^b.isLeaf = true \\ \emptyset & if \quad node_i^b.isLeaf = false \end{cases} \qquad (5-9)$$

where

e_k: the edge of obstacles

k: the number of edges

정리하면 쿼드 트리 생성 알고리즘은 영역 내 장애물에 따라 쿼드 트리를 세분화한다. 쿼드 트리는 쿼드 트리 노드의 경계 내에 장애물의 간선이 일정 수 이상 존재하거나 쿼드 트리 노드가 최대 레벨($maxLevel$)에 도달할 때까지 분할된다. 이에 대한 분할 조건을 다음과 같이 표현할 수 있다.

$$Split\left(node_i^v\right) = true \quad if \; E_i < threshold \quad and \quad v < maxLevel \qquad (5-10)$$

where

E_i : the set of edge in quadtree node

$threshold$: Splitting threshold

본 연구에서 쿼드 트리는 탐색 영역 내 존재하는 장애물의 간선의 수에 따라 분할된다. 탐색 장애물이 많은 영역을 나타내는 쿼드 트리 노드는 통항할 수 없도록 설정되며, 경로 탐색 시 해당 노드를 탐색할 수 없다. 예를 들어, 쿼드 트리 노드 내 지형의 일부가 존재하면 그 노드를 분할하며 최대 깊이에 도달한 노드는 분할되지 않고 통항할 수 없도록 설정된다. 반대로, 장애물이 거의 없는 영역은 노드가 분할되지 않고 통항할 수 있도록 설정된다. (De Berg, Van Kreveld, Overmars, & Schwarzkopf, 1997)에 따르면 모든 쿼드 트리 노드를 n이라 할 때, PMR 쿼드 트리 구성을 위한 시간 복잡도(Time complexity)는 $O(n\log n)$이다.

5.3 Graph construction using PMR quadtree

경로 탐색 알고리즘은 환경을 모델링한 맵을 그래프로 변환한후 그래프 내 노드 간 연결 관계를 기반으로 수행된다. 여기서 그래프는 탐색 공간(search space or configuration space)이 되며 이 탐색 공간을 구성하기 위해 쿼드 트리의 리프 노드 ($node_i^v.isLeaf = true$)가 서로 연결된다. 아래 그림은 하나의 리프 노드에 대한 다른 리프 노드의 연결 관계를 보여준다.

아래 그림 (a)는 하향식 (Top-down) 관점에서 리프 노드 간 연결을 보여주며, 쿼드 트리 구조를 불규칙한 격자로 표현한다. 그림 (a)에서 어두운 쿼드 트리 노드는 현재 연결 관계를 구성하는 노드이며 직선은 연결 관계를 보여준다. 그림 (b)는 쿼드 트리의 구조를 보여주며 화살표는 Figure 5-10(a)의 동일한 노드와 연결 관계를 나타낸다. Figure 5-10(b)에서는 트리 구조의 여러 단계에서 리프 노드 간 연결 관계를 구성할 수 있다.

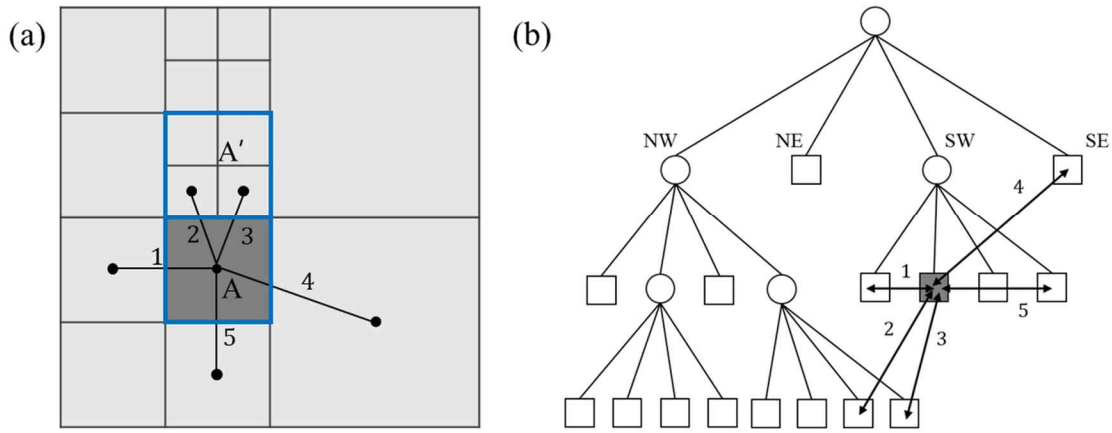

Figure 5-10 Quadtree-node neighborhood connections: Connections in (a) quadtree structure and (b) tree structure.

쿼드 트리에서 인접한 노드는 쿼드 트리 구조를 구성하는 동안 결정된다. 인접한 노드를 결정할 때 4방향, 8방향, 16방향 등으로 인접한 노드를 결정할 수 있다. 16방향 이상으로 인접한 노드를 연결하게 되면, 경로의 품질이 좋아지지만 메모리 소모량과 계산 시간이 증가하게 된다. 따라서, 4방향과 8방향으로 연결하는 방법이 효율적이다. 8방향으로 연결하면, 대각선 노드가 연결된다. 대각선 노드를 연결하면 대각선 방향으로 한 개의 점만 연결되는데 이는 경로가 일직선에 가까워지는 효과가 발생한다. 이와 같은 효과

는 가시성 그래프를 연결할 때 탐색 공간을 줄이게 되므로 가시성 그래프에서 찾은 경로의 품질을 낮출 수 있다. 가시성 그래프는 점 개수에 따라 간선이 $n \cdot (n-1)/2$개만큼 생기므로 4방향으로 표현했을 때의 가시성 그래프에서 간선의 수는 8방향으로 표현했을 때의 수보다 훨씬 많다. 가시성 그래프에서 탐색한 해의 품질을 높이기 위해서는 4방향으로 연결하는 방법이 더 적절하다고 판단하였다.

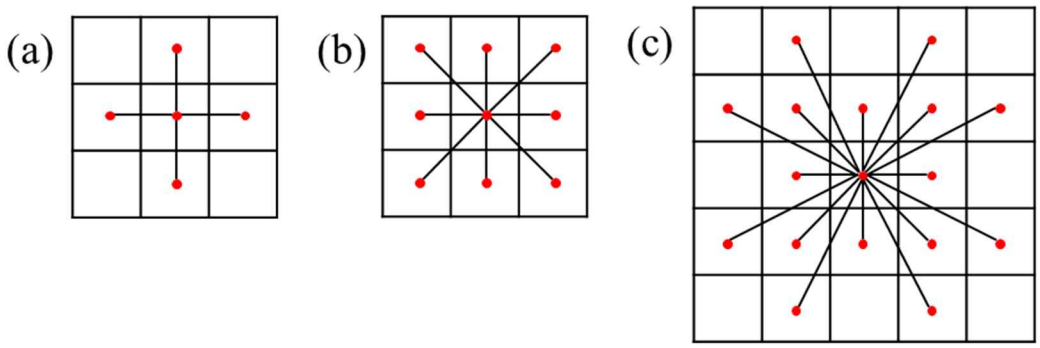

Figure 5-11 Node connection in 4,8,16 directions

노드를 연결하는 과정을 다음의 예시로 설명할 수 있다. 쿼드 트리 노드에서 북쪽 방향으로 인접한 노드를 찾는 과정은 Figure 5-13에서 제시된 의사 코드로 설명할 수 있다. 첫 번째 단계는 선택한 노드의 북쪽 방향으로 인접한 노드를 찾는 것이다. 선택한 노드의 중심에서 북쪽 방향으로 인접한 노드를 찾기 위해 미러링 방법을 사용한다(Figure 5-13의 7행 ~ 14행). 미러링된 위치를 사용하면 최상위 노드(n^r)에서 쿼드 트리 노드를 재귀적으로 검색하여 그 노드를 포함하는 쿼드 트리 노드를 확인할 수 있다(Figure 5-13의 18행). 이 노드를 대칭 노드($mirror\ Node_i^b$)라 부른다. 아래 Figure 5-12에서 대칭 노드의 예시를 볼 수 있다. A의 대칭 노드는 A'

이며, B의 대칭 노드는 B’ 이다.

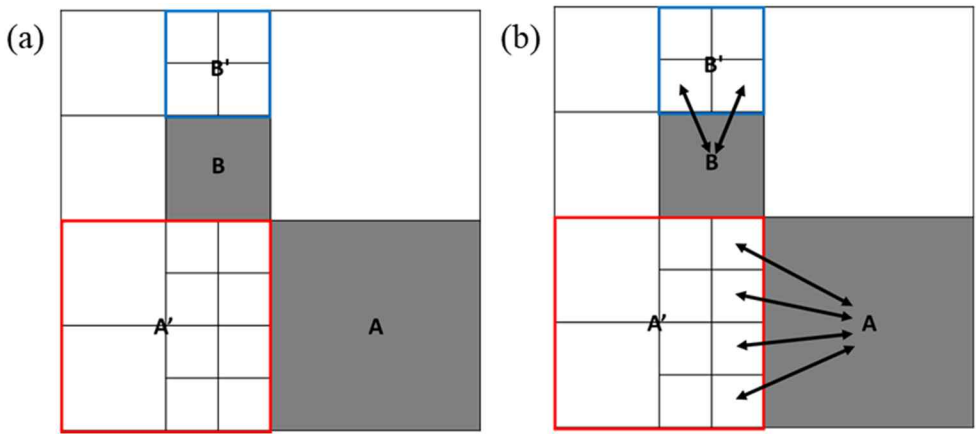

Figure 5-12 The example of symmetric node A and B

대칭 노드의 하위 노드가 없을 경우에 이 노드는 리프 노드이며, 이 노드가 선택한 노드의 북쪽 방향에서 유일하게 인접한 노드이다(Figure 5-13의 22, 23행). 하위 노드가 있을 경우 대칭 노드의 하위 노드를 탐색해야 한다(Figure 5-13의 25행). 대칭 노드의 하위 노드를 분석하는 방법은 그림 x에 설명하였다. 이 방법은 선택한 노드의 북쪽 경계 ($mirrorNode\ e_i^b$의 남쪽 경계)에 있는 대칭 노드에서 모든 하위 노드를 재귀적으로 탐색한다. 그런 다음 리스트 형식으로 북쪽 노드의 하위 노드를 반환한다(Figure 5-13의 9행에서 11 행까지). 북쪽 방향과 동일하게 남쪽, 서쪽, 동쪽 방향에 대해 동일하게 반복하여 4방향으로 인접한 노드를 찾아낸다.

Algorithm 4 Method that finds the neighbors of a node.

1: n_i^{lv}: the quadtree node
2: dir: the search direction
3: **procedure** GetNeighbors(n_i^{lv}, dir)
4: **if** n_i^{lv}==n^R **then**
5: **return** $null$
6: **end if**
7: **if** dir is $north$ **then**
8: $mPos \leftarrow$ mirrored position from the center of $n_i^{lv}.bound$ in the north direction
9: **else if** dir is $south$ **then**
10: $mPos \leftarrow$ mirrored position from the center of $n_i^{lv}.bound$ in the south direction
11: **else if** dir is $east$ **then**
12: $mPos \leftarrow$ mirrored position from the center of $n_i^{lv}.bound$ in the east direction
13: **else if** dir is $west$ **then**
14: $mPos \leftarrow$ mirrored position from the center of $n_i^{lv}.bound$ in the west direction
15: **else**
16: **return** $null$
17: **end if**
18: $mirrorNode_i^{lv} \leftarrow$ FindNodeWithPos($mPos, n^R$)
19: **if** $mirrorNode_i^{lv}$ is $null$ **then**
20: **return** $null$
21: **end if**
22: **if** ($n_i^{lv}.isLeaf$) and ($mirrorNode_i^{lv}.passable$ is $true$) **then**
23: **return** $mirrorNode_i^{lv}$
24: **else**
25: **return** FindFrontier($mirrorNode_i^{lv}, dir$)
26: **end if**
27:**end procedure**

Figure 5-13 One neighbor node search algorithm

Algorithm 5 All neighbors search in a frontier of a quadtree node.

1: $mirrorNode_i^{lv}$: the mirror node from the one being analyzed
2: dir: the search direction
3: **procedure** FindFrontier(n_i^{lv}, dir)
4: $frontierNeigbor \leftarrow$ list with the neighbors in the frontier of the node being analyzed
5: **if** $mirrorNode_i^{lv}.passable$ is $true$ **then**
6: **if** $mirrorNode_i^{lv}.isLeaf$ is $true$ **then**
7: **return** $null$
8: **else**
9: **if** dir is $north$ **then**
10: Add FindFrontier($mirrorNode_i^{lv}.children(n_{SE}^{lv+1}), dir$) into $frontierNeigbor$
11: Add FindFrontier($mirrorNode_i^{lv}.children(n_{SW}^{lv+1}), dir$) into $frontierNeigbor$
12: **else if** dir is $south$ **then**
13: Add FindFrontier($mirrorNode_i^{lv}.children(n_{NE}^{lv+1}), dir$) into $frontierNeigbor$
14: Add FindFrontier($mirrorNode_i^{lv}.children(n_{NW}^{lv+1}), dir$) into $frontierNeigbor$
15: **else if** dir is $east$ **then**
16: Add FindFrontier($mirrorNode_i^{lv}.children(n_{SW}^{lv+1}), dir$) into $frontierNeigbor$
17: Add FindFrontier($mirrorNode_i^{lv}.children(n_{NW}^{lv+1}), dir$) into $frontierNeigbor$
18: **else if** dir is $west$ **then**
19: Add FindFrontier($mirrorNode_i^{lv}.children(n_{NE}^{lv+1}), dir$) into $frontierNeigbor$
20: Add FindFrontier($mirrorNode_i^{lv}.children(n_{NE}^{lv+1}), dir$) into $frontierNeigbor$
21: **end if**
22: **end if**
23: **return** $frontierNeigbor$
24: **end procedure**

Figure 5-14 All neighbor nodes search algorithm

쿼드 트리 노드를 연결하는 방법은 Figure 5-15과 같이 진행된다. 이 예제는 쿼드 트리 노드 중 A의 서쪽 방향으로 이웃한 노드들을 찾는 과정이며, 먼저 Figure 5-13의 알고리즘을 이용하여 A의 대칭점을 계산하고 대칭 노드($A' = node_i^d$)를 찾는다. 대칭 노드가 리프 노드가 아니기 때문에 Figure 5-14의 알고리즘을 이용하여 하위 노드를 찾는다. 하위 노드를 찾는 알고리즘은 반복적으로 수행되기 때문에 하위 노드를 찾기 위해 Figure 5-15의 2단계와 같이 알고리즘을 적용한다. $node_i^d$의 하위 노드는 $node_{NW}^{d+1}$, $node_{NE}^{d+1}$, $node_{SE}^{d+1}$, $node_{SW}^{d+1}$가 있으며, Figure 5-15에서 볼 수 있듯 NW와 SW 방향 하위 노드는 A의 서쪽 경계면에 인접하지 않으므로 고려할 대상

이 아니다. NE 방향 하위 노드인 $node_{NW}^{d+1}$는 리프 노드이기 때문에 A와 연결하는 노드이며, SE 방향 하위 노드인 $node_{SW}^{d+1}$는 리프 노드가 아니기 때문에 Figure 5-14의 알고리즘이 다시 수행된다. 편의를 위해 $node_{j}^{d+1} = node_{i}^{d}.node_{SE}^{d+1}$라 표현한다. 2단계와 동일하게 $node_{j}^{d+1}$에는 NW와 SW 방향 하위 노드는 A의 서쪽 경계면에 인접하지 않으므로 인접한 노드가 아니다. NE 방향 하위 노드인 $node_{NE}^{d+2}$와 SE 방향 하위 노드인 $node_{SE}^{d+2}$ 는 리프 노드이기 때문에 A와 연결하는 노드가 된다. 따라서, A의 동쪽 방향으로 인접한 노드는 Figure 5-15의 세 번째 단계와 같이 총 세 개의 노드가 존재한다. 이와 동일한 방법으로 A의 북쪽, 동쪽, 남쪽의 인접한 노드를 계산할 수 있다.

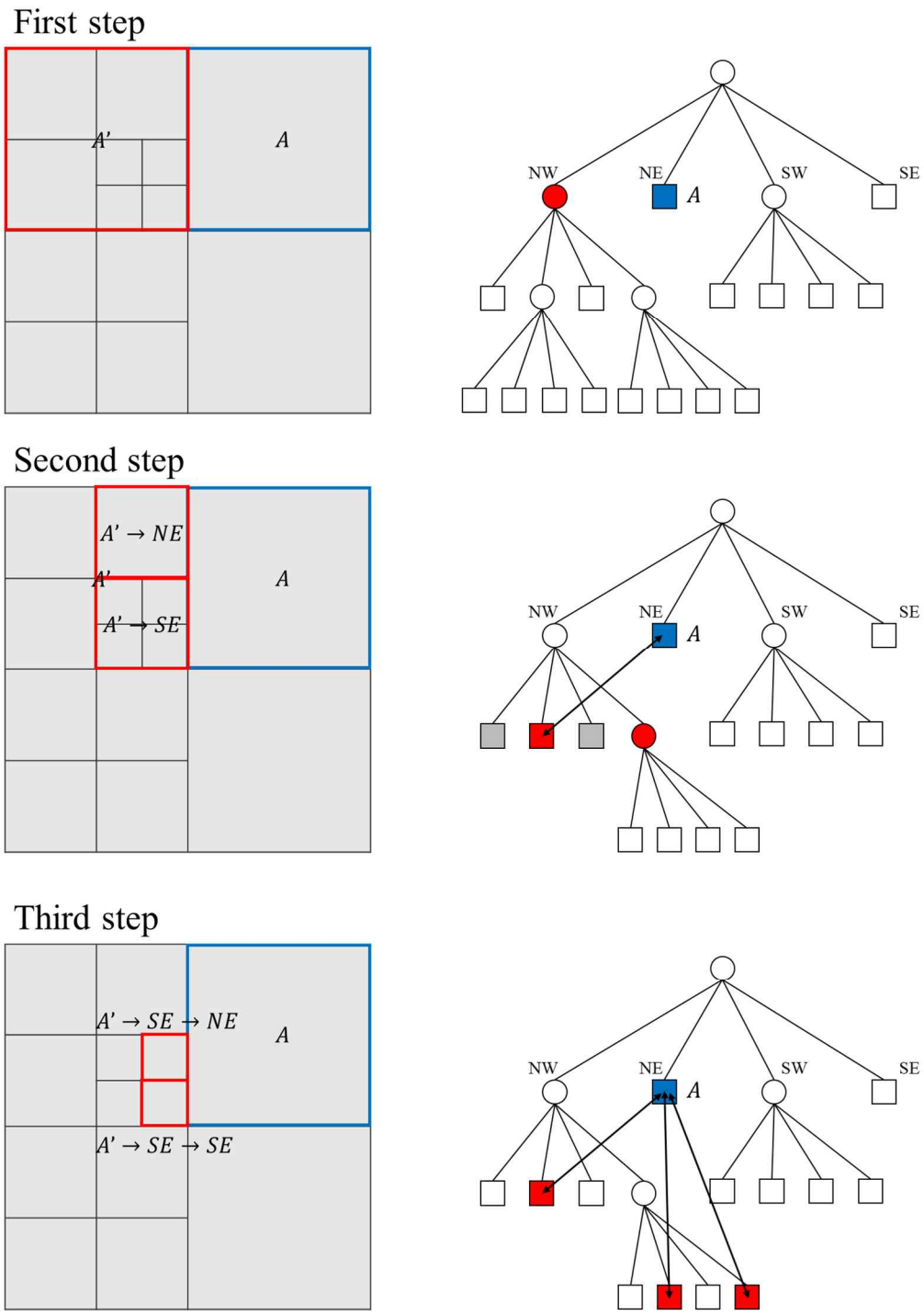

Figure 5-15 The example of connection between quadtree nodes

Figure 5-15의 모든 쿼드 트리 노드에 대해 인접한 노드를 연결할 때 각 노드간 통항가능 여부에 따라 연결 여부가 결정된다. 예를 들어, 연결하려 하는 두 노드 중 하나라도 통항 불가능이면($node_i^d.passable = false$) 두 노드는 연결되지 않는다. 노드의 통항가능 여부에 따라 연결 가능한 모든 노드를 Figure 5-16에 표현하였다. 검은색으로 표시한 블록은 통항이 불가능한 블록이다.

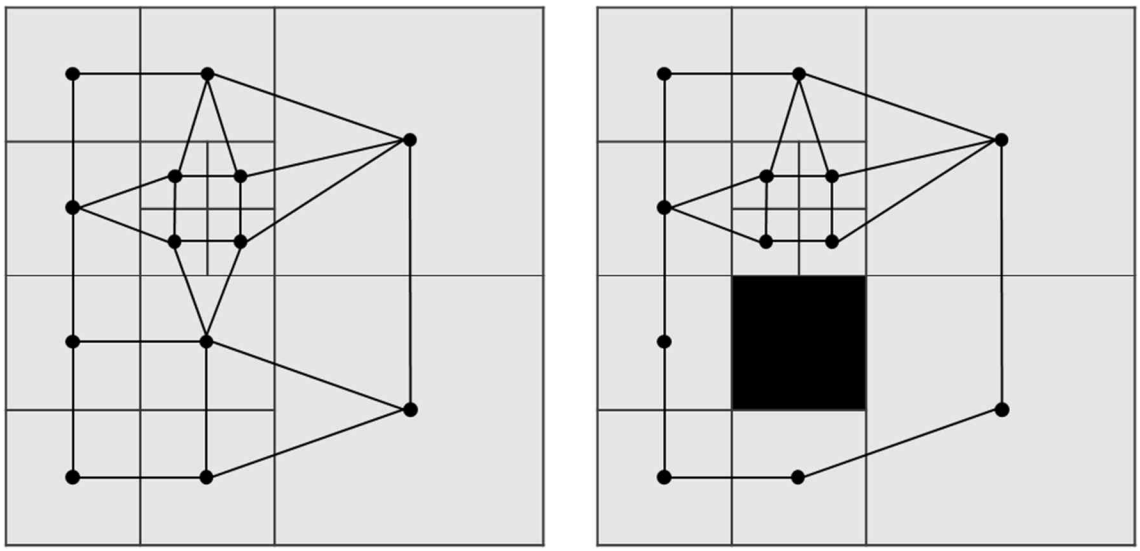

Figure 5-16 The example of all connection between quadtree nodes

각 노드간 연결 관계를 알고 있으므로, 이를 그래프로 변환할 수 있으며, 이 때 그래프는 다음과 같이 표현할 수 있다.

$$QG = (QV, QE) \tag{5-11}$$

where

QG: Quadtree graph

QV: Vertex in quadtree graph

QE: Edge in quadtree graph

5.4 The shortest path on quadtree based graph

쿼드 트리 그래프 QG에서 최적 경로를 찾는다. 이 때 사용한 최적 경로는 가시성 그래프를 생성하는 입력 정보가 된다. 그래프에서 최적 경로를 찾는 알고리즘은 Dijkstra 알고리즘을 사용하였다.

Algorithm 6 Dijkstra algorithm

```
1: V: Vertex set
2: Q: Priority queue
3: Procedure Dijkstra(Graph, start):
4:     foreach v in V
5:         dista[v] = Infinity
6:         prev[v] = None
7:         add (v,dist[v]) to Q
8:     end for
9:     while Q is not empty
10:        u ← Q.pop()
11:        foreach neighbor v of u
12:            if dist[v] > dist[u] + distance(u, v)
13:                dist[v] ← dist[u] + distance(u, v)
14:                prev[u] ← u
15:                Q.push(v, dist[v])
16:            end
17:        end for
18:    end while
19:    return dist, prev
20: end Procedure
```

Figure 5-17 Dijkstra algorithm

쿼드 트리 그래프에서 Dijkstra 알고리즘을 수행하는 이유는 최적 경로로 사용하기 위해서가 아니라 탐색 공간(search space)을 줄이기 위해서이다. Dijkstra 알고리즘으로 인해 찾은 해에 해당하는 쿼드 트리 블록이 탐색 영역(Configuration space)이 되며, 이는 장애물이 많고 복잡한 해양환경에서 계산 시간을 줄이는 역할을 한다.

이 과정은 가시성 그래프에서 탐색 공간을 축소하여 그래프를 효율적으로 생성할 수 있도록 한다. 가시성 그래프의 시간 복잡도는 별도의 최적화 과정 없이 생성할 때 $O(n^3)$이다. 여기서 n은 장애물에 존재하는 모든 점이다. 가시성 그래프의 생성 시간은 n에 따라 아주 크게 증가하는데, 점의 개수가 1,000개만 있어도 10억번의 연산이 필요하다. 본 논문의 적용 영역인 연안에서는 점의 개수가 상당히 많기 때문에 그대로 적용하기 어렵다. 가시성 그래프를 생성하기 위해 필요한 점들만 선택한다면, 해의 최적성을 잃지 않고 계산 시간을 단축할 수 있다. 따라서, 가시성 그래프를 생성하기 위한 점들을 선별하기 위해 쿼드 트리 그래프를 구성하고 이를 이용하여 가시성 그래프를 생성한다.

6 Visibility graph construction with quadtree

6.1 Visibility graph

가시성 그래프(Visibility graph)는 평면에서 장애물의 모든 점 중 두 점을 선택하고 이 선분이 가시성(Visibility)을 지닌다면 그 선분을 연결하여 생성한 그래프이다. 가시성 그래프에서 간선은 간선의 두 점이 연결되어 있다는 의미이며, 두 점 사이에 장애물이 없다는 것을 의미한다. 가시성 그래프는 기하 문제와 경로 계획 문제 등에서 A*, Dijkstra 알고리즘과 함께 최단 거리 경로를 계산하는데 사용되었다.

가시성 그래프는 주어진 환경에서 최적 경로를 찾을 수 있는 방법이지만, 매우 제한적인 상황에서 사용해야 한다. 이는 가시성 그래프를 생성하기 위해 걸리는 시간 때문이다. 주어진 환경의 모든 점 중 두 점을 선택하여 연결 여부를 결정해야 하며, 이 과정에서 시간 복잡도가 $O(n^2)$이다(Kapoor and Maheshwari, 2000). 연결 여부를 판별하기 위해 환경 내 모든 장애물과 간섭 여부를 확인해야 하는데 이 과정에서 시간 복잡도가 $O(n)$이다. 따라서 별도의 최적화 과정 없이 가시성 그래프를 사용하면, 시간 복잡도는 $O(n^3)$이 된다. 예를 들어, 주어진 환경에 1,000개의 점이 있다면 연산은 약 10^9(10억) 회 수행된다. Figure 6-1, Figure 6-2는 가시성 그래프를 생성하는 코드이다. 환경에 장애물이 많을수록 가시성 그래프를 생성하는데 연산 횟수가 기하급수적으로 올라가기 때문에 장애물이 적은 환경에서 로봇의 경로 계획에 사용되었다.

Algorithm 7 VisibilityGraphGeneration

```
1: V: all vertices of obstacles in given environment
2: E: the line segment with visibility
3: procedure GenerateVisibilityGraph(V, E)
4:     for each i = 1 : length(V)
5:         for each j = i : length(V)
6:             if GeneralLineOfSight(line(V(i),V(j)), O)
7:                 continue
8:             else
9:                 E(i, j) = distance(V(i), V(j))
10:            end if
11:        end for
12:    end for
13:end procedure
```

Figure 6-1 Visibility graph generation algorithm

Algorithm 8 GeneralLineOfSight

```
1: O: all obstacles in environment
2: line: the line segment
3: minVector, maxVector: minmax box of all obstacle
3: procedure GeneralLineOfSight(line, O)
4:     foreach i = 1 : length(O)
5:         minPoint ← minVector(O(i))
6:         maxPoint ← maxVector(O(i))
7:         if line is intersect with box(minPoint, maxPoint)
8:             foreach p_i, p_{i+1} in O(i)
9:                 if line is intersect with p_i p_{i+1}
10:                    return false
11:                end if
12:            end for
13:        end if
14:    end for
15:return true
16:end procedure
```

Figure 6-2 General line of sight algorithm

가시성 그래프 구성 예시와 가시성 그래프 위에서 최단 경로를 탐색한 예시를 Figure 6-3에 나타냈다. 만약, 가시성 그래프를 생성할 때 가시성 선분이 Figure 6-4와 같이 장애물과 충돌한다면 두 점을 연결하지 않고 제거한다.

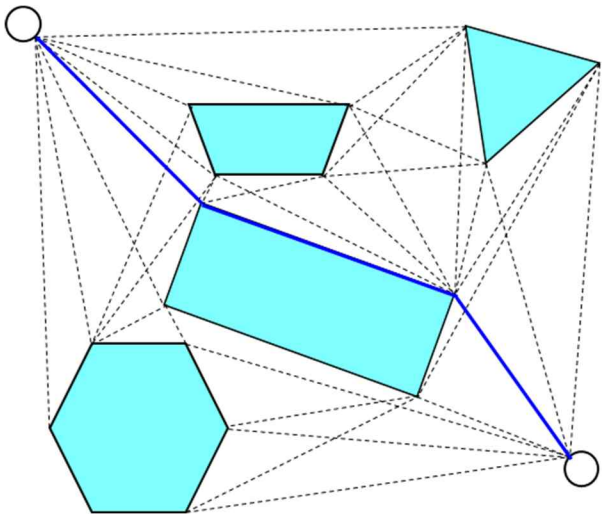

Figure 6-3 The shortest path on visibility graph

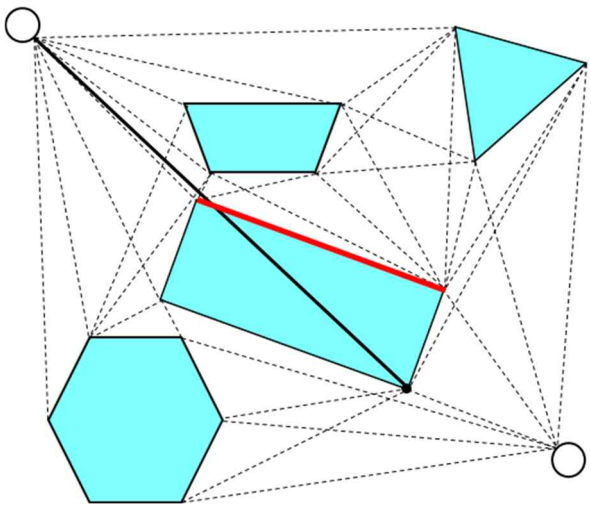

Figure 6-4 The collision between visible line and obstacle

6.2 Visibility graph with quadtree

가시성 그래프의 장점인 최적 해를 찾으면서 계산 시간을 줄이기 위해 본 논문에서는 쿼드 트리를 함께 사용하였다. 임의의 공간에서 점 n개가 주어졌을 때 가시성 그래프의 시간 복잡도는 $O(n^2 \cdot n)$이다. Figure 6-1에 있는 알고리즘을 이용하여 가시성 그래프를 생성하는데 필요한 시간 복잡도는 $O(n^2)$이다. 각각의 간선에 대해 Figure 6-2의 가시화 선분 체크 알고리즘을 수행하여야 하며, 가시화 선분 체크 알고리즘의 시간 복잡도는 $O(n)$이다. 본 논문에서는 쿼드 트리를 이용하여 가시성 그래프의 시간 복잡도를 $O(k^2)$, $k \ll n$으로 낮춰서 가시성 그래프의 최적 해와 유사한 해를 도출하고자 한다.

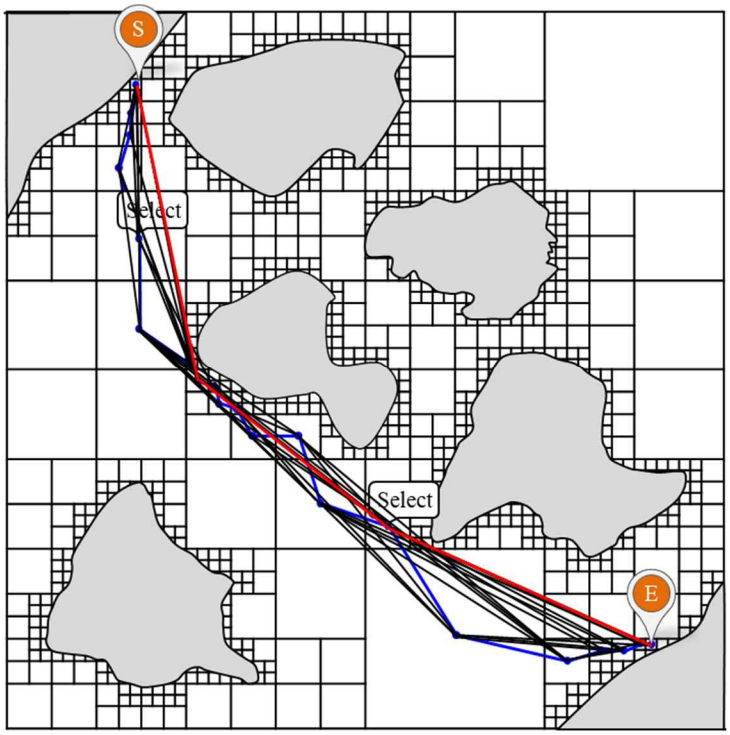

Figure 6-5 An example of visibility graph with quadtree

쿼드 트리를 이용하여 가시성 그래프를 생성하고 그 위에서 최적 경로를 구하는 방법은 Figure 6-5와 같다. 먼저 쿼드 트리를 이용하여 영역을 불규칙하게 표현한다. 이 때 격자의 크기는 설정한 분할 조건(Splitting threshold)과 최대 깊이(max depth)에 따라 결정된다. Figure 6-5의 예제에 대해서는 깊이 4와 분할 조건 2를 설정했으며, PMR 쿼드 트리를 사용하였다. 먼저, 쿼드 트리 노드의 통항 가능 여부를 판단하고 통항 가능한 노드를 4방향으로 연결하여 그래프를 생성한다. 그래프를 생성한 후 Dijkstra 알고리즘을 이용하여 최단 거리가 되는 경로를 계산한다. 이 경로가 Figure 6-5의 보라색으로 표시된 경로이다. 쿼드 트리 기반 경로의 점들이 가시성 그래프의 입력 점이 된다. 이 점들을 이용하여 가시성 그래프를 생성하며, 경로 위 점들을 이용하여 생성한 가시성 그래프는 Figure 6-5의 검은색 그래프이다. 이 그래프 위에서 다시 한번 Dijkstra 알고리즘을 수행하여 출발지와 목적지 사이의 최적 경로를 찾는다. 이렇게 찾은 최적 경로는 Figure 6-5의 파란색 경로이며, 붉은색 경로는 쿼드 트리 기반 그래프 위에서 최적 경로를 찾은 경로이다. Figure 6-5에서 볼 수 있듯, 파란색 경로의 불필요한 점들이 제거된 것을 확인할 수 있다.

6.3 Improved line of sight algorithm

가시성 그래프를 생성할 때, 임의의 두 점을 선택하여 가시성 선분(Visible line segment)을 연결하고 가시성 선분이 다른 장애물과 충돌하는

지 확인해야 한다. 가시성 그래프에서 가시성 선분이 다른 장애물과 충돌하는 것을 판단하기 위해 모든 장애물의 최소최대 경계를 계산하고 가시성 선분의 최소최대 경계와 겹치는지 확인한다. 겹치는 장애물들을 선별하여 선별된 장애물과 가시성 선분의 겹침 여부를 확인한다. 이 과정이 일반적으로 $O(n)$의 시간 복잡도를 가진다. 이 시간 복잡도를 낮추기 위해 쿼드 트리 구조를 사용한다. 일반적인 가시성 선분의 장애물 교차 판단은 Figure 6-6과 같다. 하지만 PMR 쿼드 트리는 블록 안에 간선 정보를 저장하고 있으므로 가시성 선분이 지나는 쿼드 트리 블록들을 탐색하고 그 블록 안에 있는 간선들만 가시성 선분과의 교차를 수행한다. 이 방법은 $O(bgn)$이며, PMR 쿼드 트리의 특성을 이용하여 가시성 선분을 체크하는데 시간을 단축했다고 할 수 있다. Figure 6-7은 Figure 6-6의 예제에서 쿼드 트리를 이용하여 가시성 선분과 쿼드 트리 블록이 겹치는 부분을 계산한 것이다. 이를 이용하여 가시성 그래프를 생성할 때의 시간을 대폭 줄일 수 있다. 특히, 가시성 선분이 길면 길수록 가시성 선분의 장애물 판단 시간은 더더욱 줄어든다.

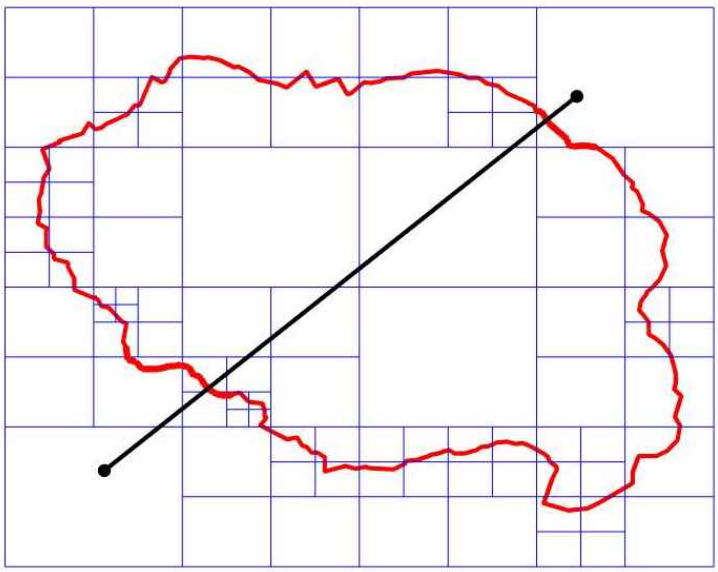

Figure 6-6 Straight line and quadtree representation with a polygon

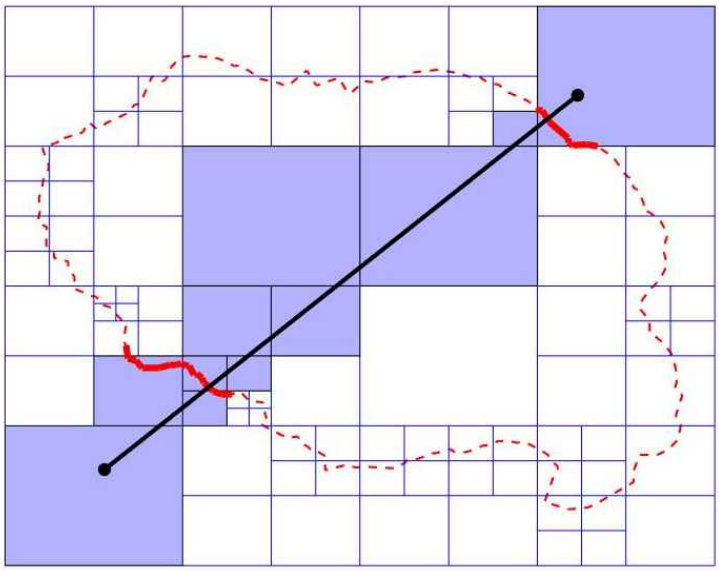

Figure 6-7 Sample line-of-sight algorithm implementation with quadtree representation

7 Fuel oil consumption estimation

선박의 항로를 계획할 때 고려하는 요소 중 효율과 관련된 요소는 연료 소모량이다. 연료 소모량을 계산하기 위해 선박의 저항과 속력을 계산해야 한다. 본 장에서는 선박의 저항과 속력을 계산하는 방법을 살펴보고, 연료 소모량을 항로 계획에 반영하고자 한다.

먼저, 7.1절에서는 구조물에 작용하는 정수 중 저항, ISO 15016에 기반하여 계산한 바람에 의한 저항 증가, 파랑에 의한 저항 증가를 설명한다. 7.2절에서는 해류에 의한 속도 변화를 설명한다. 7.3절에서는 정수 중 저항, 바람에 의한 저항 증가, 파랑에 의한 저항 증가를 반영한 총 저항과 해류에 의한 속력 변화 예시를 설명한다. 7.4절에서는 총 저항과 속력을 이용하여 제동 동력을 계산한 후 연료 소모량을 계산하는 방법에 대해 설명한다.

7.1 Ship resistance

본 논문에서는 기상 환경 중에서 항해하는 선박의 저항을 계산하고자 한다. 먼저, 정수 중 선박의 저항을 계산하고, ISO 15016의 부가저항 계산 식을 이용하여 파랑과 바람의 부가 저항을 계산한다(ISO, 2015). 본 절에서는 이러한 계산을 위하여 필요한 계수를 계산하고자 한다. 정수 중 전 저항과 ISO 15016에 있는 부가 저항을 이용하면 총 저항은 다음과 같이 계산할 수 있다.

$$R_T = R_{TS} + R_{AA} + R_{AW} + R_{AS} \quad (7-1)$$

여기서 R_T는 총 저항 (N), R_{TS}는 정수 중 저항, R_{AA}는 바람에 의해 생기는 부가저항, R_{AW}는 파랑에 의해 생기는 부가 저항, R_{AS}는 수온과 밀도에 의한 저항 변화이다. 여기서 수온과 밀도는 거의 변하지 않는다는 가정 하에 수온과 밀도에 의한 저항 변화 R_{AS} 를 무시한다.

정수 중 저항 계산은 다음 식과 같이 나타낼 수 있다.

$$R_{TS} = \frac{1}{2} C_{TS} \rho S V^2 = \frac{1}{2}\left((C_{FS} + \Delta C_F) \frac{S + S_{BK}}{S} + C_R \right) \rho S V^2 \quad (7-2)$$

여기서 C_{TS}는 전 저항 계수, C_{FS}는 평판마찰계수, ΔC_F는 마찰 계수 증가, S는 침수표면적 (m^2), S_{BK}는 빌지 킬의 침수표면적 (m^2), C_R은 마찰 계수이다. 이 때 평판마찰계수 C_{FS}는 ITTC 1957 모형선 실선 상관 곡선 식을 적용하여 다음과 같이 구한다.

$$C_{FS} = \frac{0.075}{(\log R_{nS} - 2)^2} \quad (7-3)$$

이 때 레이놀즈 수 R_{nS}는 다음과 같이 구한다.

$$R_{nS} = \frac{V * L}{\nu} \quad (7-4)$$

동점성계수 ν는 해수에 대해 다음과 같이 계산한다.

$$\nu_S = [(0.000659 * (T_S - 1.0) - 0.05076) * (T_S - 1.0) + 1.7688] * 0.00001 \quad (7-5)$$

여기서 T_S는 수온(℃) 이다.

일반적으로 선박의 제원에 C_{TS}는 포함되어 있으므로 제공받은 C_{TS} 정보를 그대로 사용하며, C_{TS} 정보가 없는 경우에만 위 식을 사용한다.

ISO 15016에 따라 바람에 의해 생기는 부가저항은 아래 식과 같이 나타낸다(ISO, 2015).

$$R_{AA} = 0.5\rho_A C_{AA}(\psi_{WR}) A_{XV} V_{WR}^2 - 0.5\rho_A C_{AA}(0) A_{XV} V_G^2 \qquad (7-6)$$

여기서 ρ_A는 공기의 밀도 (kg/m^3), C_{AA}는 공기 저항 계수, ψ_{WR}은 선박에 대한 바람의 방향, A_{XV}는 선박의 상부 투영 면적 (m^2), V_{WR}은 바람의 상대 속도 (m/s)이다. C_{AA}는 선박의 종류와 크기에 따라 결정되며, 이 값도 주어지는 정보이다. 선박의 진행방향과 바람의 방향, 선박의 속도, 바람의 속도 관계를 아래 그림에 나타냈다(ISO, 2015). 바람이 불어오는 방향과 선박의 진행방향을 고려하면 선박을 기준으로 바람의 방향을 계산할 수 있으며, 이 방향이 ψ_{WR}이다. 상대 속도는 선박의 대지속도(Speed over ground)와 바람 속도를 이용하여 계산하며, 이 상대속도가 V_{WR}이다.

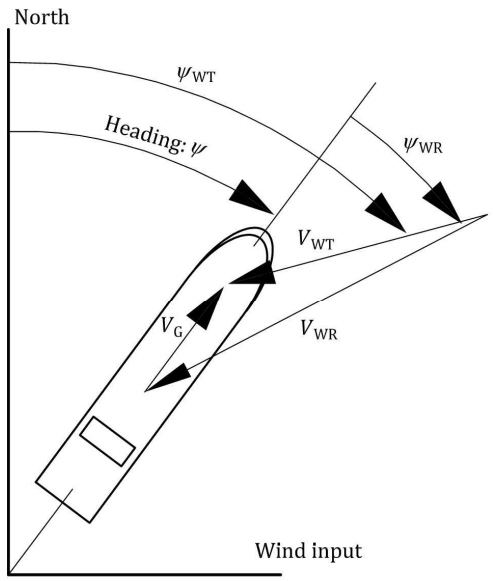

Figure 7-1 The relationship between ship direction and wind direction (ISO, 2015)

아래 그림은 26.78m 소형선의 C_{AA}를 바람 방향에 따라 나타낸 것이다.

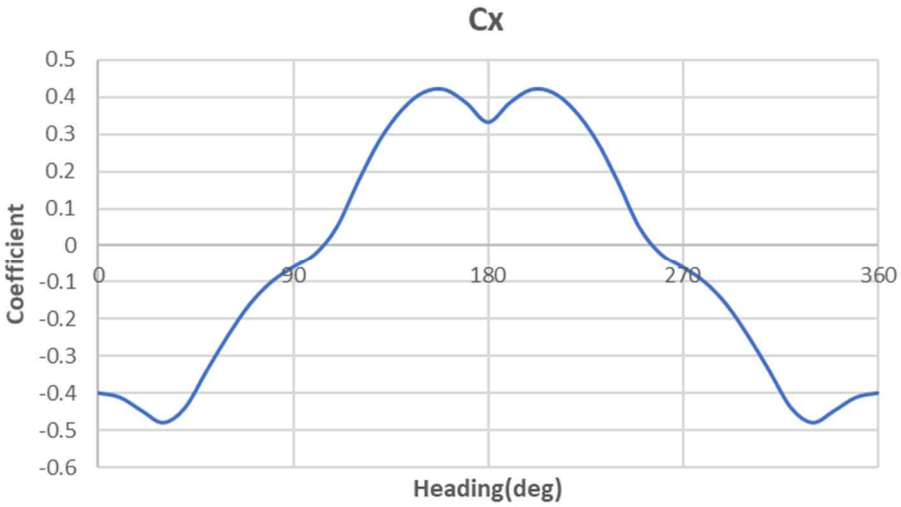

Figure 7-2 Wind resistance coefficients for sample ship

ISO 15016에 따라 파랑에 의해 생기는 부가저항은 아래 식과 같이 나타낸다(ISO, 2015).

$$R_{AW} = 2\int_0^{2\pi}\int_0^{\infty} \frac{R_{w\,ave}(\omega,\alpha,V_S)}{\zeta_A^2} E(\omega,\alpha)d\omega\, d\alpha \qquad (7-7)$$

여기서 $R_{w\,ave}$는 규칙 파랑 중 부가 저항 (N), ζ_A는 파랑의 진폭 (m), ω는 파랑의 주파수 (rad/s), α는 선박의 진행방향과 파랑 사이 각도 (rad), V_S는 선박의 대수속도(Speed through water), E는 스펙트럼 $(m^2 \cdot s)$이다.

$R_{w\,ave}$를 모형선 테스트를 통해 알고 있어야 하지만, 소형선의 경우에는 이러한 테스트를 진행하지 않기 때문에 파랑 부가저항을 추정(approximation)해야 한다. ISO 15016에는 파랑 부가저항을 추정하는 두 가지 방법이 있다. STAWAVE-1과 STAWAVE-2 방법이 있다.

STAWAVE-1 방법을 이용하여 파랑 부가저항을 추정하는 방법은 아래와 같다.

$$R_{AW} = \frac{1}{16}\rho_S g H_{1/3}^2 B\sqrt{\frac{B}{L_{BW\,L}}} \tag{7-8}$$

여기서 ρ_S는 물의 밀도 (kg/m^3), g는 중력 가속도 (m/s^2), $H_{1/3}$은 유의파고 (m), B는 선박의 폭 (m), $L_{BW\,L}$은 선수에서 폭의 95%가 되는 지점까지의 거리 (m)이다. STAWAVE-1 방법은 $H_{1/3} \le 2.25\sqrt{\frac{L_{pp}}{100}}$이며, 파랑 방향이 선수 기준으로 $0° \pm 45°$ 안에 적용되어야 한다.

STAWAVE-2 방법을 이용하여 파랑 부가저항을 추정하는 방법은 아래와 같다.

$$R_{w\,ave} = R_{AW\,M\,L} + R_{AW\,RL} \tag{7-9}$$

여기서 $R_{AW\,M\,L}$는 선박의 거동에 의해 생기는 저항 (N) 이며, $R_{AW\,RL}$은 파랑의 반사에 의해 생기는 평균 저항 (N) 이다. 각 저항은 다음 식을 이용하여 계산된다.

$$R_{AW\,M\,L} = 4\rho_S g \zeta_A^2 \frac{B^2}{L_{PP}} \overline{r_{aw}}(\omega) \tag{7-10}$$

$$\overline{r_{aw}}(\omega) = \bar{\omega}^{b_1} exp\left(\frac{b_1}{d_1}\left(1-\bar{\omega}^{d_1}\right)\right) a_1 Fr^{1.5} exp\,(-3.5Fr) \tag{7-11}$$

$$\bar{\omega} = \frac{\sqrt{\frac{L_{PP}}{g}}\sqrt[3]{k_{yy}}}{1.17Fr^{-0.143}}\omega \tag{7-12}$$

$$a_1 = 60.3C_B^{1.34} \tag{7-13}$$

$$b_1 = \begin{cases} 11 & for \quad \bar{\omega} < 1 \\ -8.5 & elsew \quad here \end{cases} \tag{7-14}$$

$$d_1 = \begin{cases} 14 & for \quad \bar{\omega} < 1 \\ -566\left(\frac{L_{PP}}{B}\right)^{-2.66} & elsew \quad here \end{cases} \tag{7-15}$$

$$R_{AW\ ML} = 4\rho_S g \zeta_A^2 \frac{B^2}{L_{PP}} \overline{r_{aw}}(\omega) \tag{7-16}$$

$$R_{AW\ RL} = \frac{1}{2}\rho_S g \zeta_A^2 B \alpha_1\ (\omega) \tag{7-17}$$

$$\alpha_1(\omega) = \frac{\pi^2 I_1^2(1.5kT_M)}{\pi^2 I_1^2(1.5kT_M) + K_1^2(1.5kT_M)} f_1 \tag{7-18}$$

$$f_1 = 0.692\left(\frac{V_S}{\sqrt{T_M g}}\right)^{0.769} + 1.81 C_B^{6.95} \tag{7-19}$$

여기서 ρ_S는 물의 밀도 (kg/m^3), g는 중력 가속도 (m/s^2), ζ_A는 파랑의 진폭 (m), L_{pp}는 선박의 길이 (m), B는 선박의 폭 (m), T_m은 중앙에서 선박의 흘수 (m), C_B는 방형 계수, Fr은 Froude 수, V_s는 대수 속도(speed through water), k_{yy}는 측면 방향 무차원화 회전 반경, I_1은 차수가 1인 제 1종 변형 베셀 함수, K_1은 차수가 1인 제 2종 변형 베셀 함수, k는 파수이다.

파랑 부가저항은 아래 식을 이용하여 계산한다.

$$R_{AW} = 2\int_0^{\infty} \frac{R_{w\,ave}(\omega,\ V_S)}{\zeta_A^2} S_\eta(\omega) d\omega \tag{7-20}$$

여기서 R_{AW}는 비규칙파에서 파랑 부가저항 증가 (N), $R_{w\,ave}$는 규칙파에

서 저항 증가 (N), ζ_A는 파랑의 진폭 (m), ω는 파랑의 주파수 (rad/s), V_S는 선박의 대수속도(Speed through water), S_η는 변형 Pierson-Moskowitz 스펙트럼이다.

STAWAVE-2 방법은 다음의 제약 조건 안에서 적용된다. 아래 조건과 함께 파랑 방향이 선수 기준으로 $0° \pm 45°$ 이어야 적용이 가능하다.

$$75 < L_{pp}, 4.0 < \frac{L_{pp}}{B} < 9.0, 2.2 < \frac{B}{T_M} < 9.0, 0.1 < Fr < 0.3, 0.5 < C_B < 0.9 \qquad (7-21)$$

본 논문에서는 소형 선박에도 ISO 15016을 적용하고자 한다. 소형선에 대해 파랑 부가저항을 계산할 때 주어진 정보가 너무 부족하기 때문에 다른 방법은 사용하기가 어렵다 따라서 파랑 부가저항을 계산하기 위해 ISO 15016을 사용하였다.

7.2 Ship speed

선박의 운항 중에 조류가 흐른다면, 조류의 속도는 선박의 대지 속도와 진행 방향에 영향을 준다. 선박과 조류가 아래 그림과 같이 존재한다고 가정하자. 이 때 조류를 x 방향으로 V_{cx}, y 방향으로 V_{cy}로 성분을 분해할 수 있다. 조류를 반영한 선박의 속도와 선수각은 V_N과 θ_N으로 표현하며, 아래 수식을 이용하여 계산한다. 조류를 반영했을 때 속력 변화는 아래 식과 같이 나타낼 수 있다.

$$V_x = V\sin\theta + V_{cx}, V_y = V\cos\theta + V_{cy} \qquad (7-22)$$

$$V_N = \sqrt{V_x^2 + V_y^2}, \theta_N = \tan^{-1}(\frac{V\cos\theta + V_{cx}}{V\cos\theta + V_{cy}}) \quad (7-23)$$

$$V_N - V = \sqrt{V_x^2 + V_y^2} - V = \sqrt{(V\sin\theta + V_{cx})^2 + (V\cos\theta + V_{cy})^2} - V \quad (7-24)$$

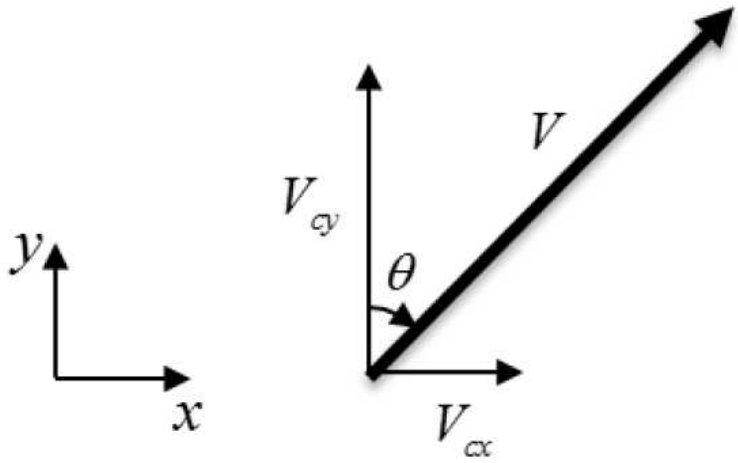

Figure 7-3 Ship speed with current

7.3 Fuel oil consumption

선박의 연료 소모량을 계산하기 위해서 제동 동력을 계산해야 한다. 유효 동력을 계산한 후 추진 효율을 이용하여 제동 동력을 계산한다. 유효 동력, 전달 동력, 제동 동력은 Figure 7-4에 표현하였다. 선박의 저항과 속력이 클수록 선박의 연료 소모량은 커진다.

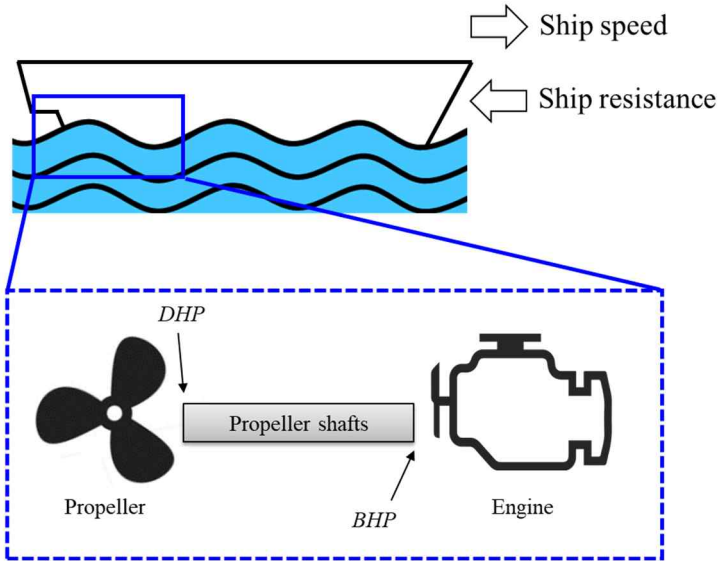

Figure 7-4 Brake horsepower estimation

$$EHP = R_T \cdot V_N = (R_{TS} + R_{AW} + R_{AA}) \cdot (V + \Delta V) \quad (7-25)$$

여기서 EHP는 유효동력(Effective Horse Power), R_T는 전체 저항, V_N은 조류를 반영한 속도이다. 전체 저항 R_T는 정수 중 저항 R_{TS}, 파랑 부가저항 R_{AW}, 바람 부가저항 R_{AA}의 합으로 이루어진다. 조류를 반영한 속도 V_N은 선박의 대지 속도 V와 조류 속도 ΔV의 합을 이루어진다.

$$DHP = \frac{EHP}{\eta_D} \quad (7-26)$$

$$\eta_D = \eta_H \eta_R \eta_O = \frac{1-t}{1-\omega} \eta_R \eta_O \quad (7-27)$$

여기서 DHP는 전달 동력(Delivered power), EHP는 유효 동력, η_D는 준추진효율이다. 준추진효율 η_D는 선각효율 η_H, 프로펠러 회전 효율 η_R, 프로펠러 단독 효율 η_O의 곱으로 이루어진다. η_H는 추력감소비 $1-t$와 반류비 $1-\omega$의 곱으로 이루어진다.

일반적으로 선체의 세가지 추진계수인 추력감소비, 반류비, 상대회전효율

은 흘수가 달라지면 모두 변하는 값이다. 그러나 흘수 차이가 크지 않은 연안 선박의 경우 추진계수 변화량이 작다고 가정하여 동일한 값을 사용한다. 프로펠러 단독 효율도 흘수 상태가 달라지면 그 값이 달라질 수 있으나, 이 값 역시 흘수 변화량에 관계없이 일정하다고 가정한다.

$$BHP = \frac{DHP}{\eta_T} \tag{7-28}$$

여기서 BHP는 제동 동력(Brake Horse Power)이며, DHP는 전달 동력, η_T는 축 전달효율이다. 축 전달효율은 흘수 변화량에 상관없이 일정한 값을 가진다.

$$FOC = f_{rate} \cdot BHP \cdot t \tag{7-29}$$

여기서 FOC는 연료 소모량 (L), f_{rate}는 연료 소모율 $(g/ps \cdot h)$, BHP는 제동 동력, t는 항해 시간 (h)이다.

7.4 Example of fuel oil consumption computation

본 절에서는 앞 절에서 설명한 기상 조건 변화에 따른 선박의 저항 증가, 속도 변화를 고려하여 연료소모량을 계산하고자 한다. 선박이 시작점 (128.3001° E, 37.09° N)으로부터 끝점 (122.7071° E, 34.7084° N)으로 2020년 12월 23일 오전 0시 (한국 표준시)에 상대속도 8 노트(4.1152 m/s)로 항해하는 상황을 가정하였다. 이 때 예상되는 기상 조건은 KMA, KHOA로부터 아래 Table 7-1과 같이 예상되었다.

Table 7-1 Test conditions for fuel oil consumption estimation

Item		Value
Wave	Wave height (m)	0.40
	Wave period (sec)	0.62
	Wave direction (rad)	0.2246 (12.86°)
Wind	Wind speed (m/s)	5.10
	Wind direction (rad)	0.1180 (6.77°)
Current	Current speed(m/s)	0.10
	Current direction(rad)	0.0561 (3.21°)

선박의 항로와 기상 조건을 정리하면 Figure 7-5과 같다.

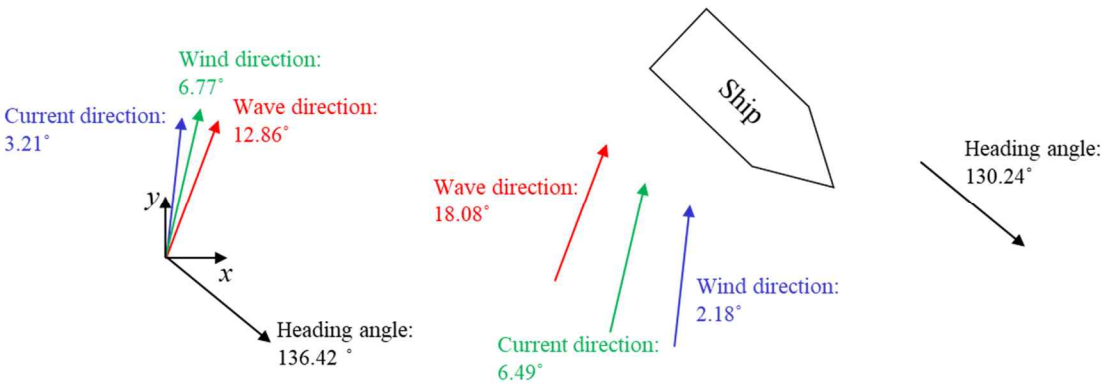

Figure 7-5 Ocean and weather conditions for fuel oil consumption

먼저 선박의 항로를 분석하면, 시작점과 끝점까지의 항해 거리는 1.1332km이며, 선수각은 2.381 rad (136.42 °)이다. 상대 속도는 4.1152m/s였으며, 조류를 고려한 선박의 속력은 식 (7-30)과 같이 4.047m/s로 감소하였다.

$$\begin{aligned} V_N &= \sqrt{(V\sin\theta + V_{cx})^2 + \left(V\cos\theta + V_{cy}\right)^2} \\ &= \sqrt{\begin{matrix}(8\times 0.5144\times \sin(2.381) + 0.099843)^2 \\ +(8\times 0.5144\times \cos(2.381) + 0.005607)^2\end{matrix}} \\ &= 4.047 m/s (7.87\ knots) \end{aligned} \quad (7-30)$$

선박에 작용하는 정수 중 저항은 식 (7-31)와 같이 12,953.03 N이다. 식 (7-32)에서 볼 수 있듯 바람 부가 저항은 1142.49 N이다. 식 (7-33)과 같이 362.19 N이다. 선박의 전체 저항은 정수 중 저항, 바람 부가저항, 파랑 부가저항의 합으로 14,457 N이다.

$$\begin{aligned} R_{TS} &= \frac{1}{2}\rho C_D S V^2 \\ &= \frac{1}{2}\times 1{,}025\times 221.2\times 0.006747\times (8\times 0.5144)^2 \\ &= 12{,}953.03(\mathrm{N}) \end{aligned} \quad (7-31)$$

$$\begin{aligned} R_{AA} &= 0.5\rho_A C_{AA}(129.66) A_{XV} V_{W\ R}^2 - 0.5\rho_A C_{AA}(0) A_{XV} V_G^2 \\ &= 0.5\cdot 1.293\cdot 0.273\cdot 47.55\cdot 8.35^2 \\ &\quad -0.5\cdot 1.293\ \cdot(-0.4)\cdot 47.55\cdot 4.11^2 = 778.95(\mathrm{N}) \end{aligned} \quad (7-32)$$

$$R_{AW} = 2\int_0^{\infty} \frac{R_{w\,ave}(\omega,\ V_S)}{\zeta_A^2} S_\eta(\omega)d\omega = 362.19\text{N} \qquad (7-33)$$

유효 동력(EHP)은 선체의 저항과 선박의 속력의 곱으로 식 (7-34)와 같이 계산된다.

$$EHP = R_T \cdot V_N = 58{,}507W = 77.60\ ps \qquad (7-34)$$

전달 동력(DHP)은 유효 동력을 준추진효율로 나눈 값이며 식 (7-35)과 같이 계산되며, 준추진효율은 선각효율, 프로펠러 단독효율, 상대회전효율의 곱으로 계산된다.

$$DHP = \frac{EHP}{\eta_D} = \frac{EHP}{\eta_H \eta_R \eta_O} = \frac{77.60}{1.04 \cdot 0.97 \cdot 0.550} = 139.85\ ps \qquad (7-35)$$

제동 동력(BHP)은 전달 동력을 전달효율로 나눈 값이며 식 (7-36)와 같이 계산되며, 준추진효율은 선각효율, 프로펠러 단독효율, 상대회전효율의 곱으로 계산된다.

$$BHP = \frac{DHP}{\eta_T} = \frac{139.85}{0.98} = 142.70\ ps \qquad (7-36)$$

시간당 연료 소모량($TFOC$)은 제동 동력에 연료 소모율을 곱한 값이며 식 (7-37)와 같이 계산된다.

$$TFOC = f_{rate} \cdot BHP = 0.156 \cdot 142.70 = 22.26L/h \qquad (7-37)$$

연료 소모량(FOC)은 시간당 연료 소모량과 항해 시간을 곱한 값이며 식 (7-38)과 같이 계산된다.

$$FOC = f_{rate} \cdot BHP \cdot t = 0.156 \cdot 142.70 \cdot 0.0777 = 1.729L \qquad (7-38)$$

8 Coastal path planning algorithm

본 장에서는 단기 기상 예보를 이용하여 연안 선박의 최적 경로를 찾는다. 최적 경로를 찾는 방법으로 Dijkstra 알고리즘을 사용하였다.

8.1 Problem definition

본 장에서는 항해가 시작될 시점에서 단기 예보가 확보되었을 경우를 가정하여, 연안 선박의 최적 항로를 찾고자 한다. 이러한 최적화를 수행하기 위해서 아래의 정보가 필요하다.

(1) 선박의 항로

(2) 항로의 탐색 영역에 대한 기상 정보

(3) 기상 조건 하에서 선박의 연료 소모량

먼저 선박의 항로가 결정되어야 한다. 항로 계획을 세울 시점에 선박의 출항지와 입항지는 결정되어 있다. 따라서 이 두 장소를 연결하는 효율적인 경로를 선택해야 한다. 기본적으로 장애물과 겹치지 않는 항로를 찾아야 하며, 기상 정보를 고려하여 저항을 적게 받아 연료 소모량이 최소가 되는 항로를 찾아야 한다.

두번째로 기상 정보는 선박의 항해 계획을 세울 때 시간과 공간에 따라 변하는 중요한 변수이다. 기상 조건에 따라 항해가 어려울 수 있으며, 항해 중 위험한 상황이 될 수도 있기 때문이다. 기상 정보의 제공 영역은 충분히

넓기 때문에 항로를 탐색하는데 활용될 수 있다. 하지만, 본 논문에서는 시간에 따른 기상 정보는 고려하지 않았다. 시간을 고려하게 되면 2차원 탐색에서 3차원 탐색을 수행해야 하기 때문에 연산 시간이 크게 증가하게 된다. 일반적으로 연안 선박의 항해는 6시간 미만으로 충분히 짧기 때문에 선박의 출항 시각에 예보한 정보를 입항지에 도달할 때까지 동일하게 사용한다.

세번째는 기상 조건하에서의 선박의 연료 소모량이다. 앞에서 언급한 기상 조건이 되었을 때, 선박의 저항을 계산할 수 있어야 어떤 항로가 최적 항로인지 계산할 수 있다. 7장에서 언급한 방법을 이용하여 연료 소모량을 계산하였다.

정리하면 본 논문에서는 선박의 출항지와 입항지, 출항 시각에 따라 장애물을 회피하면서 주어진 기상조건 하에 연료 소모량이 최소가 되는 항로를 계산하였다. 정리한 최적화 문제는 다음과 같다.

$$Minimize \quad FOC(R) \tag{8-1}$$

여기서 R은 항로를 의미하며, FOC는 연료 소모량을 의미한다. 항로 R은 시작점과 도착점 사이의 중간 지점들의 연결로 얻을 수 있다. 이러한 중간 지점들을 변침점(waypoint)이라 하며, 항로의 지형적 특성에 따라 정해질 수 있다. 변침점은 꼭 필요한 위치에만 존재해야 하며, 불필요하게 설정될 경우 항로의 거리, 시간, 연료 소모량을 증가시키는 요소이다. 따라서 6장에서 설명한 방법을 이용하여 불필요한 변침점을 제거하였다.

8.2 Assumptions for route optimization

최적화 문제를 푸는데 다음의 가정을 적용하였다.

(1)기상 조건은 예측된 기상 데이터인 KMA와 KHOA의 정보를 이용한다.

(2) 기상 조건은 단일 격자 내에서 동일하다.

(3) 선박의 저항 증가는 짧은 변화 내에서 선형성(linearity)을 가진다.

(4)선박의 항해 시간이 짧기 때문에 출항 시각의 기상 예보를 입항할 때까지 사용한다.

8.3 Procedure of coastal path planning algorithm

항로 최적화는 쿼드 트리와 가시성 그래프를 이용하여 탐색 영역을 설정하고 Dijkstra 알고리즘을 이용하였다. 8.1은 본 논문에서 제안한 알고리즘의 순서도이다. 기상 정보는 현재 시간으로부터 중기 예보가 가능한 시간(3일)의 정보를 저장하고 있으며, 출항 시각에 따른 기상 정보를 가져온다. 선박의 제원을 이용하여 사전에 파랑 부가저항, 바람 부가저항, 조류에 따른 선박의 속력 변화는 계산하였다. 최적 항로를 찾기 위해 다음의 순서를 따르며, 순서도는 Figure 8-1에 나타냈다.

(1) Step 1: 선박의 출항 시각에 따라 조위 정보를 전달받고 육지와 흘

수보다 낮은 수심 영역을 수집한다.

(2) Step 2: Polygon offset을 이용하여 수집한 장애물을 일정 크기만큼 확장한다.

(3) Step 3: PMR 쿼드 트리를 이용하여 쿼드 트리 기반 그래프를 생성한다.

(4) Step 4: step 3에서 생성한 쿼드 트리 그래프의 가중치를 선박의 연료 소모량으로 설정한다.

(5) Step 5: 출항지와 입항지를 입력한다.

(6) Step 6: Dijkstra 알고리즘을 이용하여 쿼드 트리 기반 그래프에서 연료 소모량이 최소가 되는 항로를 계산한다.

(7) Step 7: step 6에서 계산한 항로의 변침점을 이용하여 가시성 그래프를 생성한다. 그래프의 간선 가중치는 연료 소모량으로 설정한다.

(8) 가시성 그래프에서 다시 Dijkstra 알고리즘을 적용하여 최적 항로를 계산한다.

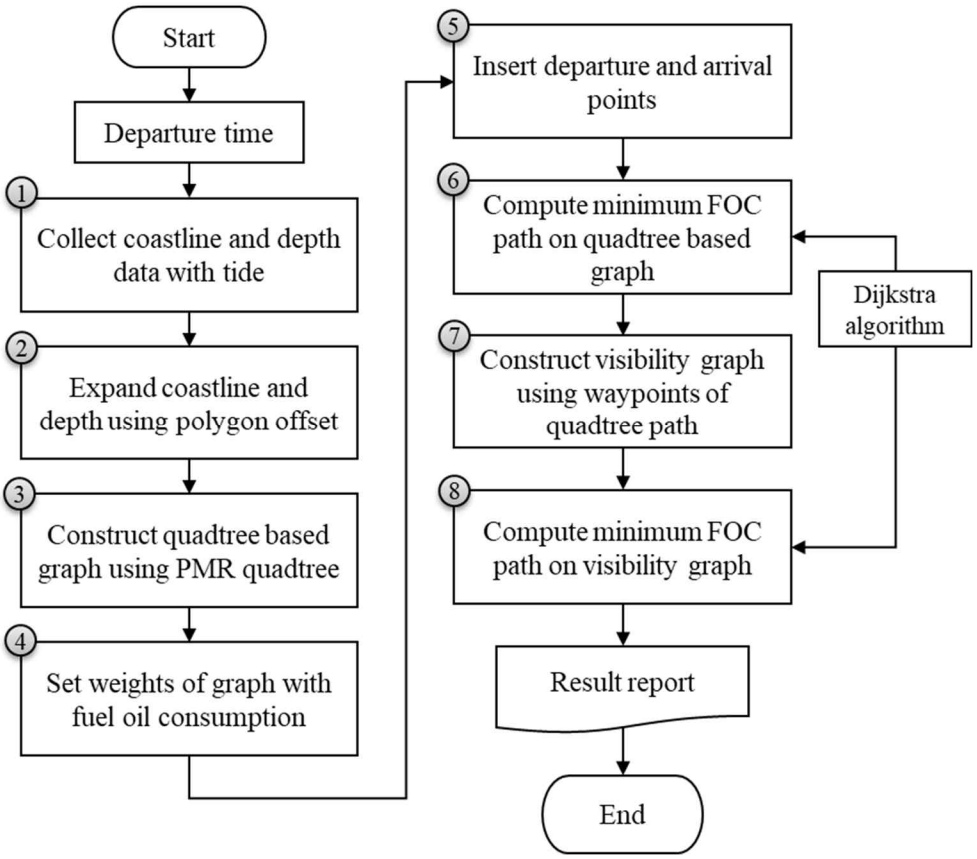

Figure 8-1 Flow chart of coastal path planning algorithm

Figure 8-1의 step 3에서 PMR 쿼드 트리를 이용한 그래프는 다음의 과정을 거쳐 수행된다. 순서도는 Figure 8-2에 나타냈다. 처음 그래프를 생성할 때 가중치는 거리 혹은 연료 소모량으로 설정할 수 있다.

(1) Step 1: 앞 단계에서 수집한 장애물을 이용하여 PMR 쿼드 트리를 생성한다.

(2) Step 2: 쿼드 트리에서 하위 노드가 없는 리프 노드만을 추출한다.

(3) Step 3: 리프 노드의 중앙점에 대해 PIP(Point In Polygon) 알고

리즘을 수행하며, 통항 가능 여부를 판단한다.

(4) Step 4: 하나의 노드를 선택한다.

(5) Step 5: 4방향으로 연결 가능하며 인접한 노드를 탐색한다.

(6) Step 6: 연결 가능한 간선을 쿼드 트리 기반 그래프에 추가한다.

(7) Step 7: 모든 노드에 대해 step 4-6을 반복한다.

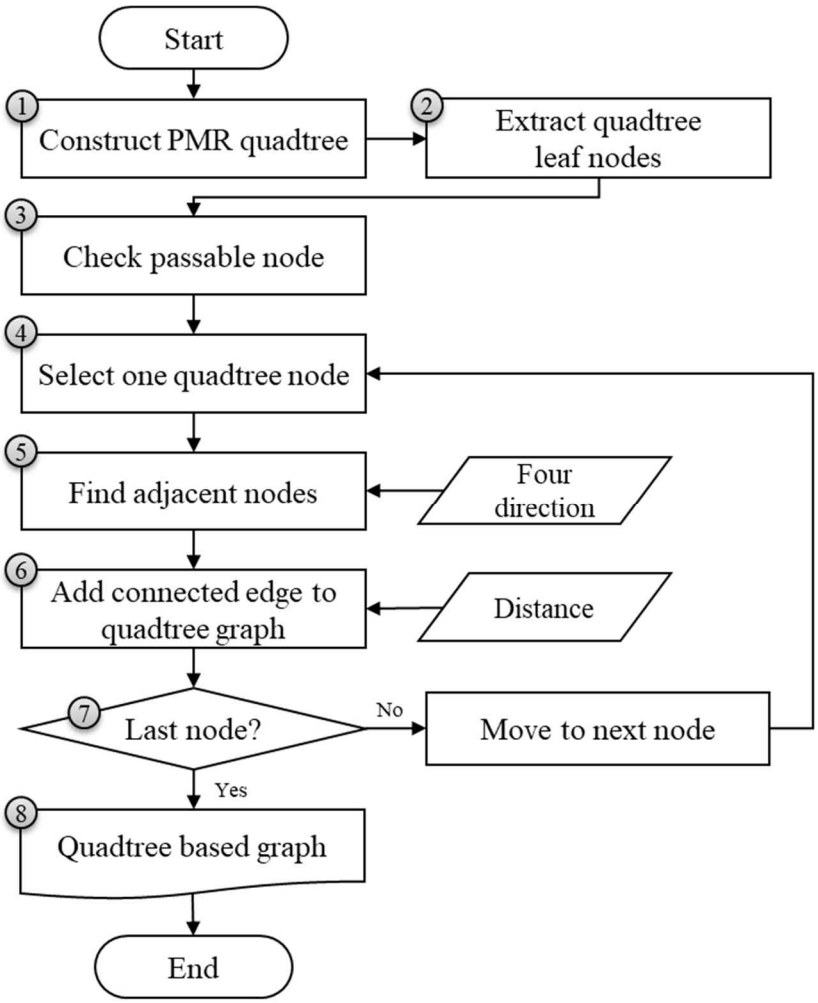

Figure 8-2 Flow chart of quadtree based graph generation

9 Experimental results

4 – 8장에서 제안한 방법을 한국의 남해안과 서해안에 적용하였다. 8.1절에서는 쿼드 트리를 생성하여 균일 격자와 격자 수 및 계산 시간을 비교하였다. 8.2절에서는 남해안에서 육지와 섬을 고려하여 최단 거리 기반으로 계획한 5가지 항로를 생성하였다. 8.3절에서는 서해안에서 육지와 섬을 고려하여 최단 거리 기반으로 계획한 5가지 항로를 생성하였다. 8.4절에서는 서해안과 남해안에서 수심, 기상 정보, 조위를 고려한 항로를 계획하고 비교하였다. 조위와 수심에 따라 항해 가능한 영역이 많이 달라지며, 연료 소모량이 최소가 되는 항로와 최단 거리 항로를 비교하였다.

시뮬레이션에서는 윈도우 10 환경에서 Visual studio 2019 c++를 사용하였으며, Intel Core i7-10700K CPU와 16.0GB 램을 사용하였다. 본 연구의 알고리즘에는 멀티 스레드나 GPU와 같은 가속화 기법을 사용하지 않았다.

9.1 Simulation results in quadtree generation

Table 9-1는 균일 격자와 쿼드 트리의 통항 가능한 노드 수와, 계산 시간, 각 구조의 최소 격자 크기를 비교한다. 그래프는 각 격자에서 4방향으로 생성된다. 일반 격자에서 통항 가능한 격자와 그래프 생성 시간은 격자의 해상도(resolution)에의 제곱에 비례한다. 복잡한 해양 환경에서, 격자의 크기는 육지를 표현하기 위해 매우 작아야 한다. 따라서, 균일 격자는

복잡한 해양환경에 적용하기 어렵다. 격자가 통항 가능한지 판단하기 위해 PIP 테스트를 수행하기 때문에 계산 시간은 격자의 해상도에 따라서 크게 증가한다. 쿼드 트리는 균일 격자의 이러한 단점을 극복할 수 있다. 쿼드 트리에서 최소 격자 크기는 균일 격자에 비해 상당히 작으며, 통항 가능한 격자도 훨씬 적다. 이러한 결과는 쿼드 트리가 균일 격자에 비해 복잡한 해양 환경을 표현하는데 유용하다는 것을 보여준다.

Table 9-1 Comparison of the number of passable nodes, computation time, and minimum grid size of the regular grids and quadtree.

Method	Resolution	Number of passable grids	Construction time of graph	Minimum grid size
Regular	512	101,636	2.808	180.41
	1024	406,658	11.074	90.21
	2048	1,626,524	49.319	45.10
Quadtree	-	62,989	3.813	5.60

쿼드 트리는 약 2.215초만에 생성되었으며, 최소 쿼드 트리 블록은 5.60m였다. 연안 선박의 최소 크기가 5m 이상인 것을 고려하면, 쿼드 트리 블록의 크기는 적당하다. 만약, 5.60m 크기를 갖는 균일 격자로 1°×1°를 표현하려면, 그래프를 구성하는데 20,000 × 20,000개의 격자가 필요하다. 4억개의 격자를 그래프로 변환하는 과정은 엄청난 메모리와 시간이 필요하다. 하지만, 쿼드 트리를 이용하면 동일한 영역을 약 63,000개로 표현할 수 있으며, 이는 메모리 소모량과 계산 시간을 줄일 수 있다. 육지와의

간섭 없이 쿼드 트리를 이용하여 그래프를 생성하는데 필요한 시간은 1.598초였다. 쿼드 트리 그래프를 생성하는데 필요한 시간은 3.813초로 맵이 자주 바뀌는 환경에 적합하다고 판단하였다.

9.2 Simulation results with different draft

본 연구에서는 수심과 조위를 고려하여 항로를 계획한다. 서해안의 경우 조위 차가 크게 발생하기 때문에 항로를 계획할 때 필수적으로 수심과 조위를 고려해야 한다. 아래 Figure 9-1과 Figure 9-2에 해안선과 수심을 나타냈다. 해안선만을 고려했을 때의 경로와 수심을 고려했을 때의 경로는 분명히 차이가 존재한다. 선박이 운항할 때 육지와 부딪치는 좌초는 반드시 피해야 하며, 저 수심 영역 등에 부딪치는 접촉도 반드시 피해야 한다. 이러한 이유로 선박의 흘수에 따른 수심을 고려하는 것은 필수이다.

Figure 9-3, Figure 9-4, Figure 9-5는 흘수 0.5m, 2m, 5m에 따른 항로를 보여준다. 흘수에 따라 항로가 다르게 나오는 것을 확인하였으며, 이를 Figure 9-6에 동시에 표현하였다. Figure 9-6에 체크한 부분에서 볼 수 있듯, 흘수를 높게 설정하면 저 수심지역을 회피해서 항로가 계산되는 것을 확인하였다.

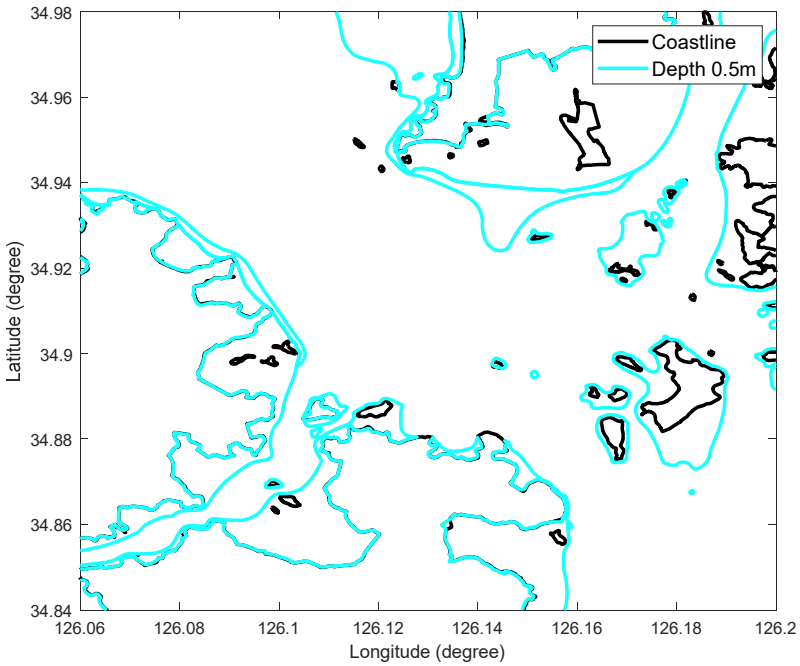

Figure 9-1 Coastline and depth

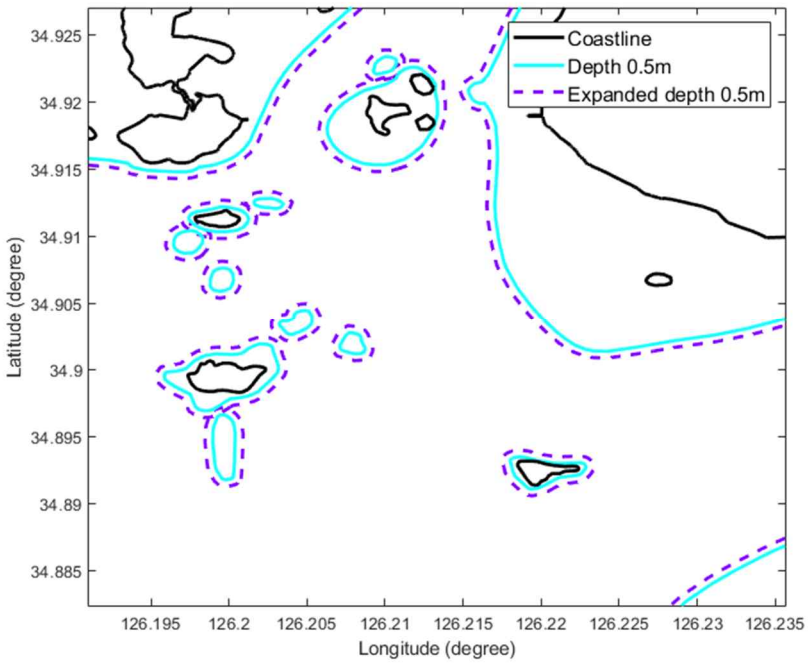

Figure 9-2 Details of coastline and depth

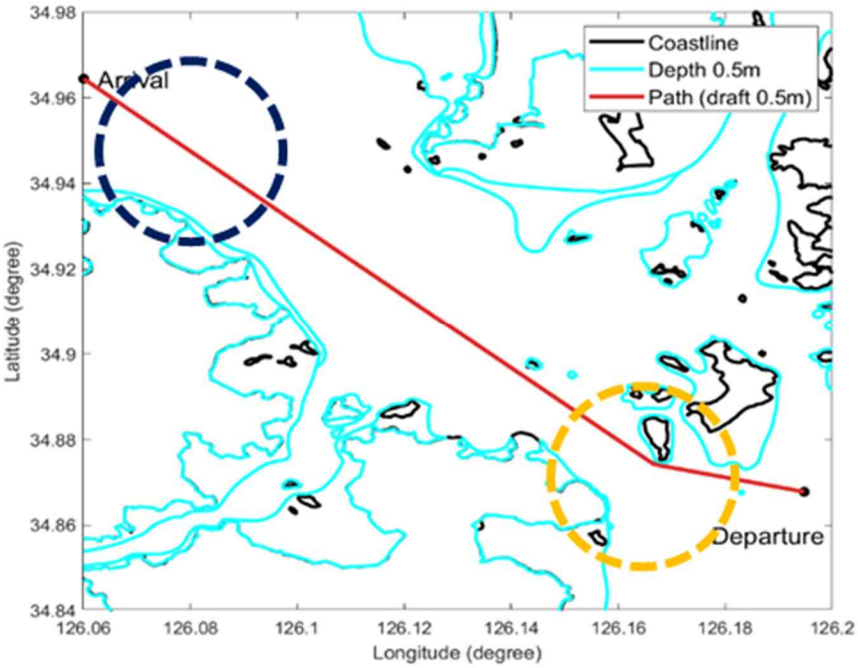

Figure 9-3 Path results with ship draft 0.5m

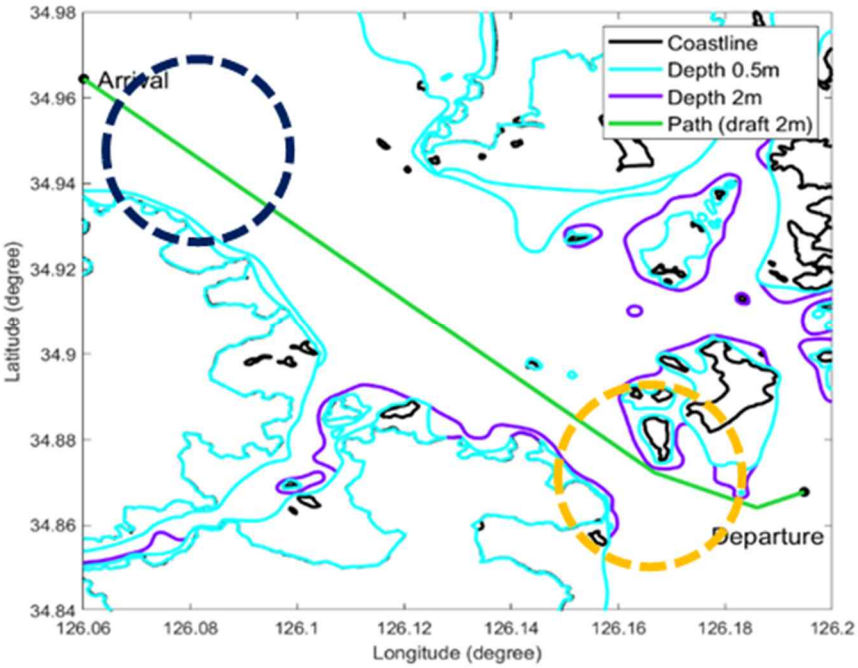

Figure 9-4 Path results with ship draft 2m

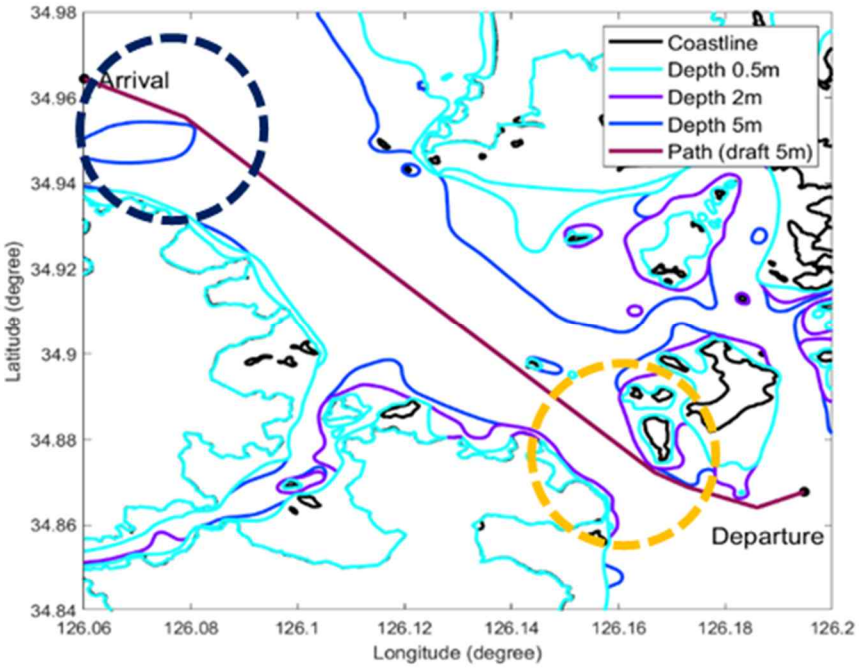

Figure 9-5 Path results with ship draft 5m

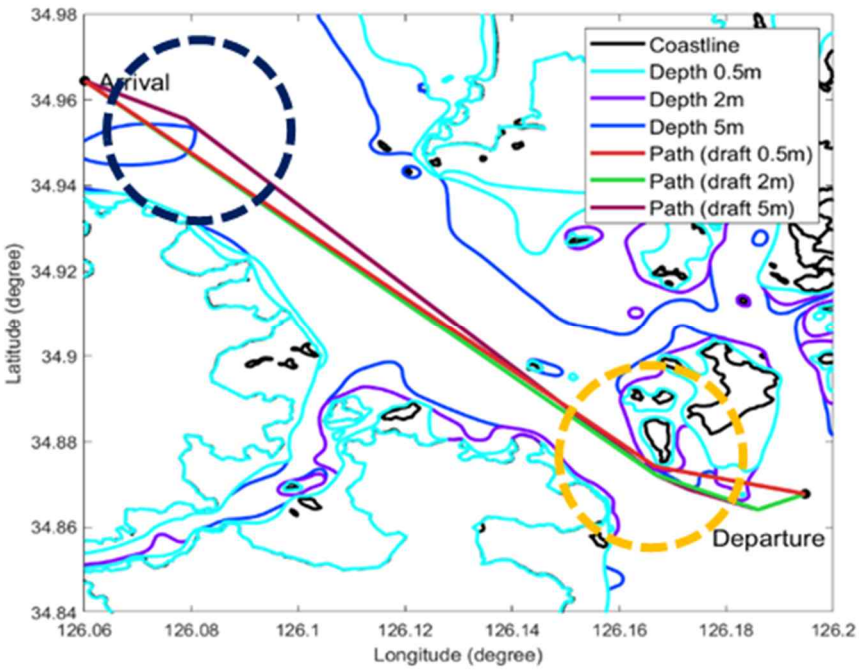

Figure 9-6 Path results with ship draft 0.5, 2, 5m

9.3 Simulation results in south sea of Korea

각 단계 별 쿼드 트리 구성, 쿼드 트리 그래프, 장애물 교차 제거 후 쿼드 트리 그래프 결과를 아래에 나타냈다. 생성한 쿼드 트리 그래프 위에서 출발지, 도착지를 설정하여 5가지 경우를 Table 9-2와 Table 9-3에 비교 및 분석하였다. Table 9-2에서 각 기호가 나타내는 의미는 다음과 같다.

Table 9-2 Comparison of quadtree and visibility-graph shortest paths for five cases in south sea of Korea (VG: visibility graph).

Case	p_d	p_a	l_q(km)	l_{VG} (km)	$l_{reduction}$ (%)
1	(127.82, 34.93)	(128.45, 34.92)	89.505	81.053	9.44
2	(127.80, 34.81)	(128.50, 35.04)	104.33	97.147	6.88
3	(127.93, 34.75)	(128.56, 34.92)	81.374	74.447	8.51
4	(127.81, 34.90)	(128.57, 34.82)	91.812	82.828	9.78
5	(127.78, 34.78)	(128.57, 34.76)	83.395	76.611	7.28

p_d: 출발지 (Departure point)

p_a: 도착지 (Arrival point)

l_Q: 쿼드 트리 기반 최적 항로 거리

l_{VG}: 가시성 그래프 기반 최적 항로 거리

$l_{reduction}$: 쿼드 트리 그래프와 가시성 그래프의 거리 감소

각 단계 별 쿼드 트리 기반 최적 항로, 가시성 그래프, 가시성 그래프 기반 최적 항로(최단 거리) 결과를 Figure 9-8, Figure 9-9, Figure 9-10, Figure 9-11, Figure 9-12에 나타냈다. Table 9-3에서 각 기호가 나타내는 의미는 다음과 같다.

n_q: 쿼드 트리 그래프 기반 최적 항로의 변침점 개수

n_V: 가시성 그래프 기반 최적 항로의 변침점 개수

$t_{Q,path}$: 쿼드 트리 그래프에서 최적 항로 계산 시 소요 시간

$t_{V,path}$: 가시성 그래프에서 최적 항로 계산 시 소요 시간

$t_{V,general}$: 일반적인 가시성 그래프 생성 시간

$t_{V,improved}$: 쿼드 트리 기반 가시성 그래프 생성 시간

Table 9-3 Waypoint numbers and computation times for five cases in South sea of Korea (VG: visibility graph).

Case	n_q	n_V	$t_{Q,path}$(s)	$t_{V,path}$ (s)	$t_{V,improved}$ (s)	$t_{V,general}$ (s)
1	242	24	1.024	< 0.01	1.424	9.287
2	193	20	1.959	< 0.01	0.802	6.593
3	194	25	1.789	< 0.01	0.773	5.789
4	250	26	1.178	< 0.01	1.555	10.424
5	133	10	1.603	< 0.01	0.396	2.476

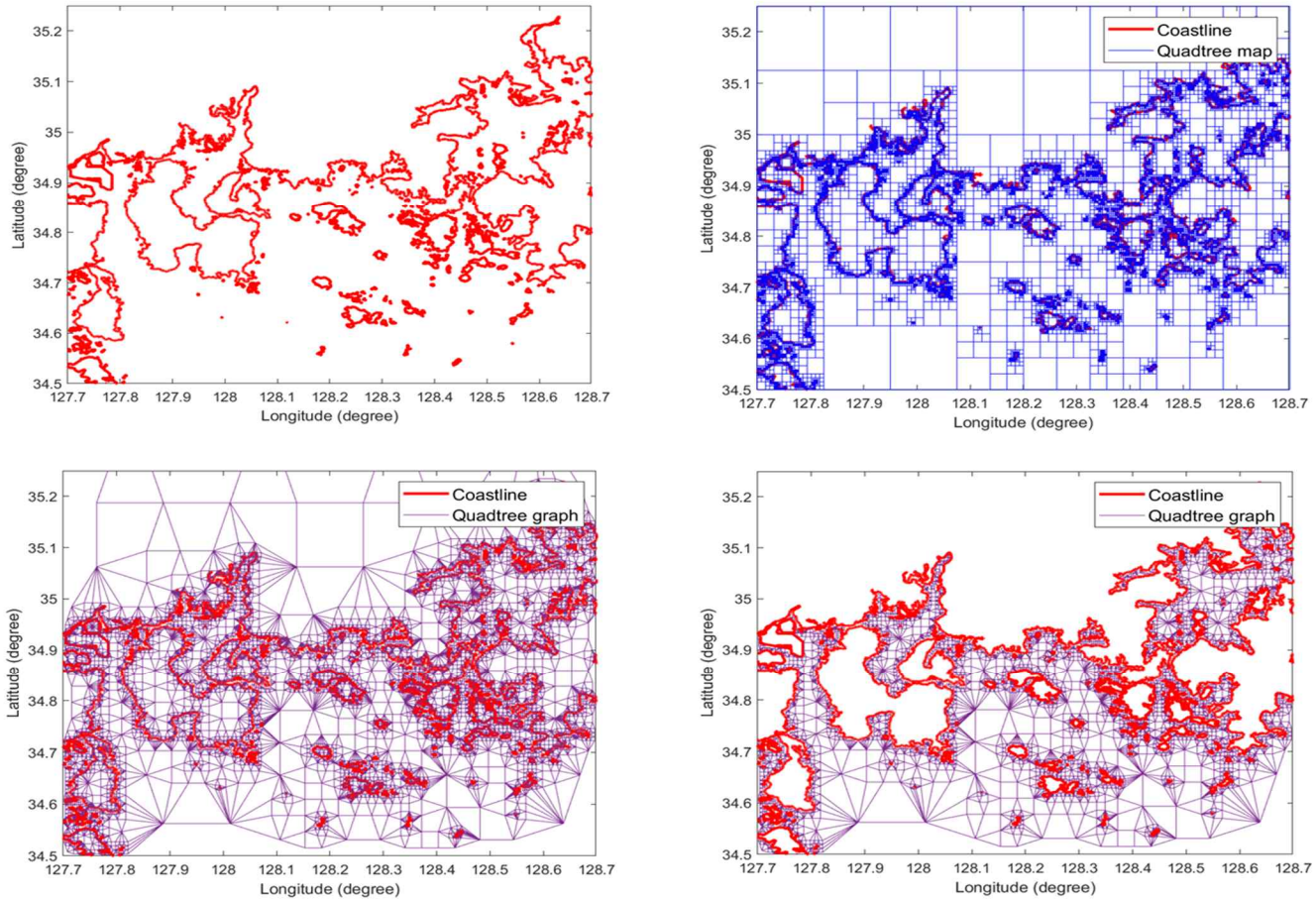

Figure 9-7 The process of quadtree graph generation in south sea

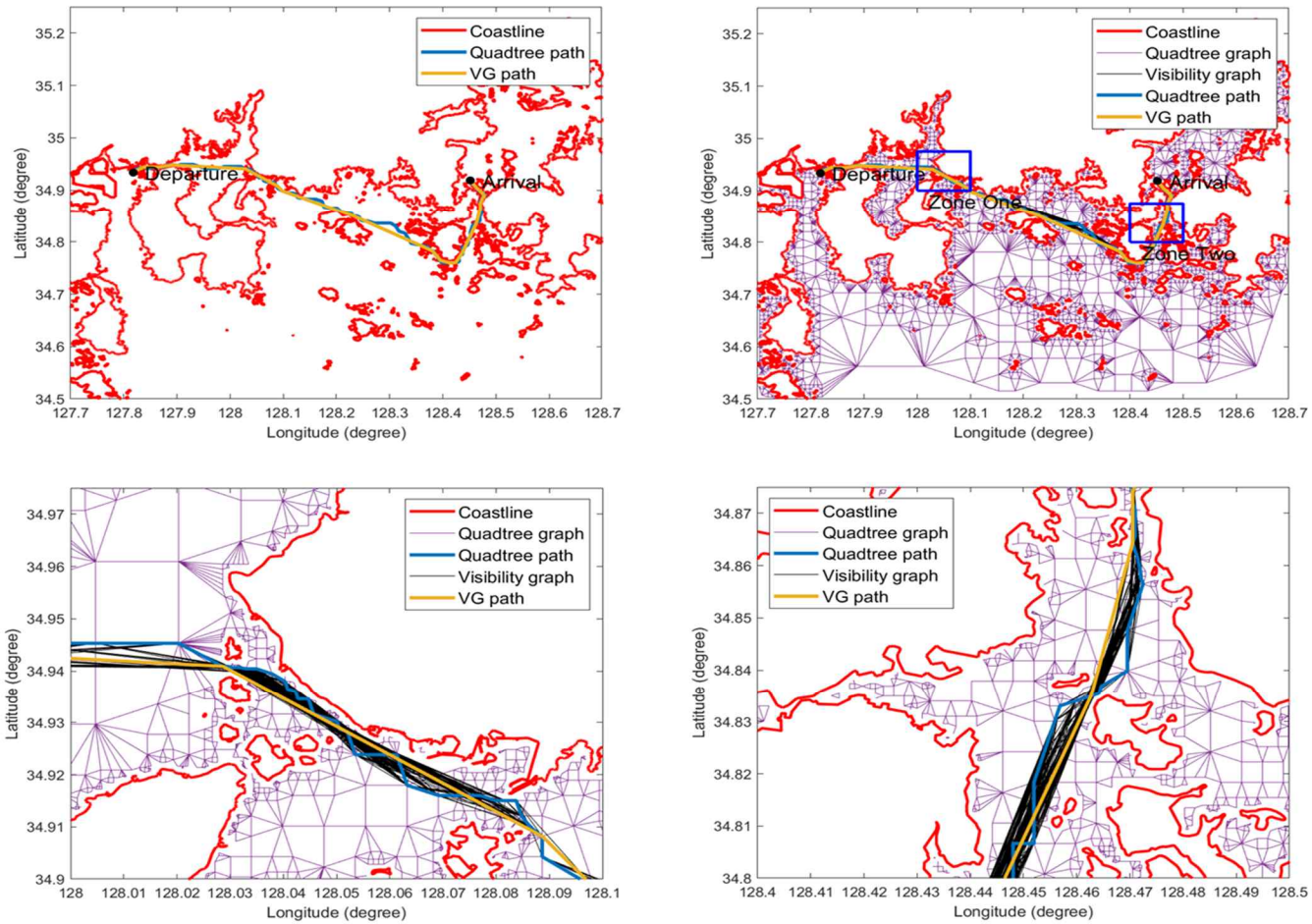

Figure 9-8 The path planning results of case 1 in the south sea of Korea

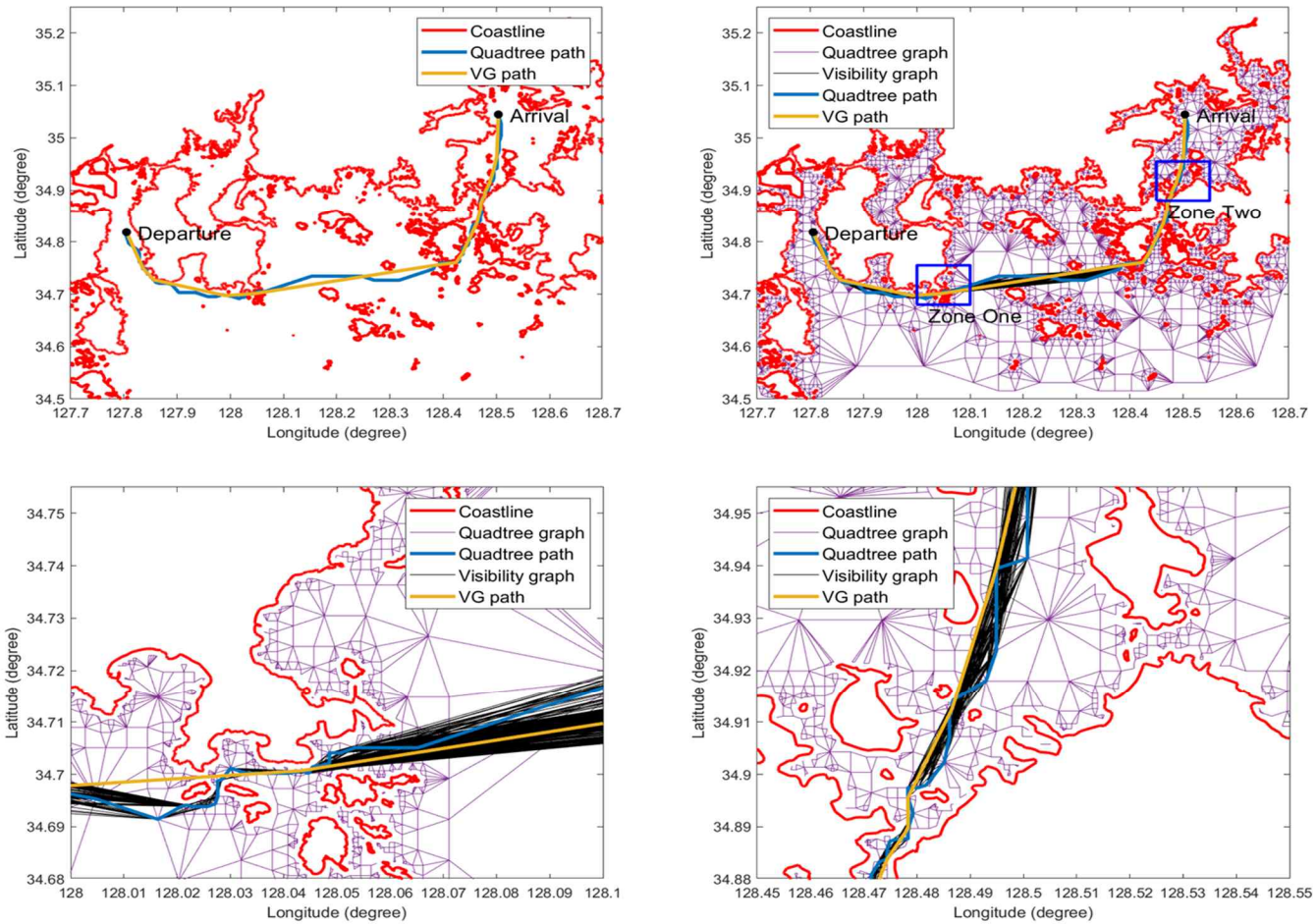

Figure 9-9 The path planning results of case 2 in south sea of Korea

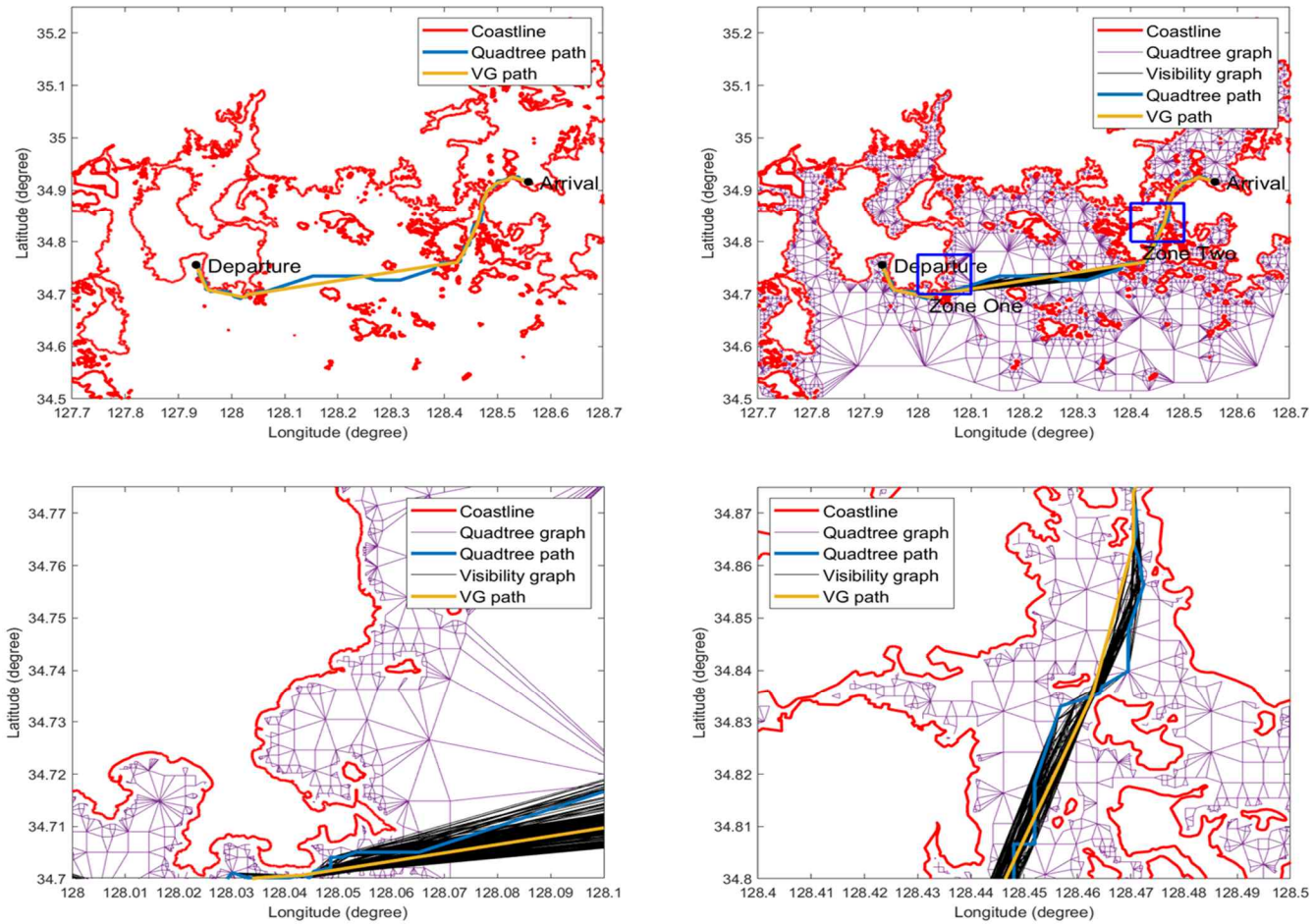

Figure 9-10 The path planning results of case 3 in south sea of Korea

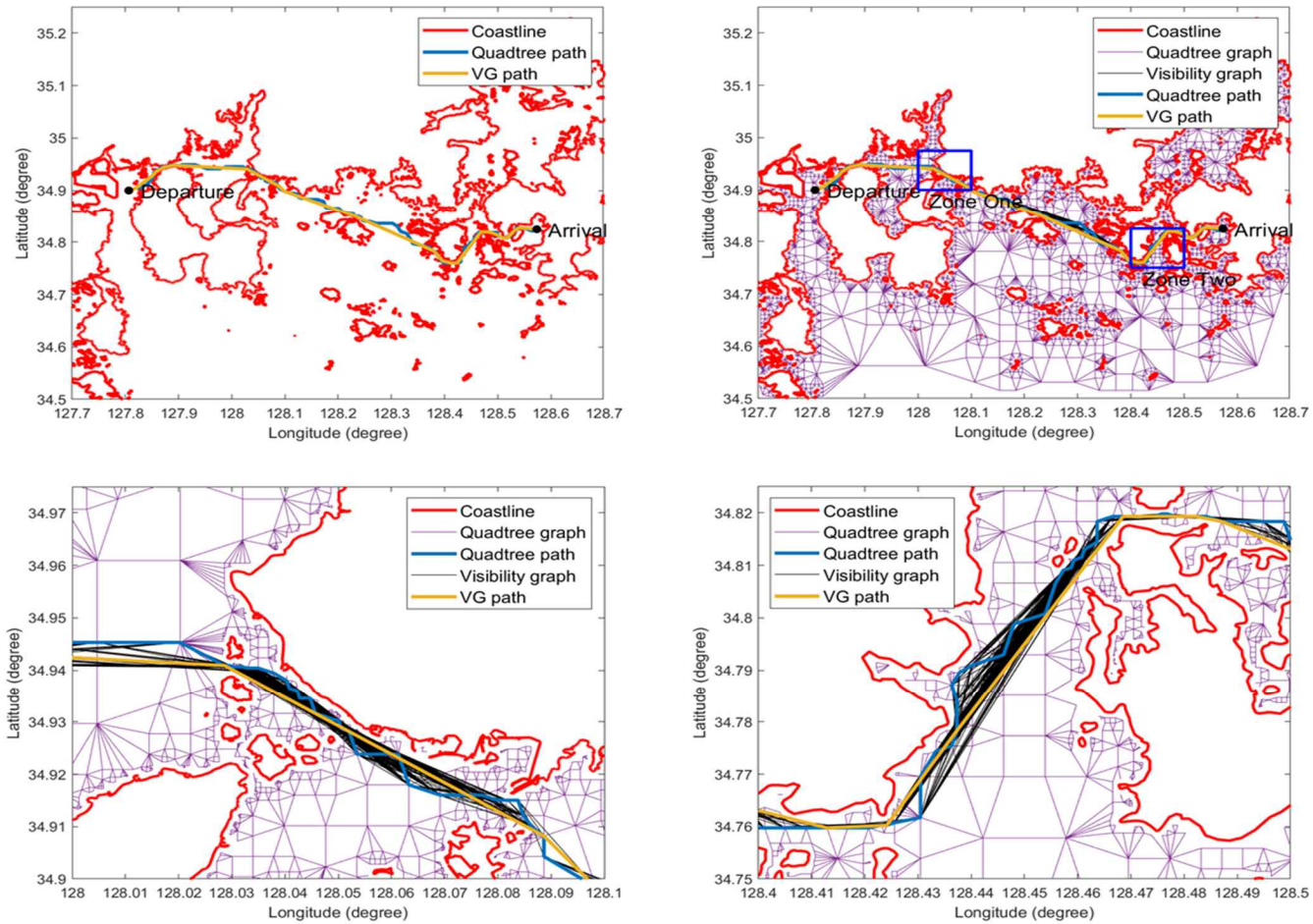

Figure 9-11 The path planning results of case 4 in south sea of Korea

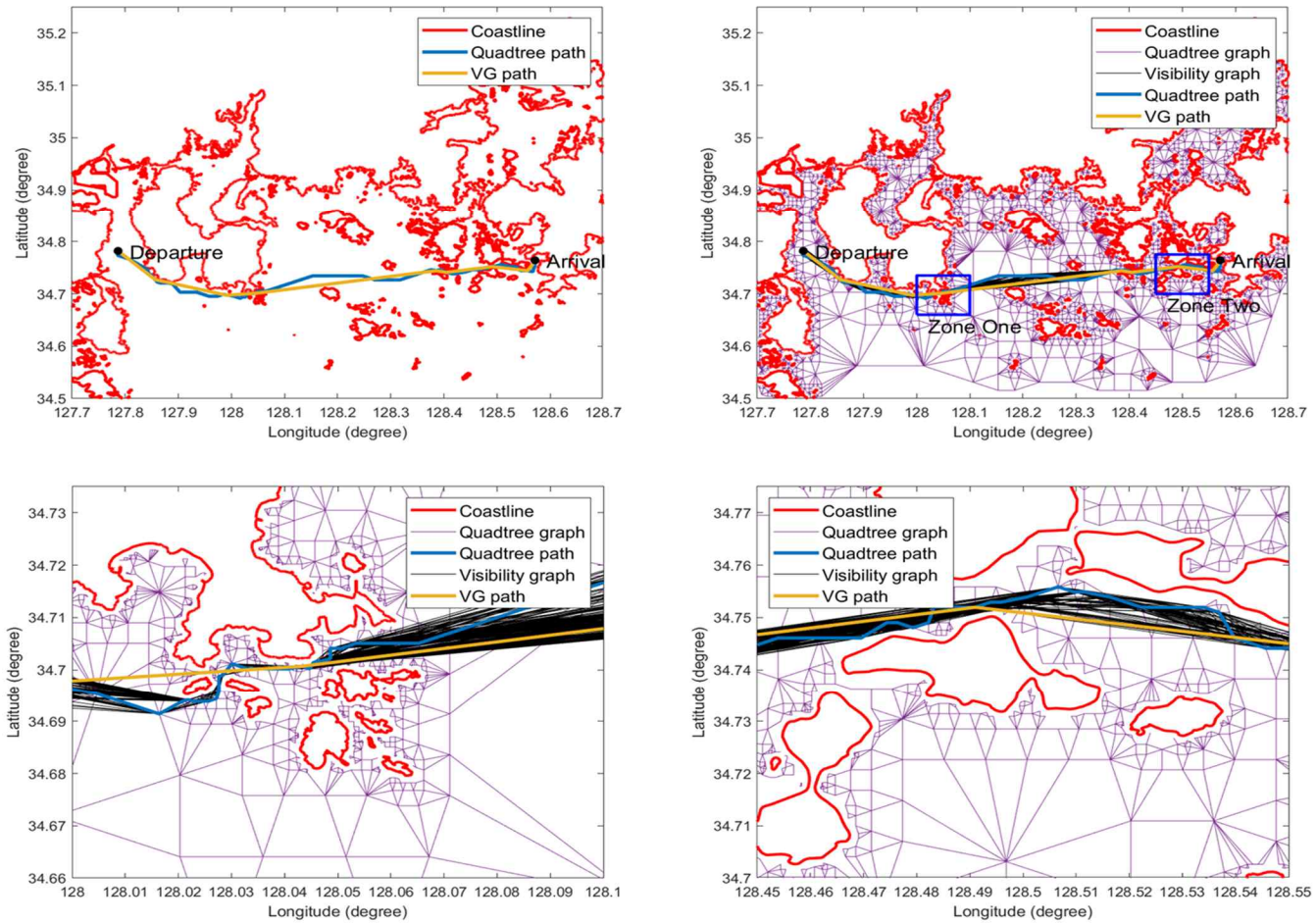

Figure 9-12 The path planning results of case 5 in south sea of Korea

또한, case 1 ~ 5까지 clearance 50, 100, 150, 200m에 따른 항로 변화를 Figure 9-13, Figure 9-14, Figure 9-15, Figure 9-16, Figure 9-17에 나타냈다.

Table 9-4 Comparison of path length with clearance in case 1,2,3 of south sea of Korea.

Case \ Clearance	50 m	75 m	100 m	125 m
1	80.539	80.906	81.053	81.567
2	97.021	97.144	97.147	97.868
5	73.610	73.785	74.447	85.154

Table 9-5 Comparison of path length with clearance in case 4,5 of south sea of Korea.

Case \ Clearance	50 m	100 m	150 m	200 m
4	80.570	82.828	84.592	97.691
5	75.888	76.611	77.739	77.683

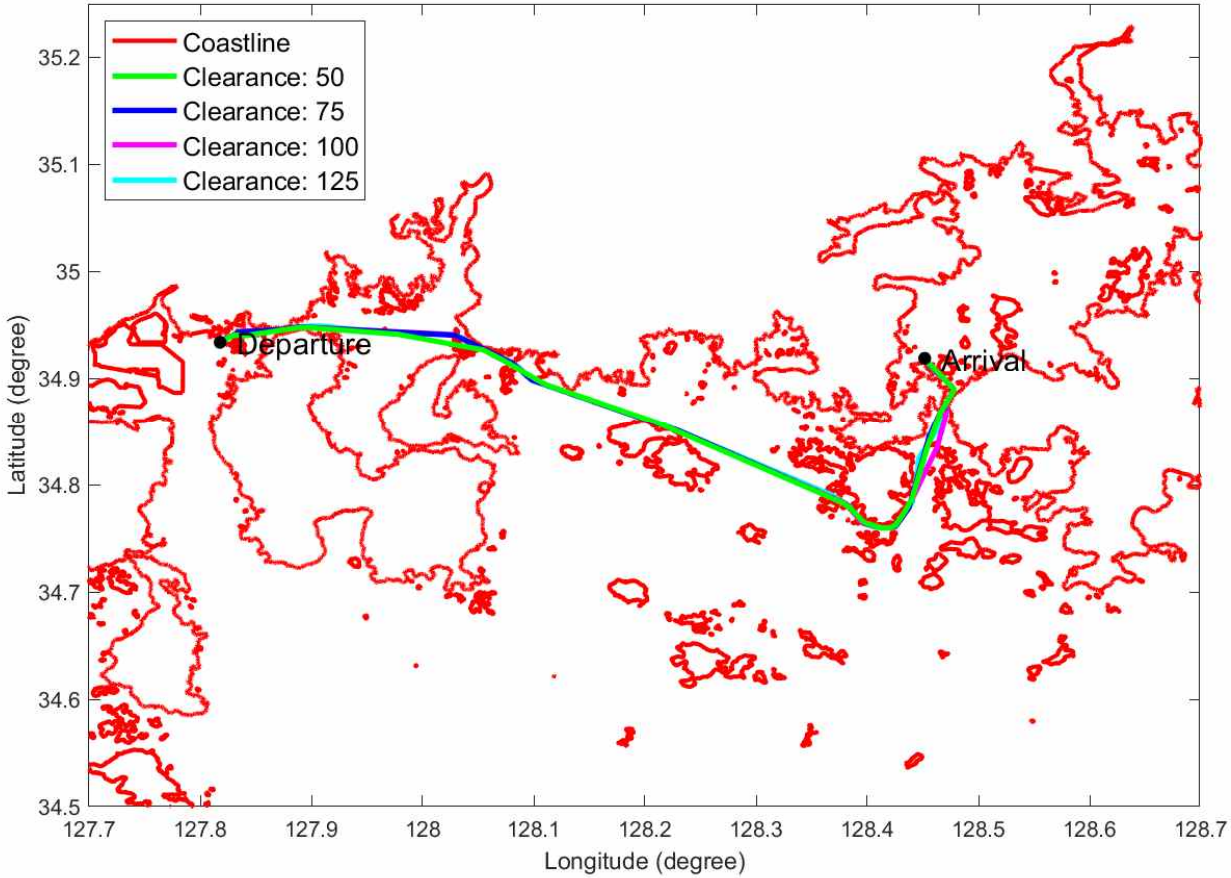

Figure 9-13 The path planning results with clearance 50, 75, 100, 125 in case 1

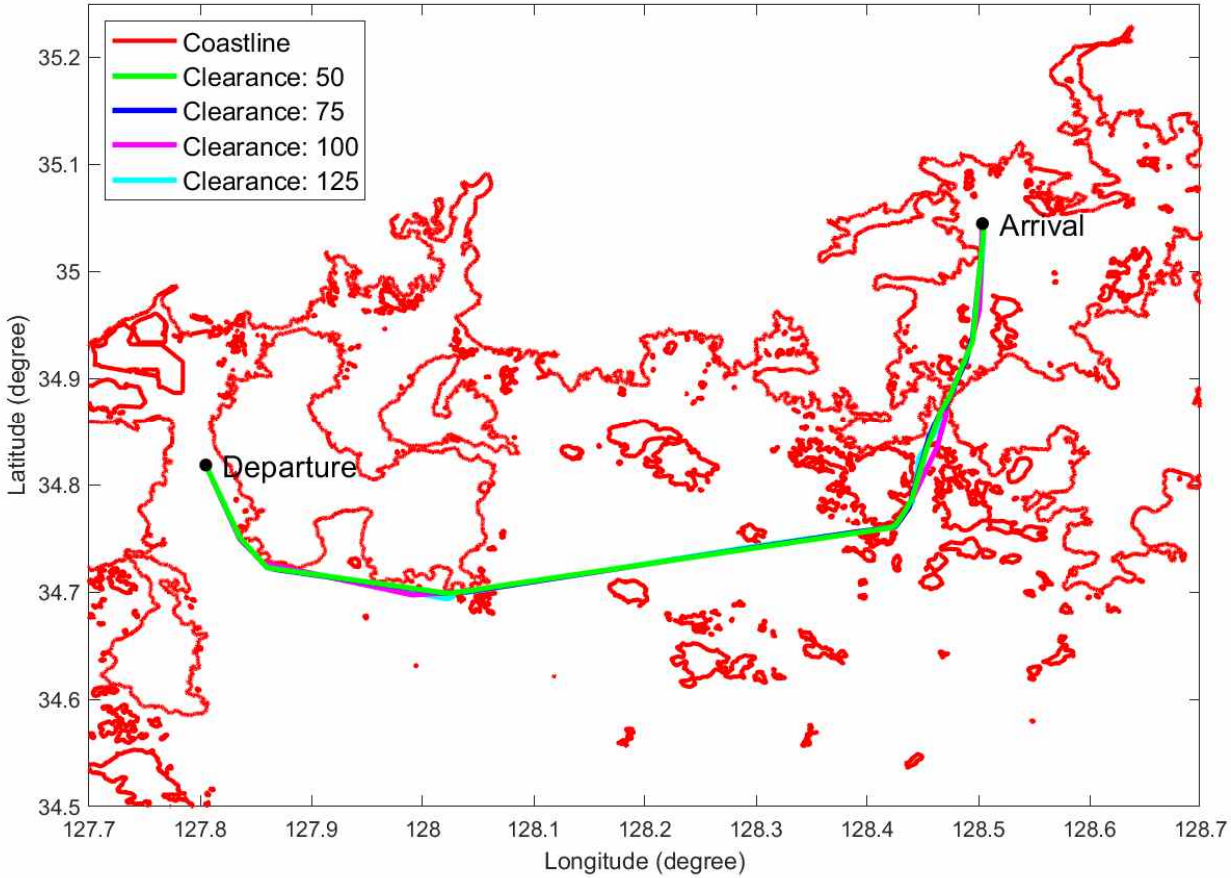

Figure 9-14 The path planning results with clearance 50, 75, 100, 125 in case 2

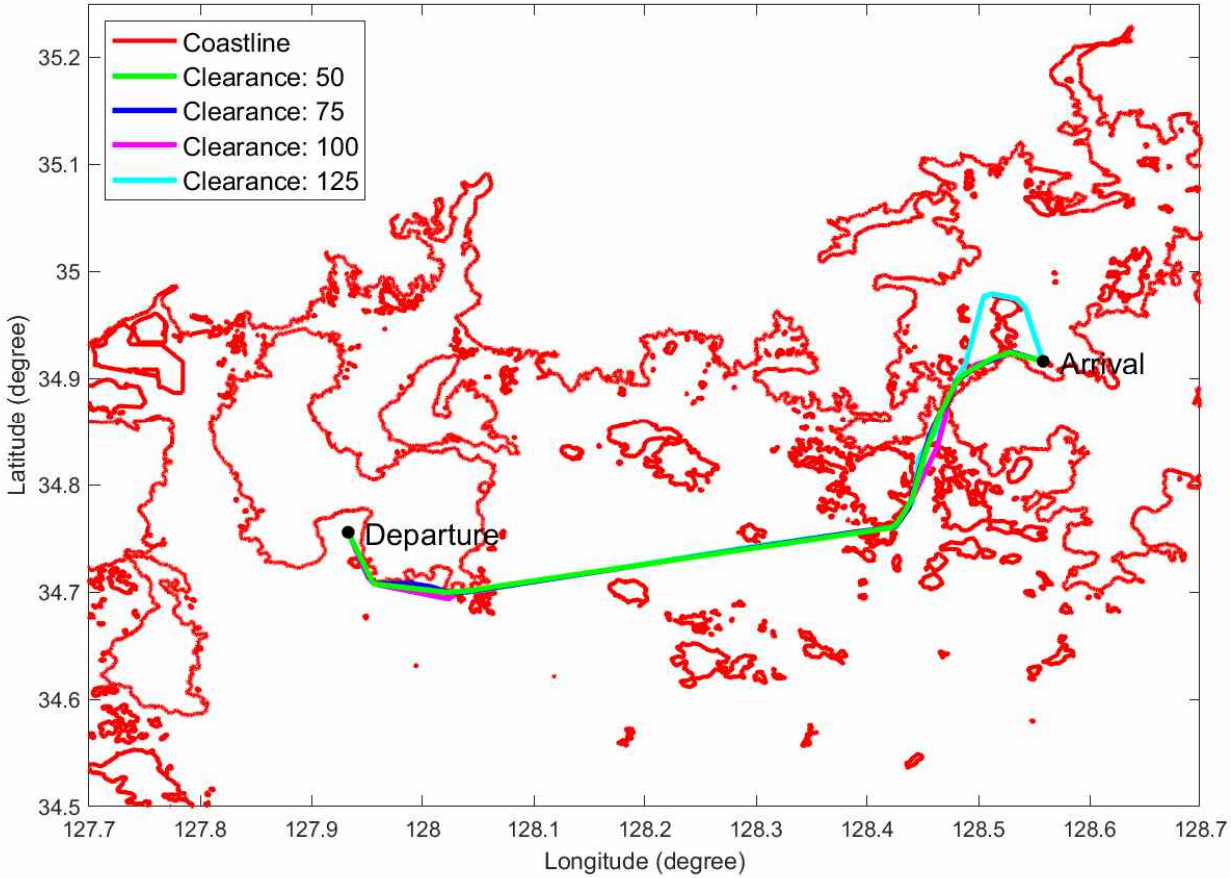

Figure 9-15 The path planning results with clearance 50, 75, 100, 125 in case 3

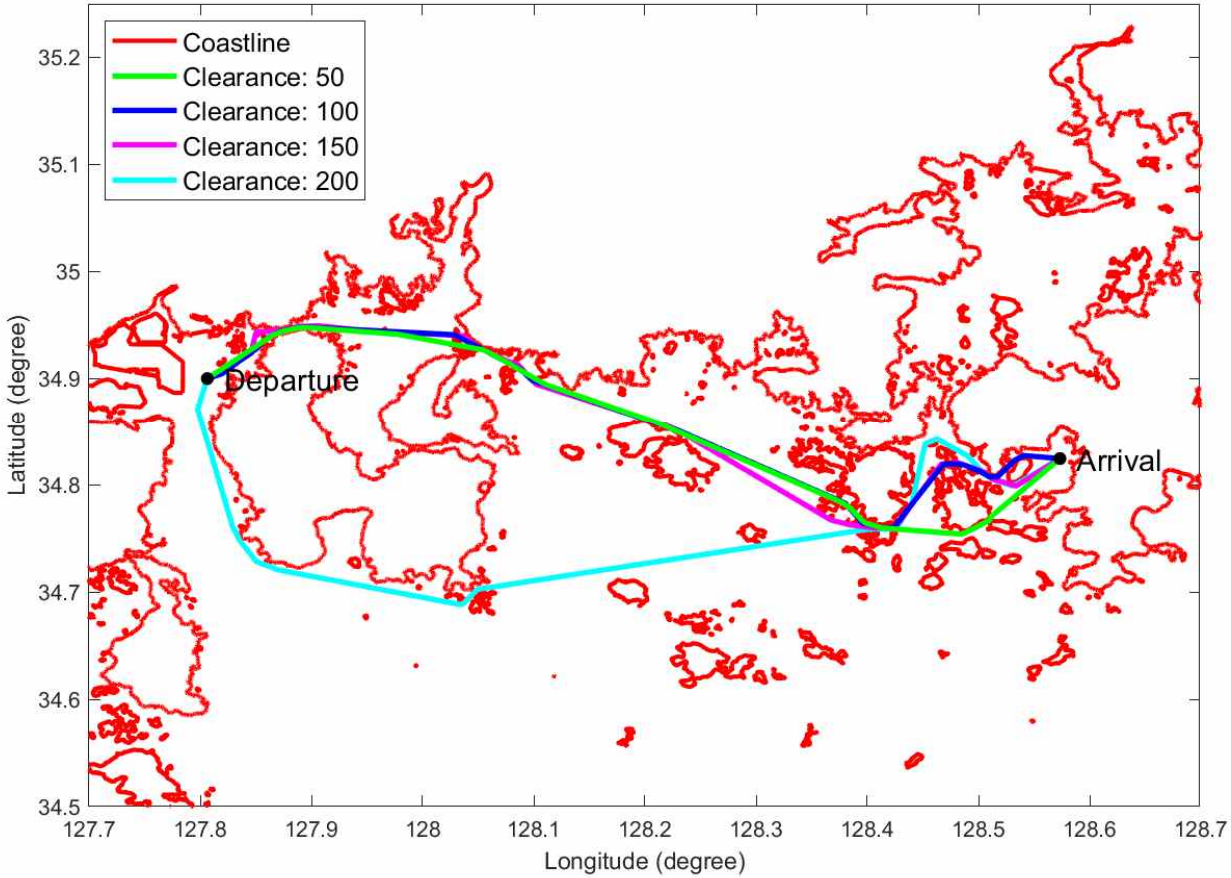

Figure 9-16 The path planning results with clearance 50, 100, 150, 200 in case 4

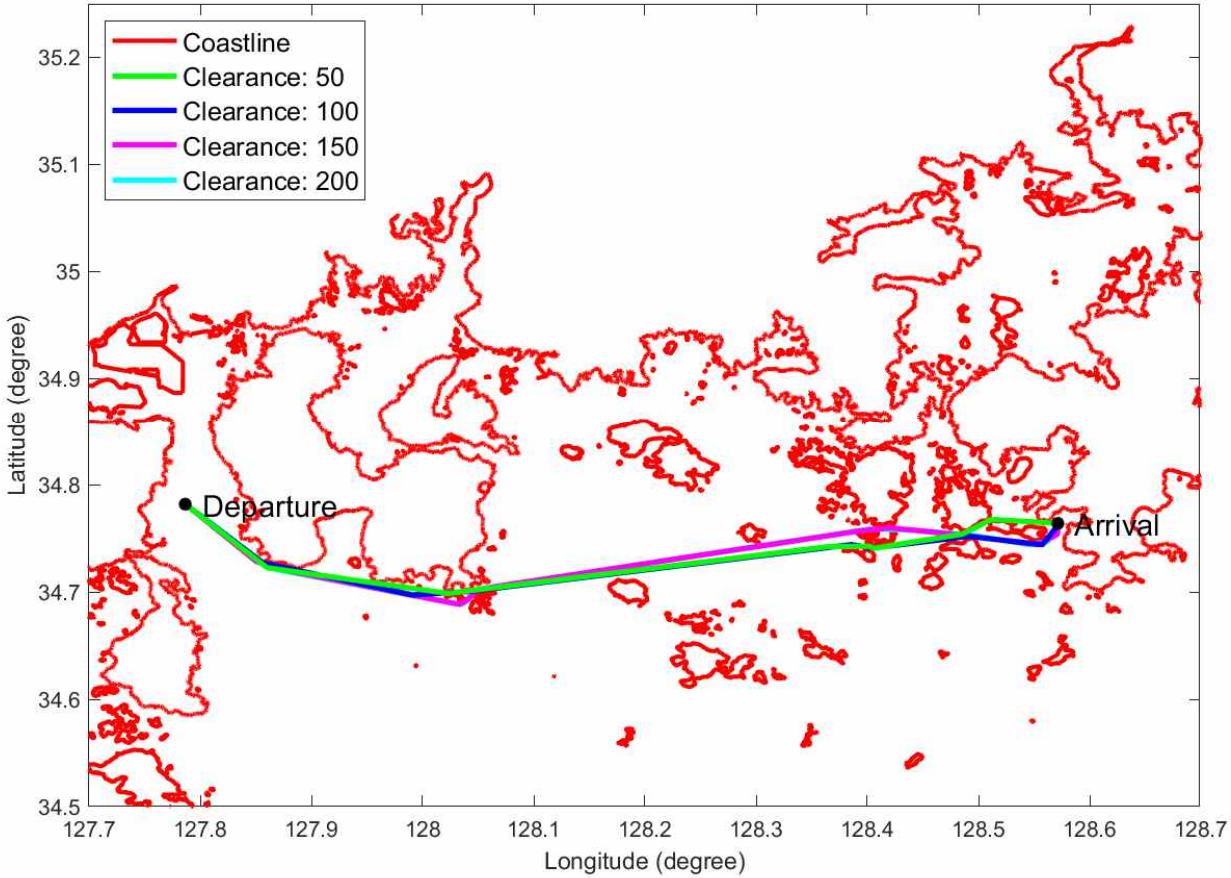

Figure 9-17 The path planning results with clearance 50, 100, 150, 200 in case 5

9.4 Simulation results in west sea of Korea

한국의 서해안에서 시뮬레이션을 수행한 결과이다. 각 단계 별 쿼드 트리 구성, 쿼드 트리 그래프, 장애물 교차 제거 후 쿼드 트리 그래프 결과를 아래에 나타냈다. 생성한 쿼드 트리 그래프 위에서 출발지, 도착지를 설정하여 5가지 경우를 Table 9-6와 Table 9-7에 비교 및 분석하였다. Table 9-6에서 각 기호가 나타내는 의미는 다음과 같다.

p_d: 출발지 (Departure point)

p_a: 도착지 (Arrival point)

l_Q: 쿼드 트리 기반 최적 항로 거리

l_{VG}: 가시성 그래프 기반 최적 항로 거리

$l_{reduction}$: 쿼드트리 그래프와 가시성 그래프의 거리 감소

Table 9-6 Comparison of quadtree and visibility-graph shortest paths for five cases in South Korea (VG: visibility graph).

Case	p_d	p_a	l_q(km)	l_{VG} (km)	$l_{reduction}$ (%)
1	(126.205, 34.758)	(125.984, 34.652)	27.767	25.583	7.87
2	(126.209, 34.911)	(125.870, 34.623)	51.589	47.205	8.50
3	(125.985, 34.855)	(126.148, 34.611)	35.675	32.576	8.69
4	(126.116, 34.920)	(126.093, 34.61)	39.496	37.708	4.53
5	(126.213, 34.738)	(125.929, 34.78)	31.220	28.337	9.23

각 단계 별 쿼드 트리 기반 최적 항로, 가시성 그래프, 가시성 그래프 기반 최적 항로 결과를 Figure 9-8, Figure 9-9, Figure 9-10, Figure 9-11, Figure 9-12에 나타냈다. Table 9-7에서 각 기호가 나타내는 의미는 다음과 같다.

n_q: 쿼드 트리 그래프 기반 최적 항로의 변침점 개수

n_V: 가시성 그래프 기반 최적 항로의 변침점 개수

$t_{Q,path}$: 쿼드 트리 그래프에서 최적 항로 계산 시 소요 시간

$t_{V,path}$: 가시성 그래프에서 최적 항로 계산 시 소요 시간

$t_{V,genera\ l}$: 일반적인 가시성 그래프 생성 시간

$t_{V,in\ porved}$: 쿼드 트리 기반 가시성 그래프 생성 시간

Table 9-7 Waypoint numbers and computation times for five cases in South Korea (VG: visibility graph).

Case	n_q	n_V	$t_{Q,path}$(s)	$t_{V,path}$ (s)	$t_{V,in\ proved}$ (s)	$t_{V,general}$ (s)
1	89	11	0.143	< 0.01	0.097	0.677
2	157	14	0.023	< 0.01	0.54	4.974
3	79	8	0.521	< 0.01	0.07	1.029
4	140	11	0.061	< 0.01	0.285	2.195
5	147	10	0.155	< 0.01	0.302	1.886

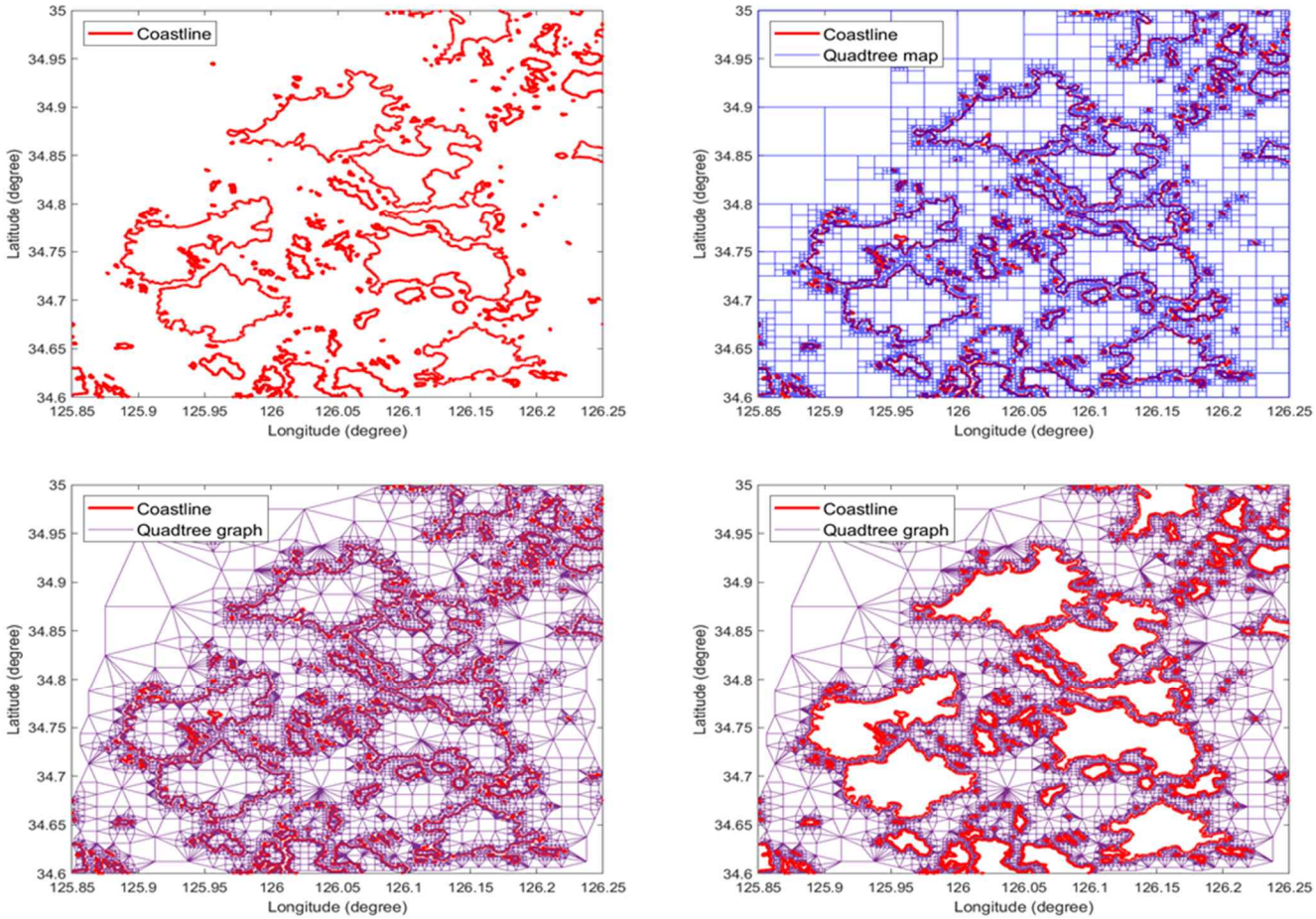

Figure 9-18 The process of quadtree graph generation in west sea

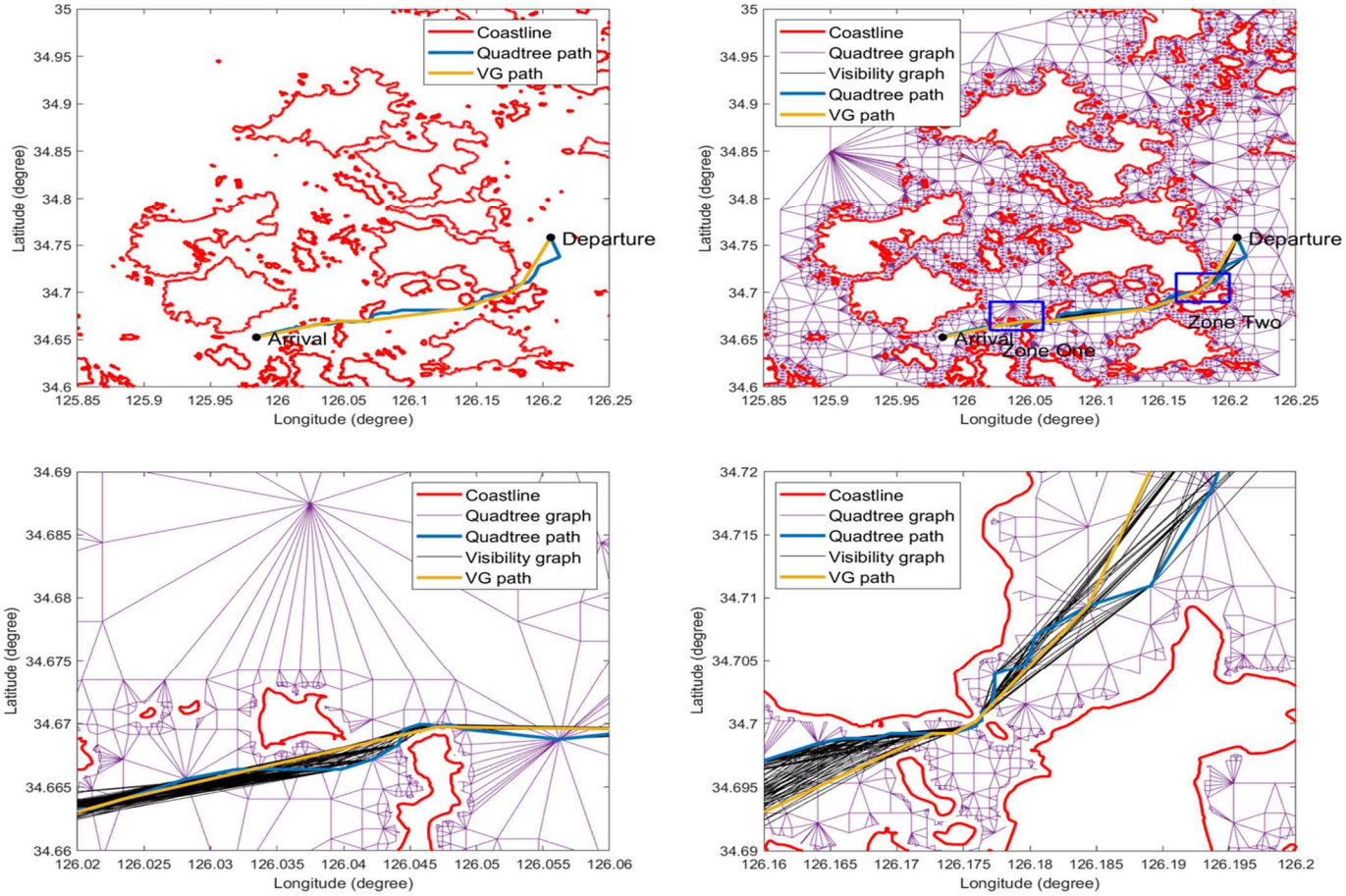

Figure 9-19 The path planning results of case 1 in west sea of Korea

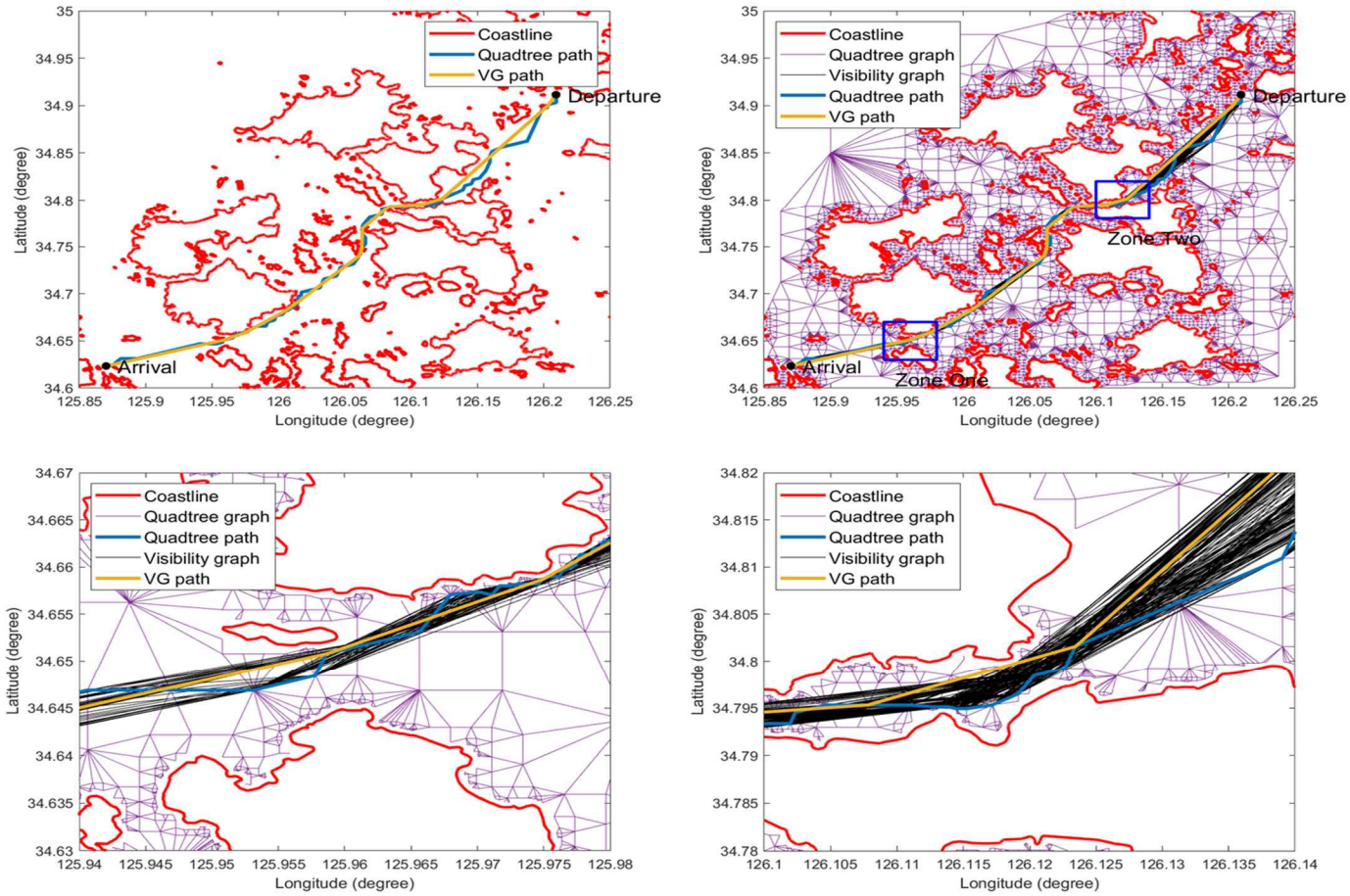

Figure 9-20 The path planning results of case 2 in west sea of Korea

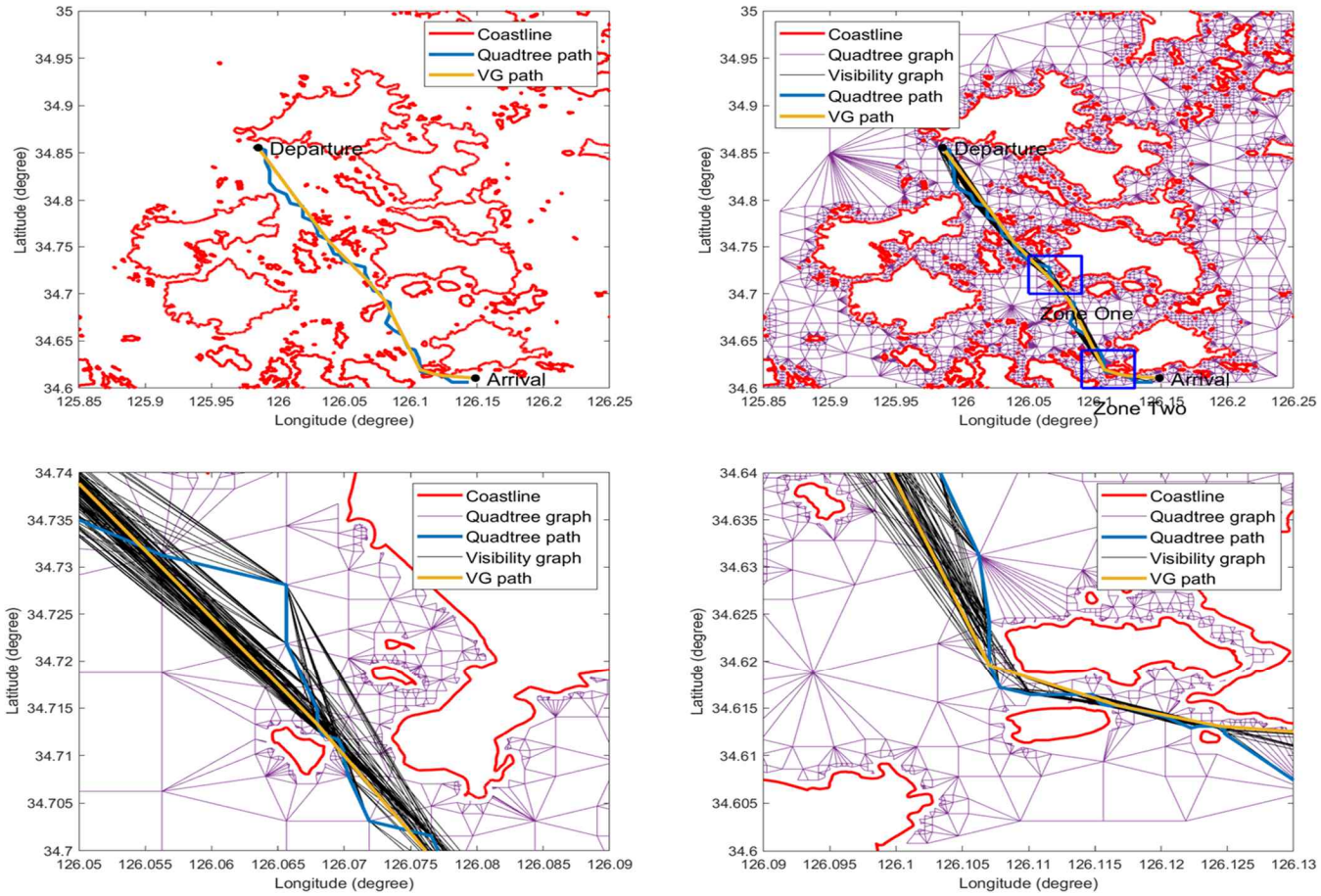

Figure 9-21 The path planning results of case 3 in west sea of Korea

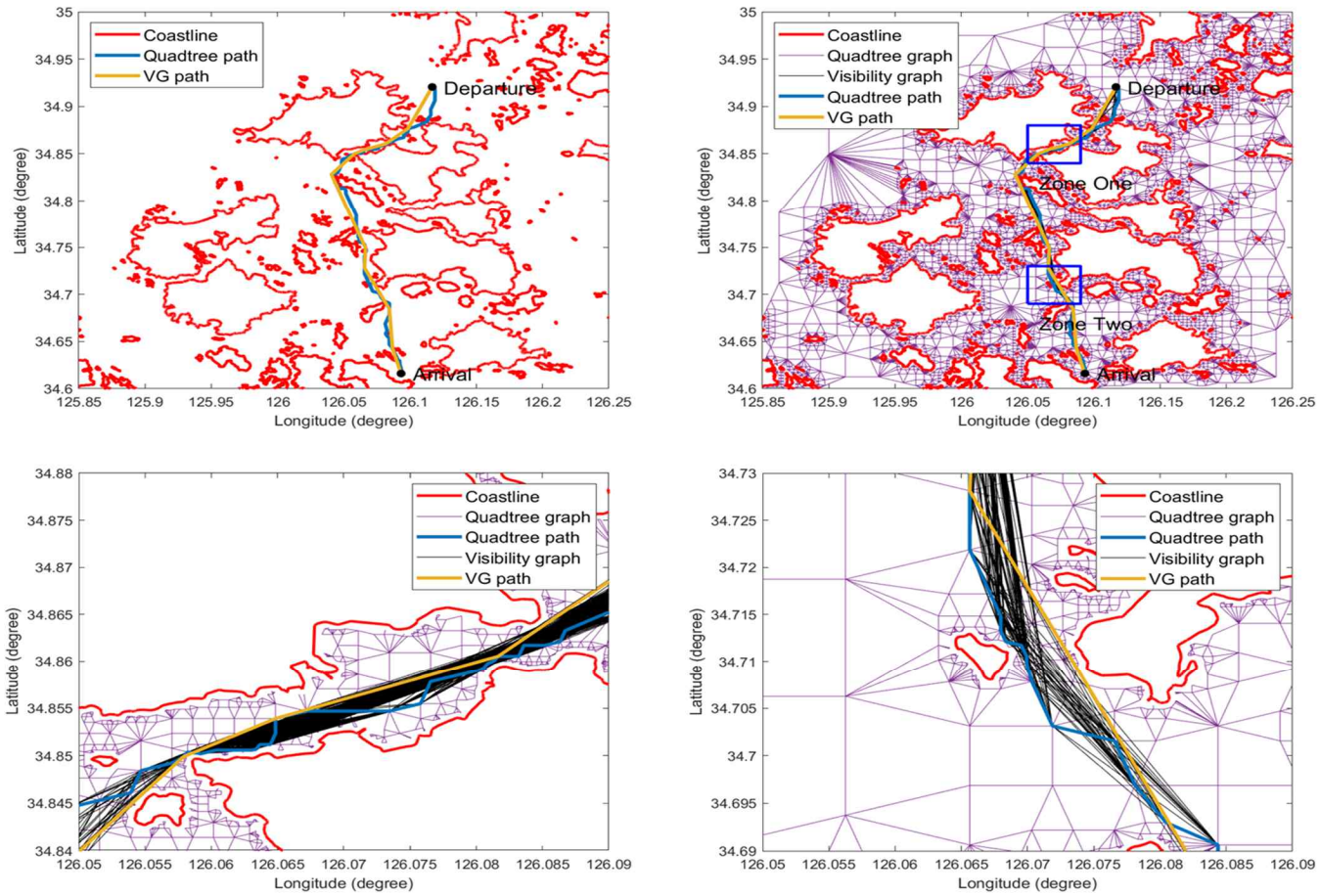

Figure 9-22 The path planning results of case 4 in west sea of Korea

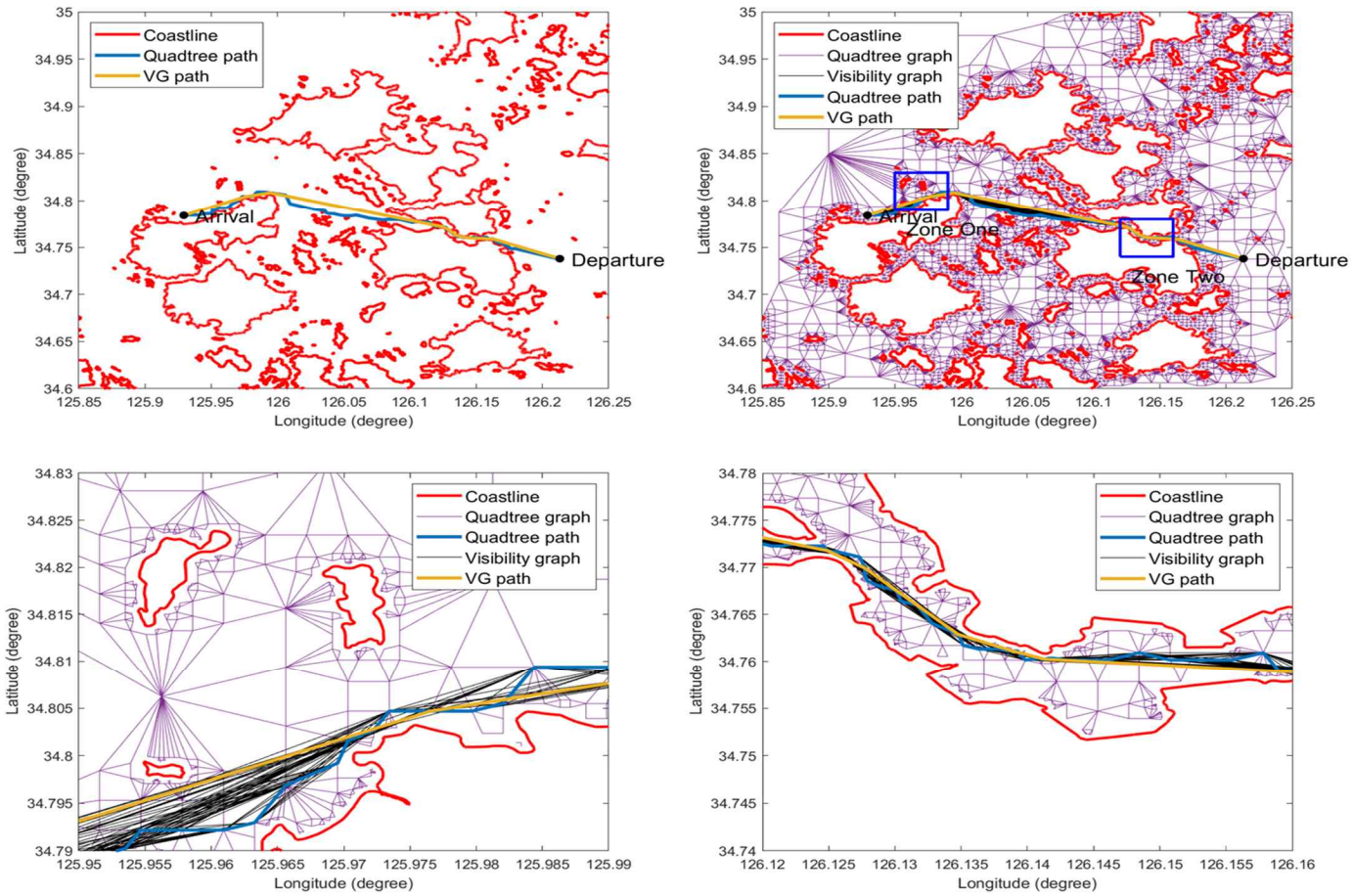

Figure 9-23 The path planning results of case 5 in west sea of Korea

또한, case 1 ~ 5까지 clearance 50, 100, 150, 200m에 따른 항로 변화를 Figure 9-24, Figure 9-25, Figure 9-26, Figure 9-27, Figure 9-28에 나타냈다.

Table 9-8 Comparison of path length (km) with clearance.

Case \ Clearance	50 m	100 m	150 m	200 m
1	25.583	28.696	29.155	30.004
2	47.205	47.549	48.251	56.758
3	32.576	32.630	33.479	33.655
4	37.078	37.113	37.277	42.106
5	28.337	29.008	35.218	39.554

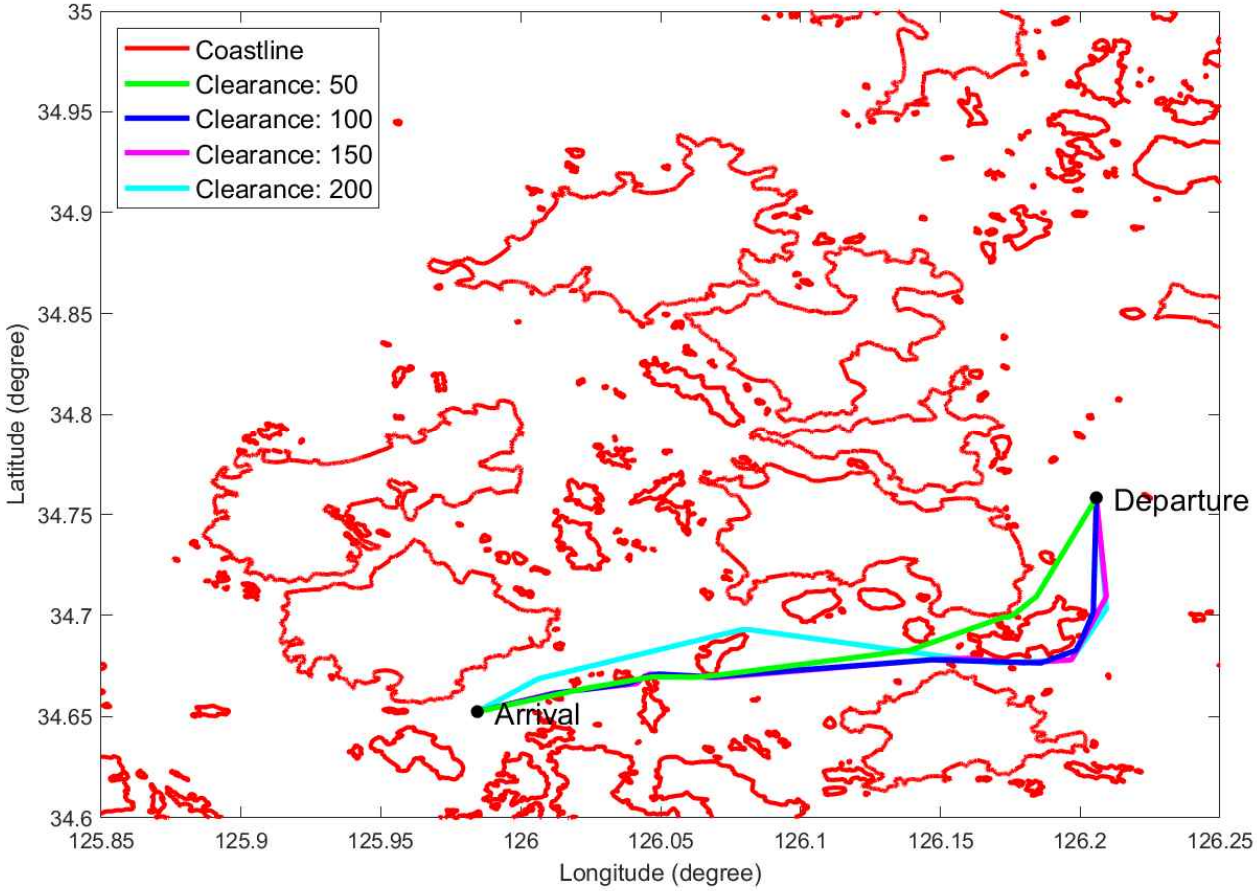

Figure 9-24 The path planning results with clearance 50, 100, 150, 200 in case 1

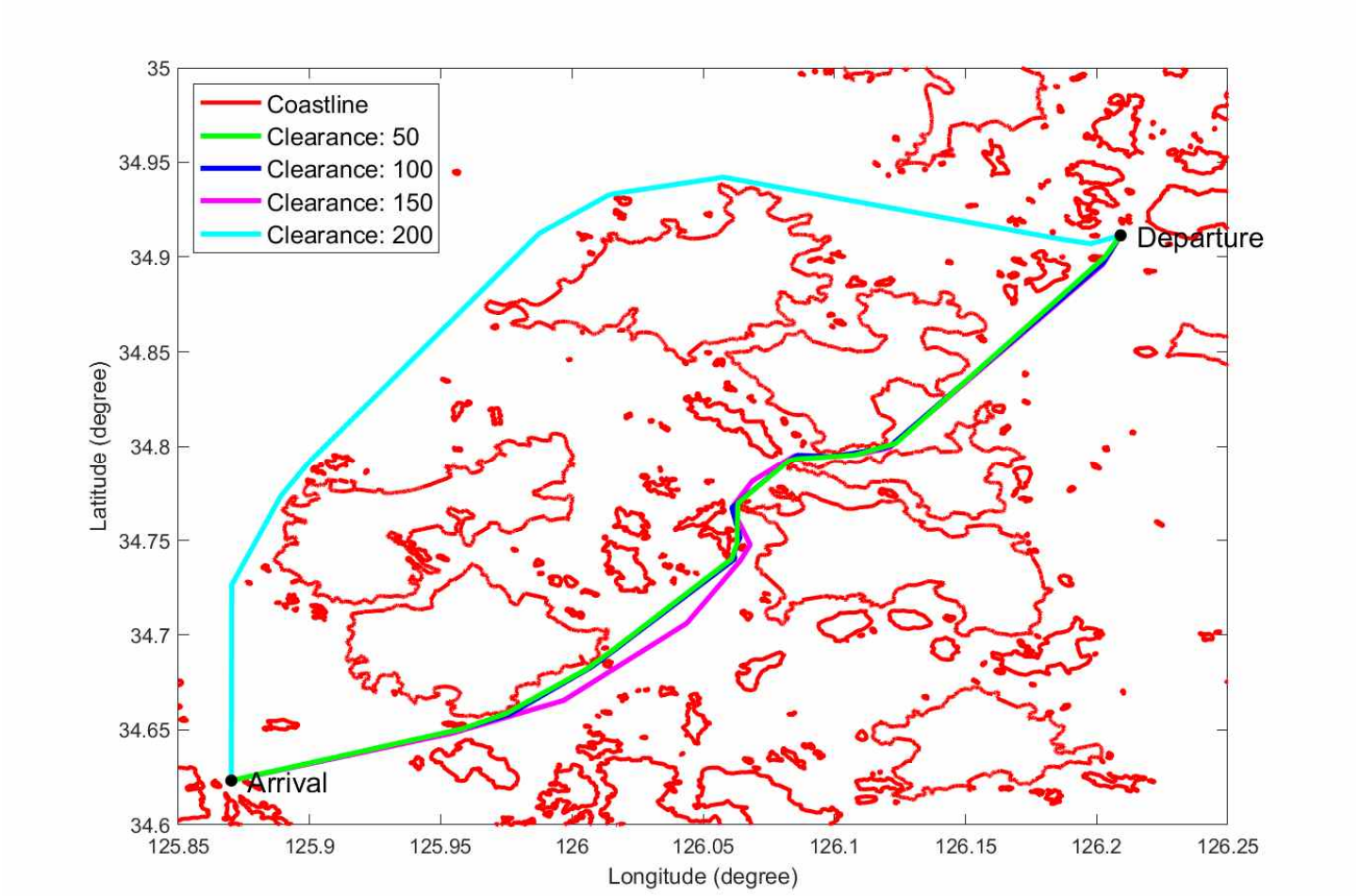

Figure 9-25 The path planning results with clearance 50, 100, 150, 200 in case 2

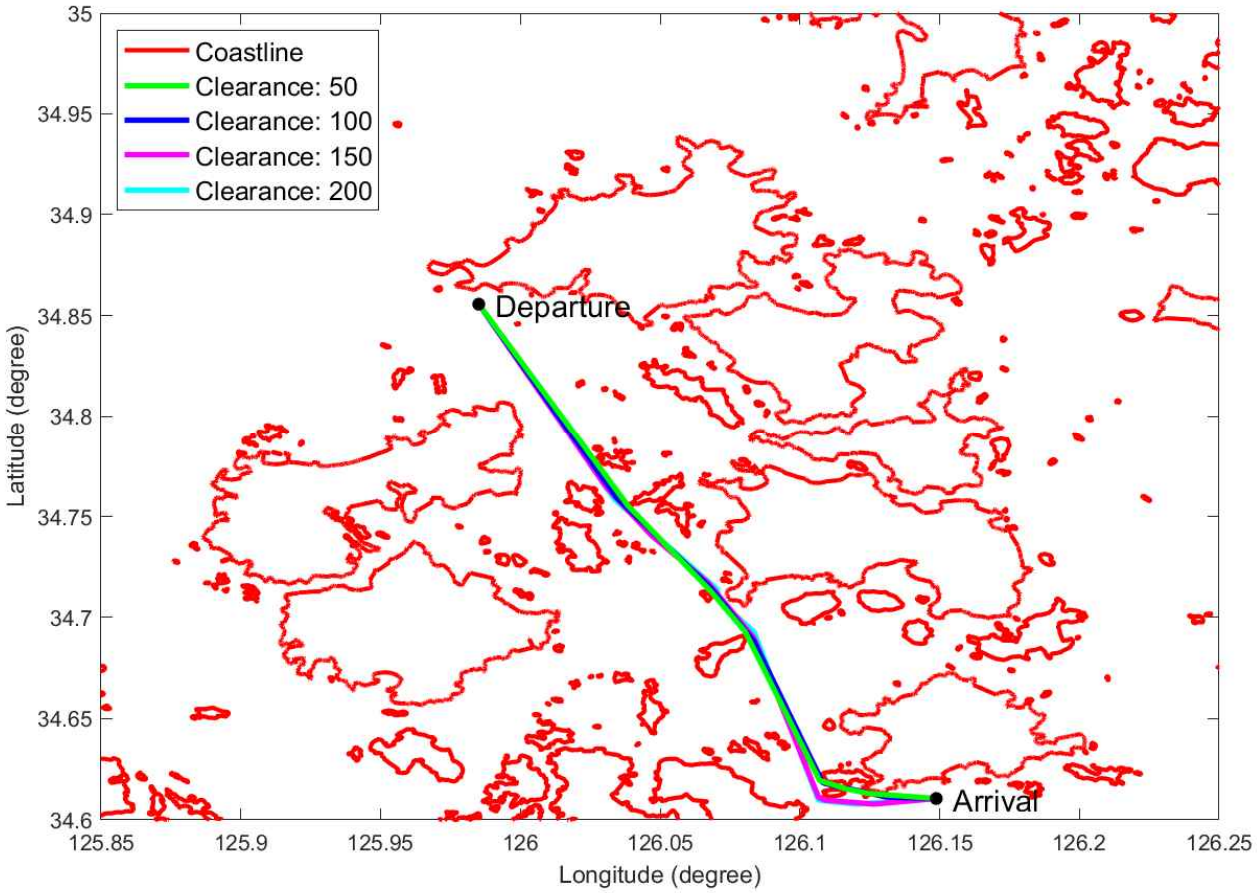

Figure 9-26 The path planning results with clearance 50, 100, 150, 200 in case 3

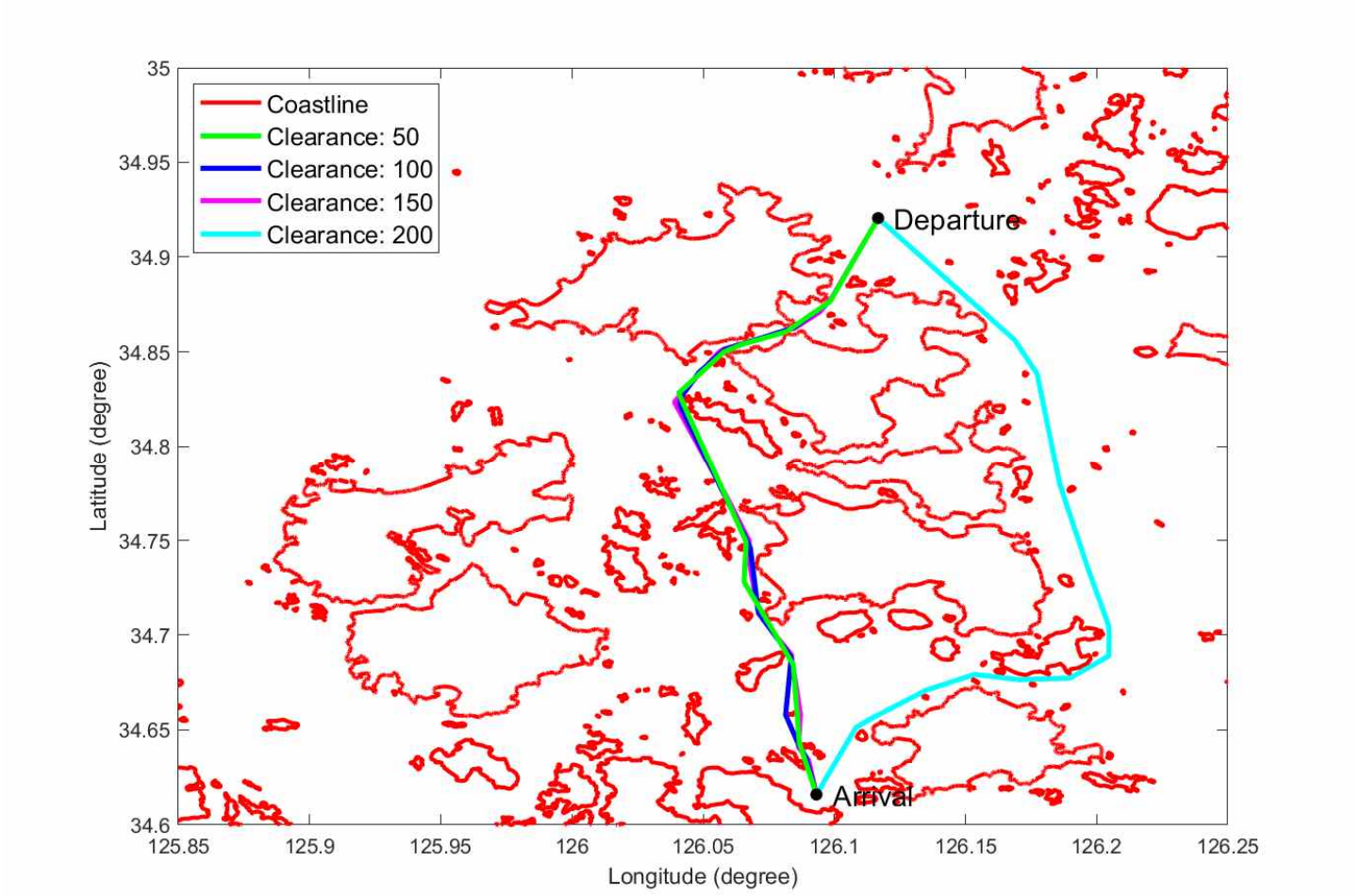

Figure 9-27 The path planning results with clearance 50, 100, 150, 200 in case 4

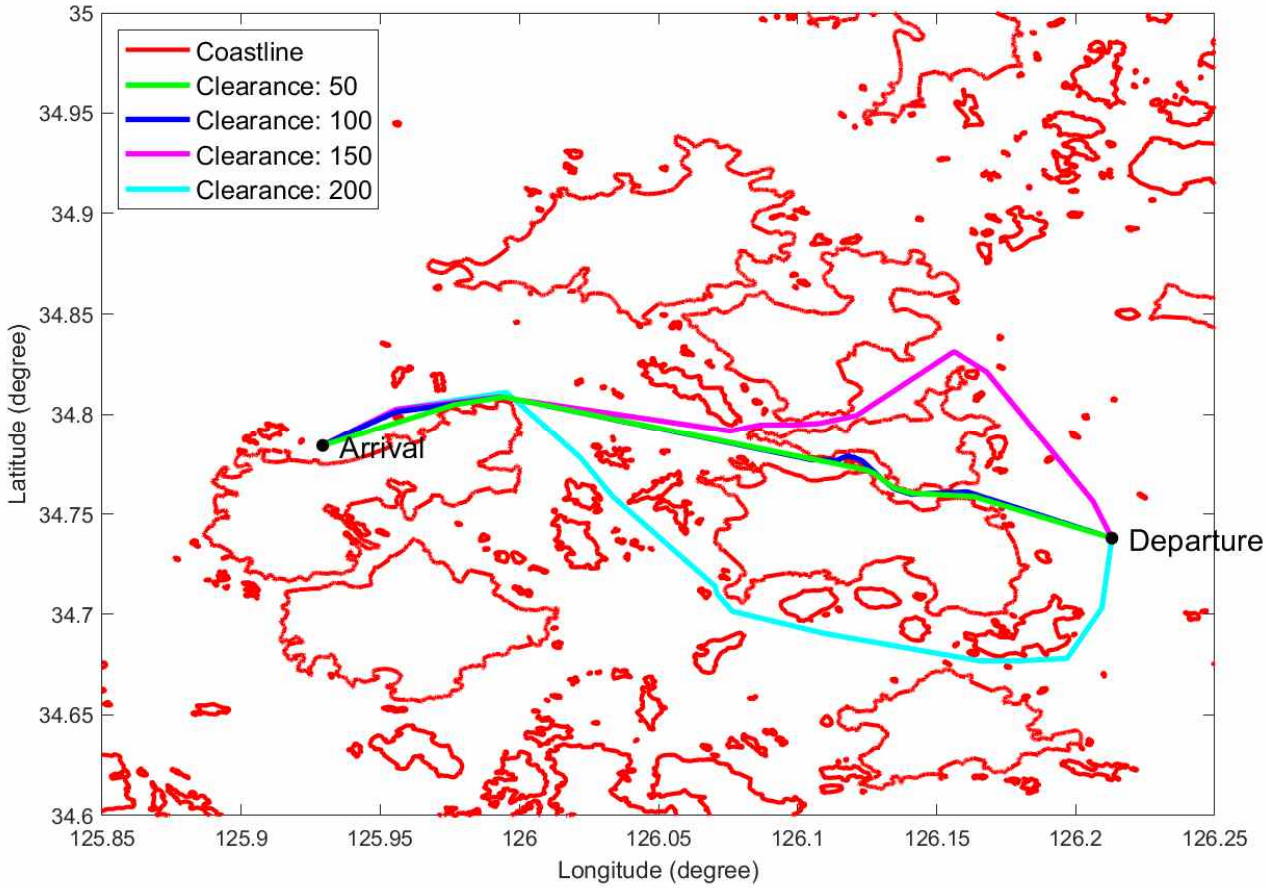

Figure 9-28 The path planning results with clearance 50, 100, 150, 200 in case 5

9.5 Fuel oil consumption minimization path results

본 절에서는 남해안과 서해안에서 임의의 두 항구를 선택하여 최적 항로 결과를 비교하고 분석한다. 먼저 9.5.1항에서는 서해안의 도초에서 발무기항을 가는 선박의 최단 거리 항로와 최소 연료 소모량 항로를 비교한다. 9.5.2항은 서해안의 옥도에서 자은도를 가는 선박의 최단 거리 항로와 최소 연료 소모량 항로를 비교한다. 9.5.3항에서는 남해안의 물건항에서 가조도 항을 가는 선박의 최단 거리 항로와 최소 연료 소모량 항로를 비교한다. 9.5.4항에서는 남해안의 노량항에서 거제도 대포항을 가는 선박의 최단 거리 항로와 최소 연료 소모량 항로를 비교한다.

9.5.1 First case in west sea of Korea

그래프를 생성하기 위해 서해안의 영역을 경도 (125.85° E, 126.25° E), 위도 (34.6° N, 35.0° N)을 선택했다. 먼저, 서해안의 영역에서 육지 정보를 가져온 결과는 Figure 9-29에 나타냈다. 조위 정보를 반영하여 수심 정보를 가져온 결과는 Figure 9-30에 나타냈다. Polygon offset을 이용하여 장애물들을 100m만큼 확장시킨 후 PMR 쿼드 트리를 생성한 결과를 Figure 9-31에 나타냈다. 쿼드 트리 기반으로 4방향으로 인접한 격자에 대해 간선을 연결한 그래프 결과는 Figure 9-32에 나타냈다. 이 때 장애물과 겹치거나 장애물 내부에 있는 그래프의 간선은 모두 제거하였다.

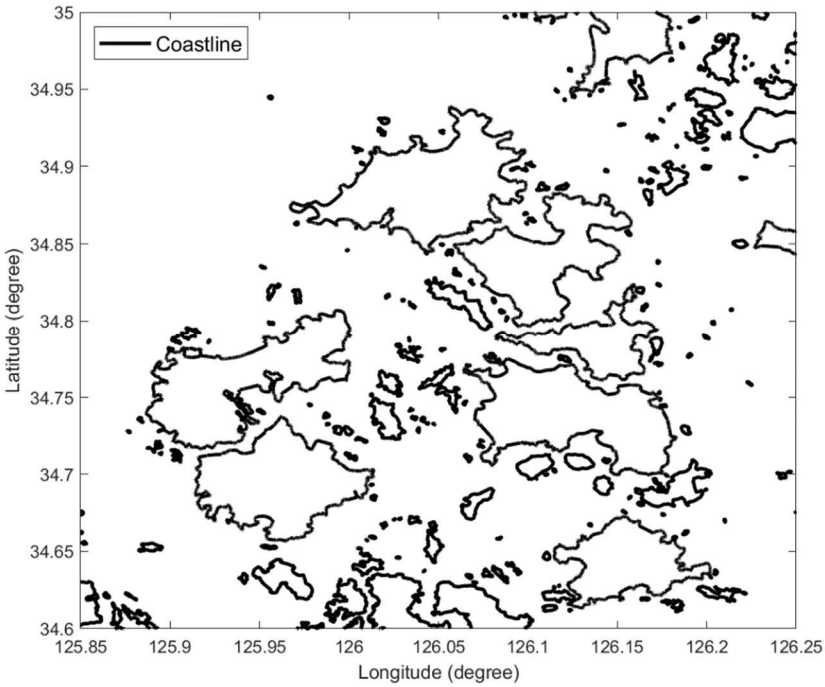

Figure 9-29 Land areas in west sea

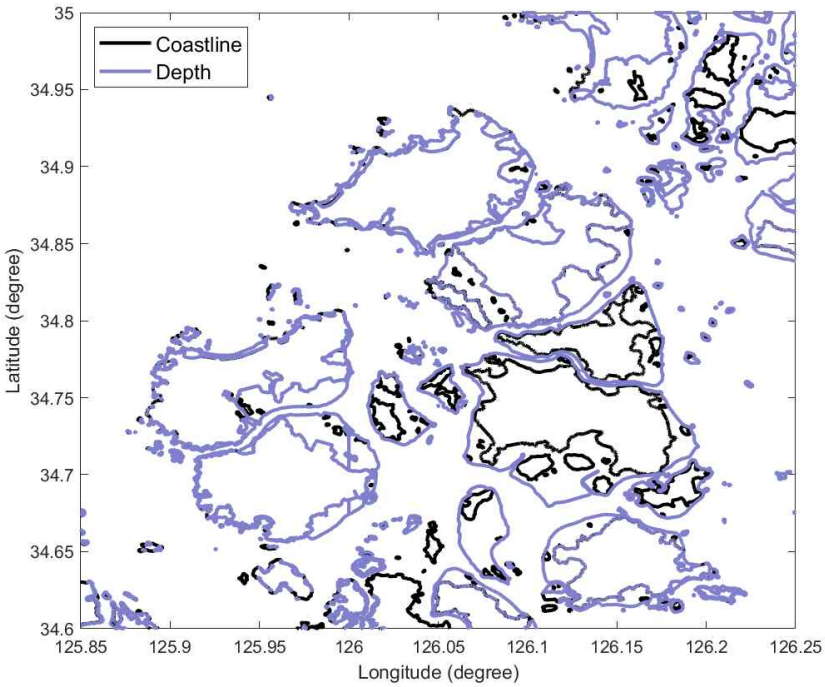

Figure 9-30 Land and depth areas in west sea

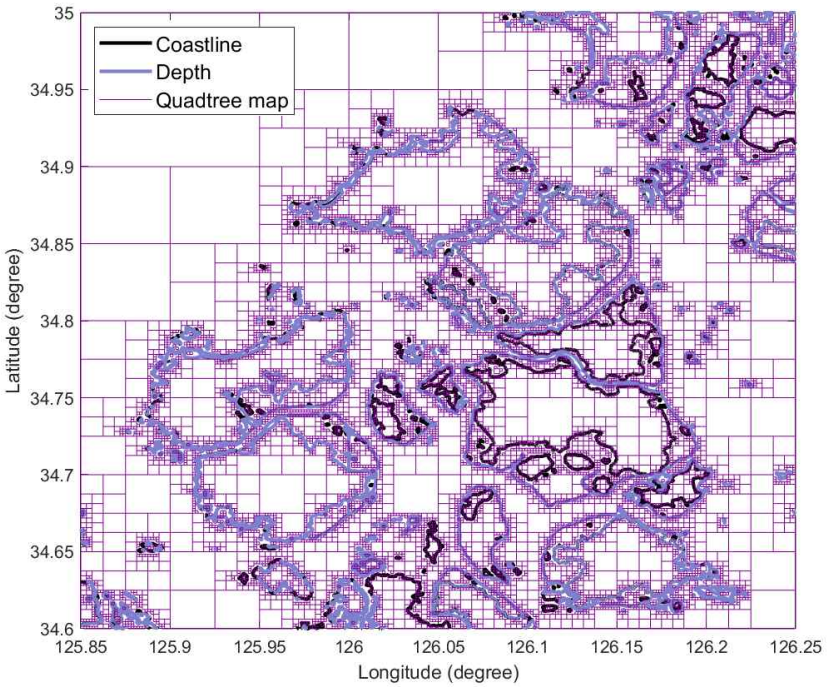

Figure 9-31 PMR quadtree generation results in west sea

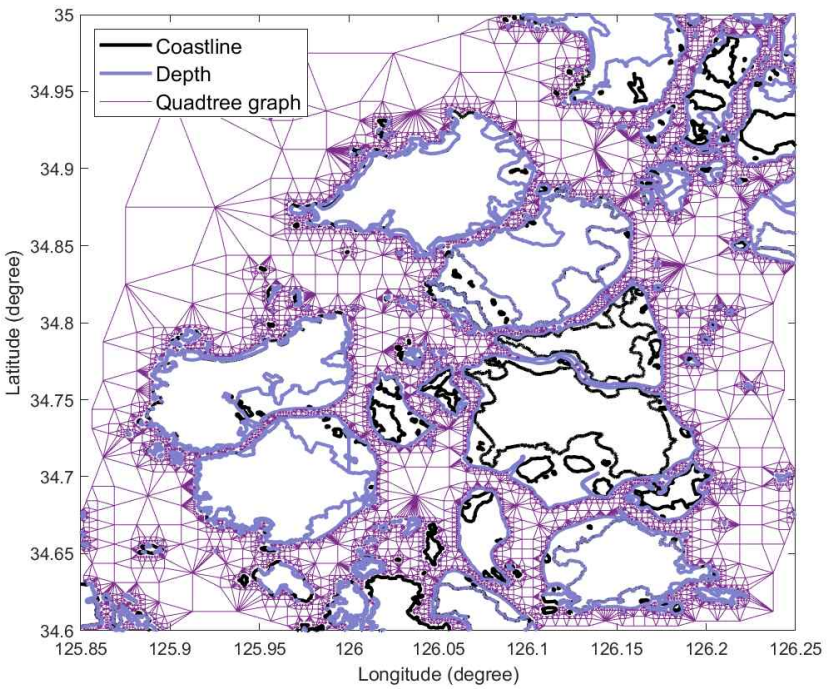

Figure 9-32 Quadtree based graph in west sea

서해안의 도초에서 발무기항을 가는 항로에 대해서 최단 거리 항로와 최소 연료 소모량 항로를 탐색하였다. 선박이 항해를 시작하는 시간은 한국표준시로 2020년 12월 23일 오전 0시이며, KMA와 KHOA 데이터를 사용하였다. 항해 시간이 짧기 때문에 기상 정보는 선박의 출항 시각 정보를 그대로 사용한다. 앞서 생성한 쿼드 트리 그래프에 출항지와 입항지 위치를 그래프에 삽입한다. 출항지와 입항지에 대해 Dijkstra 알고리즘을 수행하여 최단 거리 항로와 최소 연료 소모량 항로를 계산한다. 그 결과를 Figure 9-33에 나타냈으며, 최단 거리 항로와 최소 연료 소모량 항로가 다르게 나온 것을 확인하였다. 또한, 그림에서 볼 수 있듯이 두 항로 모두 불필요한 점이 많이 생성된 것을 확인할 수 있다. 불필요한 점이 많으면 항로의 거리가 길어지며, 선박의 실제 항로와 많이 다를 수 있으므로 불필요한 점을 제거해야 한다. 불필요한 점을 제거하기 위해 가시성 그래프를 사용하였다. 가시성 그래프의 간선 가중치는 최단 거리 항로는 거리에서는 거리로, 최소 연료 소모량 항로에서는 연료 소모량으로 설정하였다. 각각 가시성 그래프를 적용한 결과를 Figure 9-34에 나타냈다.

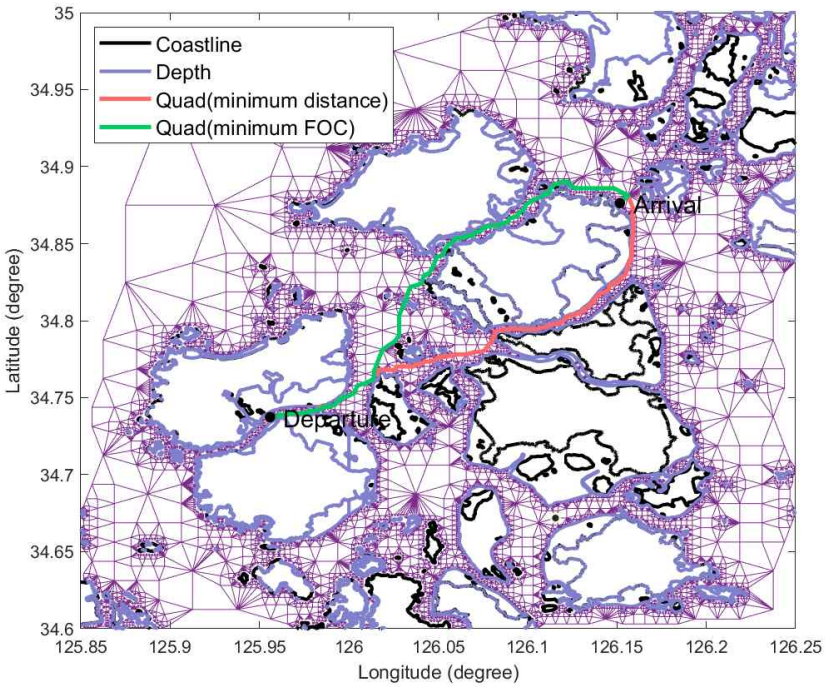

Figure 9-33 Minimum distance and FOC quadtree path

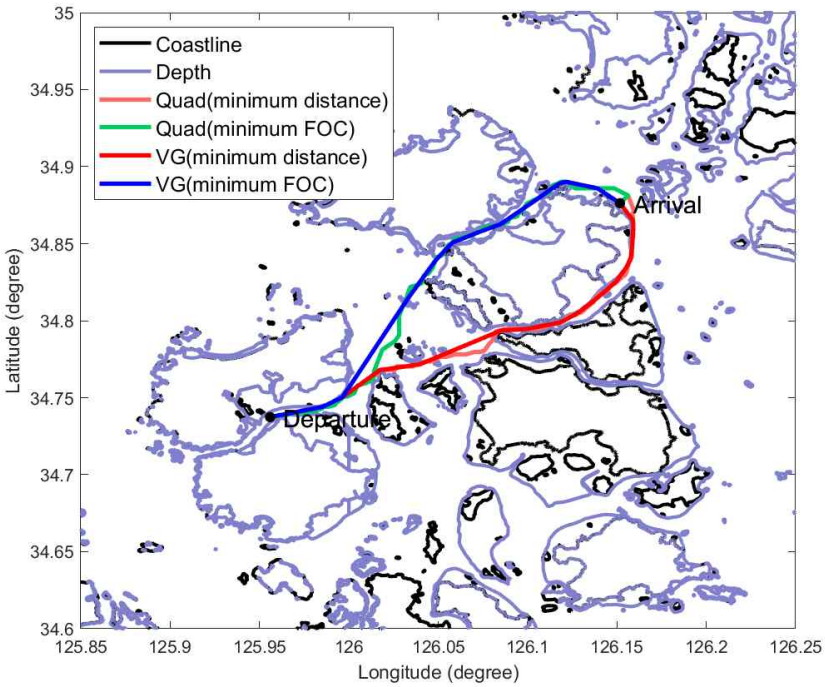

Figure 9-34 Minimum distance and FOC path on visibility graph

Table 9-9 Comparison of minimum distance and FOC quad path

Quad	Distance (km)	ETA (s)	FOC (L)	Number of waypoints	Computation time (s)
Minimum distance	**30.89 (0)**	**6871.39 (0)**	45.61 (+0.22)	311	12.86
Minimum FOC	31.04 (+0.15)	6903.13 (+31.74)	**45.39 (0)**	261	17.05

Table 9-10 Comparison of minimum distance and FOC visibility path

Visibility	Distance (km)	ETA (s)	FOC (L)	Number of waypoints	Computation time (s)
Minimum distance	**26.86 (0)**	**6002.54 (0)**	39.70 (+0.07)	22	3.19
Minimum FOC	27.13 (+0.27)	6062.52 (+59.98)	**39.63 (0)**	21	4.11

최단 거리 항로와 최소 연료 소모량 항로 계산 결과를 Table 9-9와 Table 9-10에 나타냈다. 최단 거리 항로에서는 목적 함수를 이동 거리로 설정했기 때문에 최소 연료 소모량 항로보다 거리와 항해 시간이 짧았으나 연료 소모량은 0.07L 더 많았다. 쿼드 트리 그래프 기반 항로와 가시성 그래프 기반 항로를 비교했을 때, 웨이포인트 개수는 각각 311, 261개에서 22, 21개로 크게 감소한 것을 확인하였다. 계산 시간의 경우 최단 거리 항로는 거리만 계산하면 되지만, 최소 연료 소모량 항로는 연료 소모량을 계산해야 하기 때문에 시간이 오래 걸린 것을 확인하였다. 최단 거리 항로에서 쿼드 트리 기반 항로와 가시성 그래프 기반 항로의 거리와 연료 소모량

을 비교했을 때 각각 13.05%, 11.53% 감소한 것을 확인하였다. 최소 연료 소모량 항로에서 쿼드 트리 기반 항로와 가시성 그래프 기반 항로의 거리와 연료 소모량을 비교했을 때 각각 12.60%, 10.95% 감소한 것을 확인하였다. 시간 - 단위시간 당 연료 소모량 그래프는 Figure 9-35에 나타냈으며, 강조한 부분에서 최단 거리 항로의 시간 당 연료 소모율이 크게 나타난 것을 확인하였다. 최단 거리 항로와 최소 연료 소모량 항로의 속력, 부가 저항, 기상 정보는 Figure 9-36에 나타냈다.

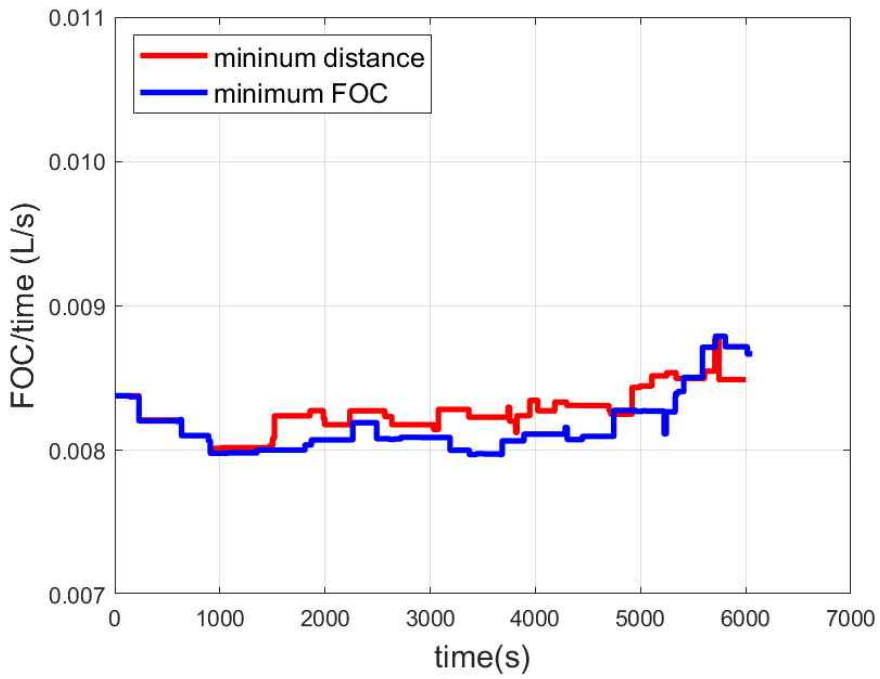

Figure 9-35 Comparison of FOC per time

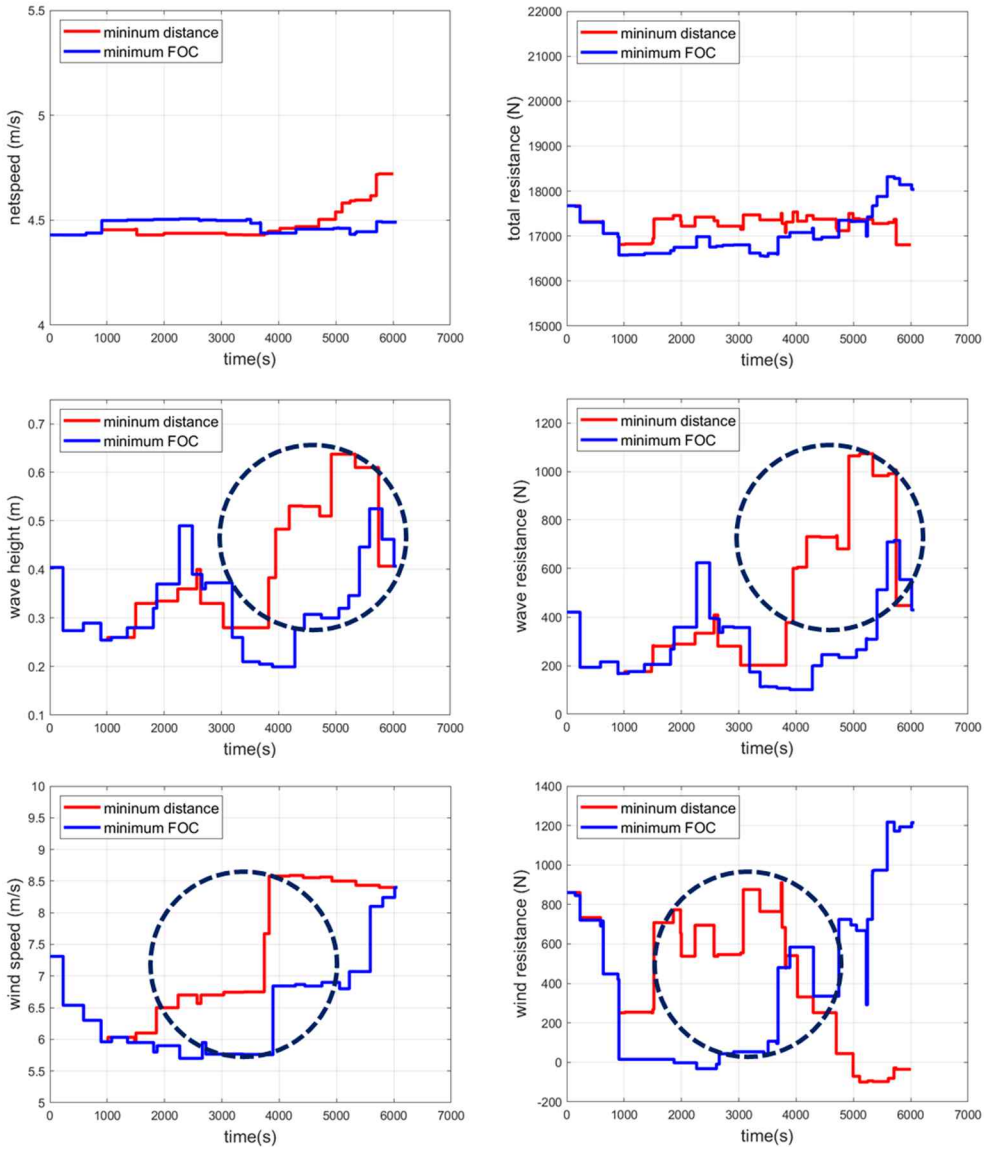

Figure 9-36 Comparison between the results of minimum distance path and minimum FOC path

9.5.2 Second case in west sea of Korea

쿼드 트리를 이용한 그래프는 9.5.1항에서 생성한 그래프를 동일하게 사용하였다. 서해안의 옥도에서 자은도를 가는 항로에 대해서 최단 거리 항로와 최소 연료 소모량 항로를 탐색하였다. 선박이 항해를 시작하는 시간은 한국표준시로 2020년 12월 23일 오전 0시이며, KMA와 KHOA 데이터를 사용하였다. 항해 시간이 짧기 때문에 기상 정보는 선박의 출항 시각 정보를 그대로 사용한다. 앞서 생성한 쿼드 트리 그래프에 출항지와 입항지 위치를 그래프에 삽입한다. 출항지와 입항지에 대해 Dijkstra 알고리즘을 수행하여 최단 거리 항로와 최소 연료 소모량 항로를 계산한다. 그 결과를 Figure 9-37에 나타냈으며, 최단 거리 항로와 최소 연료 소모량 항로가 다르게 나온 것을 확인하였다. 또한, 그림에서 볼 수 있듯이 두 항로 모두 불필요한 점이 많이 생성된 것을 확인할 수 있다. 불필요한 점이 많으면 항로의 거리가 길어지며, 선박의 실제 항로와 많이 다를 수 있으므로 불필요한 점을 제거해야 한다. 불필요한 점을 제거하기 위해 가시성 그래프를 사용하였다. 가시성 그래프의 간선 가중치는 최단 거리 항로는 거리에서는 거리로, 최소 연료 소모량 항로에서는 연료 소모량으로 설정하였다. 가시성 그래프 기반 항로는 Figure 9-38에 나타냈다.

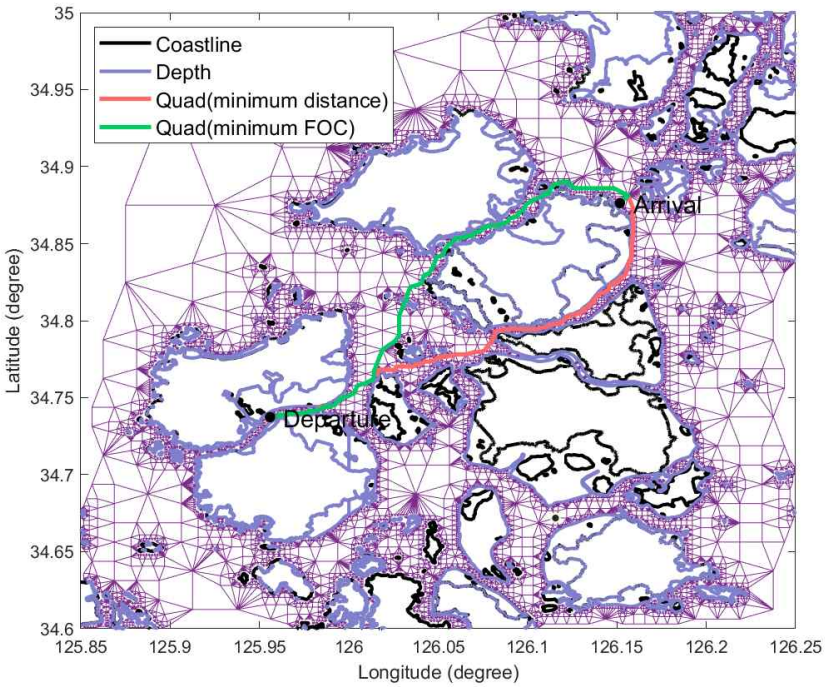

Figure 9-37 Minimum distance and FOC quadtree path

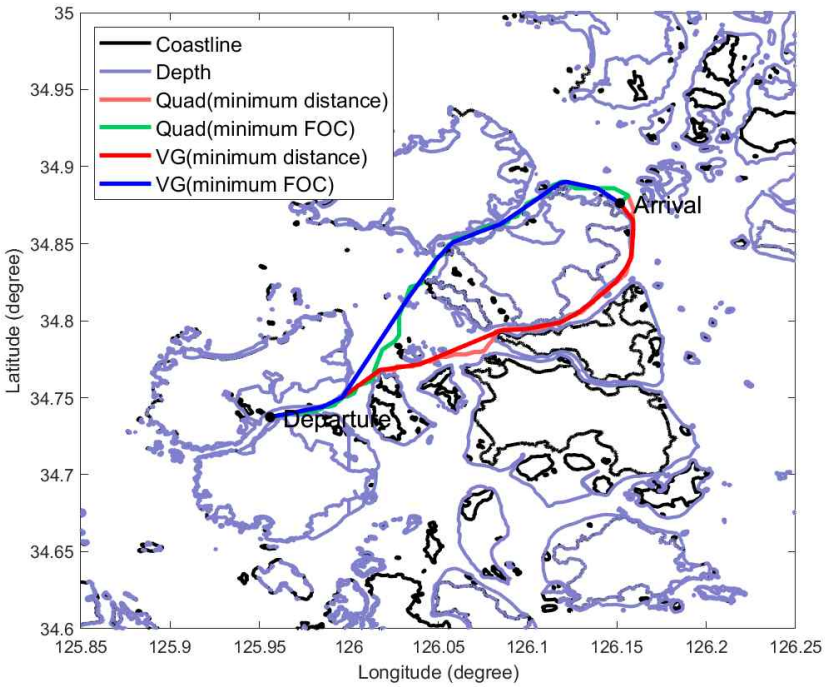

Figure 9-38 Minimum distance and FOC path on visibility graph

Table 9-11 Comparison of minimum distance and FOC quad path

Quad	Distance (km)	ETA (s)	FOC (L)	Number of waypoints	Computation time (s)
Minimum distance	**26.00 (0)**	**5569.54 (0)**	37.52 (+0.10)	150	11.47
Minimum FOC	26.07 (+0.07)	5583.74 (+14.20)	**37.42 (0)**	76	15.87

Table 9-12 Comparison of minimum distance and FOC visibility path

Visibility	Distance (km)	ETA (s)	FOC (L)	Number of waypoints	Computation time (s)
Minimum distance	**23.68 (0)**	**5061.03 (0)**	34.14 (+0.26)	10	1.45
Minimum FOC	23.69 (+0.01)	5062.33 (+1.30)	**33.88 (0)**	7	0.57

최단 거리 항로와 최소 연료 소모량 항로 계산 결과를 Table 9-11과 Table 9-12에 나타냈다. 최단 거리 항로에서는 목적 함수를 이동 거리로 설정했기 때문에 최소 연료 소모량 항로보다 거리와 항해 시간이 짧았으나 연료 소모량은 0.26L 더 많았다. 쿼드 트리 그래프 기반 항로와 가시성 그래프 기반 항로를 비교했을 때, 웨이포인트 개수는 각각 150, 70개에서 10, 7개로 크게 감소한 것을 확인하였다. 계산 시간의 경우 최단 거리 항로는 거리만 계산하면 되지만, 최소 연료 소모량 항로는 연료 소모량을 계산해야 하기 때문에 시간이 오래 걸린 것을 확인하였다. 최단 거리 항로에서 쿼드 트리 기반 항로와 가시성 그래프 기반 항로의 거리와 연료 소모량을 비교

했을 때 각각 8.99%, 8.92% 감소한 것을 확인하였다. 최소 연료 소모량 항로에서 쿼드 트리 기반 항로와 가시성 그래프 기반 항로의 거리와 연료 소모량을 비교했을 때 각각 9.46%, 9.16% 감소한 것을 확인하였다. 시간 - 단위시간 당 연료 소모량 그래프는 Figure 9-39에 나타냈으며, 강조한 부분에서 최단 거리 항로의 시간 당 연료 소모율이 크게 나타난 것을 확인하였다. 최단 거리 항로와 최소 연료 소모량 항로의 속력, 부가 저항, 기상 정보는 Figure 9-40에 나타냈다.

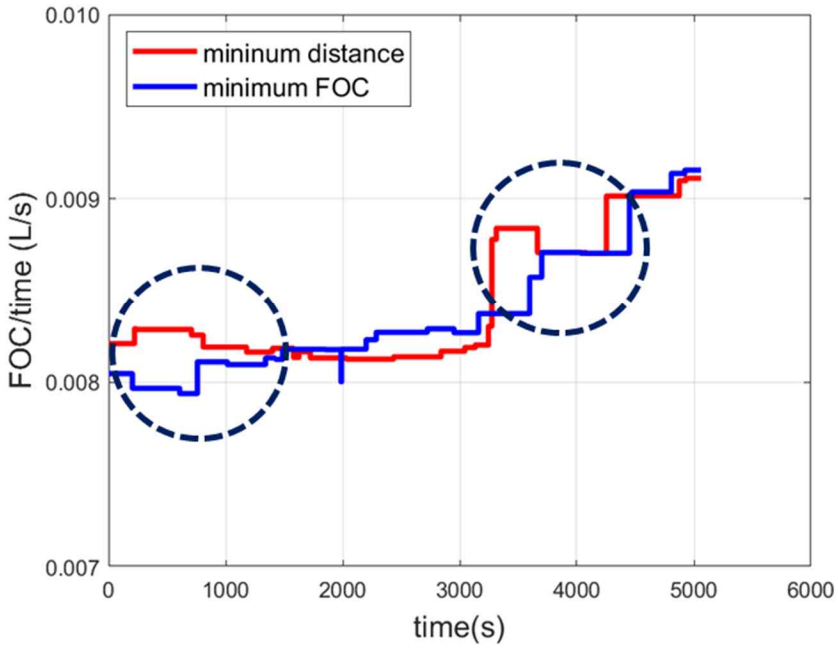

Figure 9-39 Comparison of FOC per time

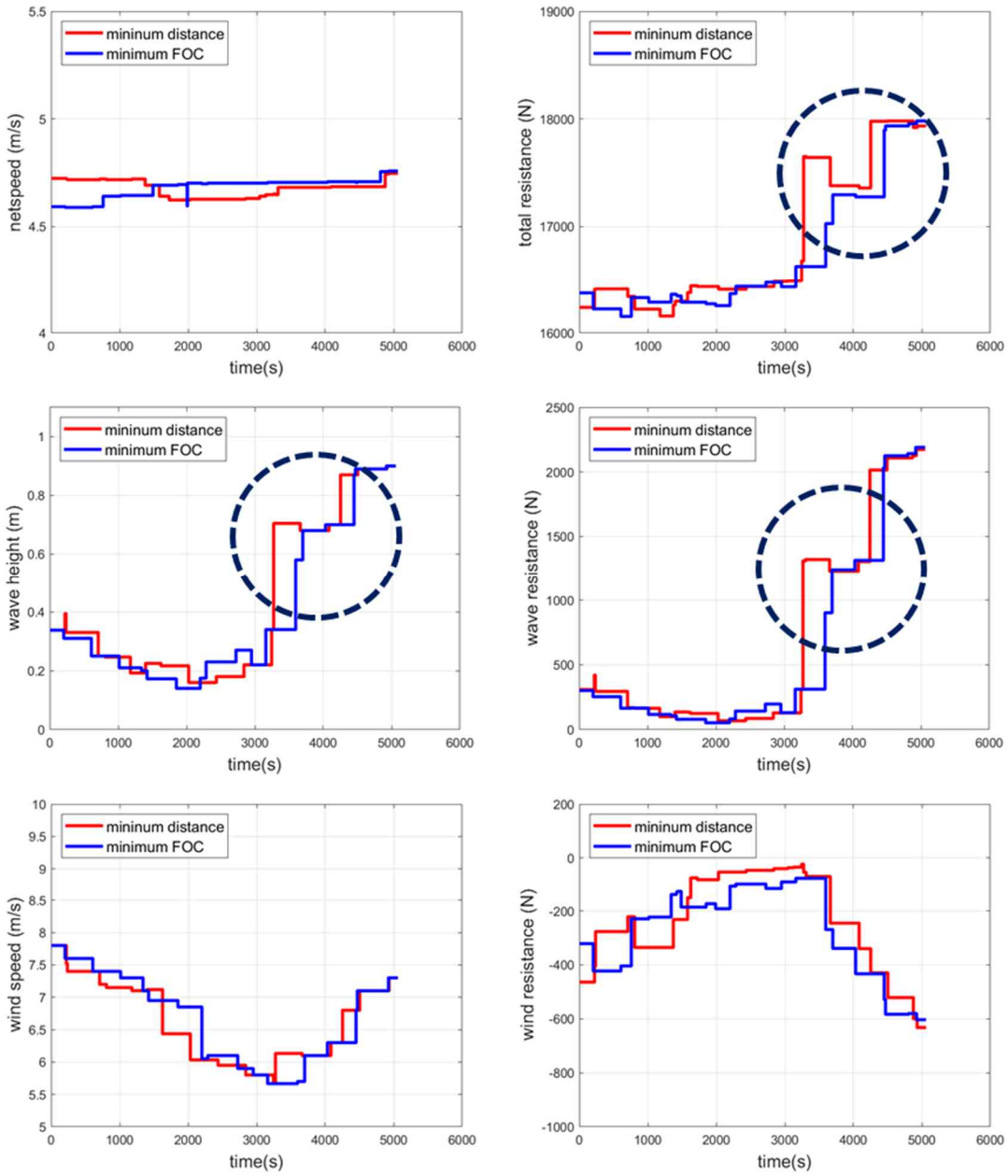

Figure 9-40 Comparison between the results of minimum distance path and minimum FOC path

9.5.3 Third case in south sea of Korea

그래프를 생성하기 위해 남해안의 영역을 경도 (127.7° E, 128.7° E), 위도 (34.5° N, 35.5° N)을 선택했다. 먼저, 남해안의 영역에서 육지 정보를 가져온 결과는 Figure 9-41에 나타냈다. 조위 정보를 반영하여 수심 정보를 가져온 결과는 Figure 9-42에 나타냈다. Polygon offset을 이용하여 장애물들을 100m만큼 확장시킨 후 PMR 쿼드 트리를 생성한 결과를 Figure 9-43에 나타냈다. 쿼드 트리 기반으로 4방향으로 인접한 격자에 대해 간선을 연결한 그래프 결과는 Figure 9-44에 나타냈다. 이 때 장애물과 겹치거나 장애물 내부에 있는 그래프의 간선은 모두 제거하였다. 남해안은 서해안에 비해 조수간만의 차가 크지 않으므로 수심으로 인해 항로가 크게 변하는 경우는 거의 없었다. 육지와 가까운 일부 영역에만 선박의 흘수보다 낮은 수심이 있어 해당 부분만 지나갈 수 없도록 설정하였다.

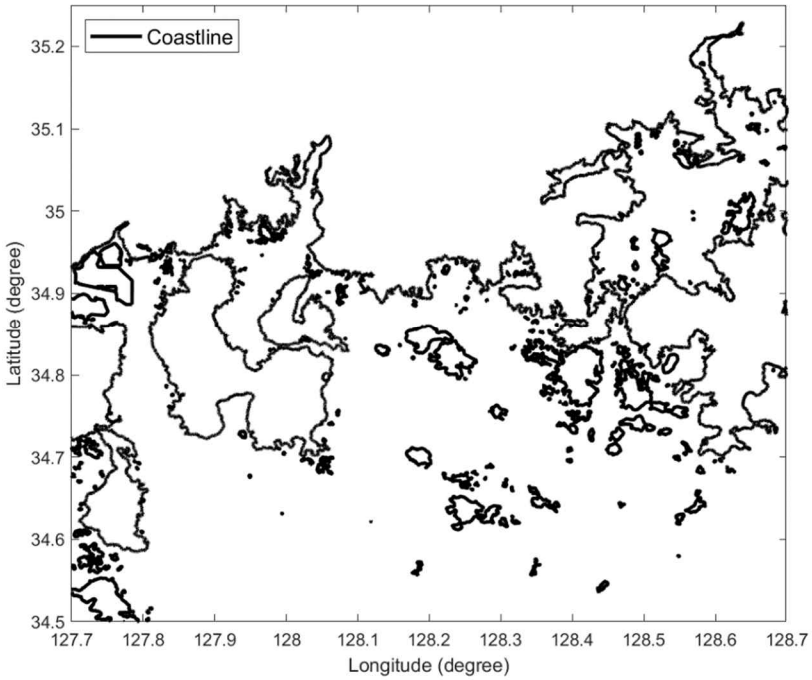

Figure 9-41 Land areas in south sea

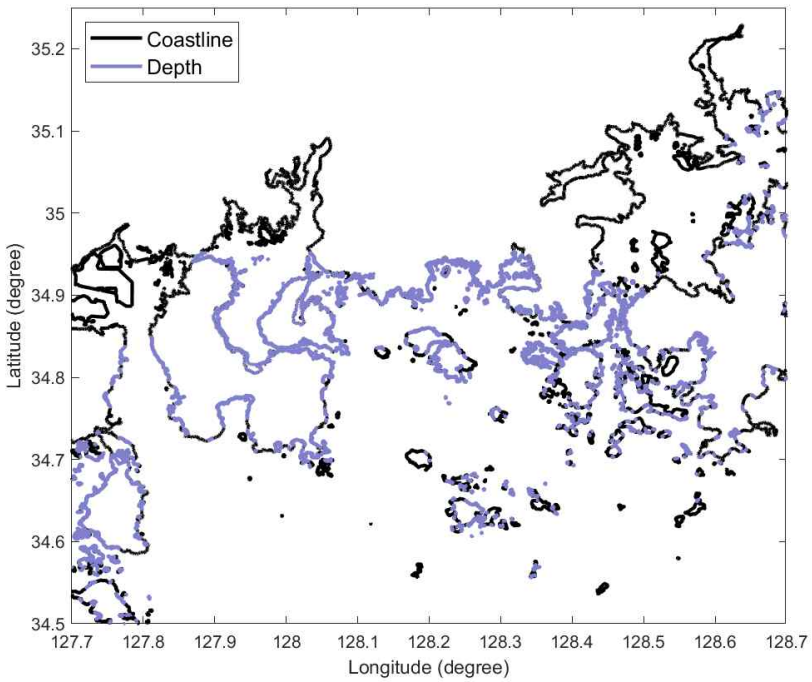

Figure 9-42 Land and depth areas in south sea

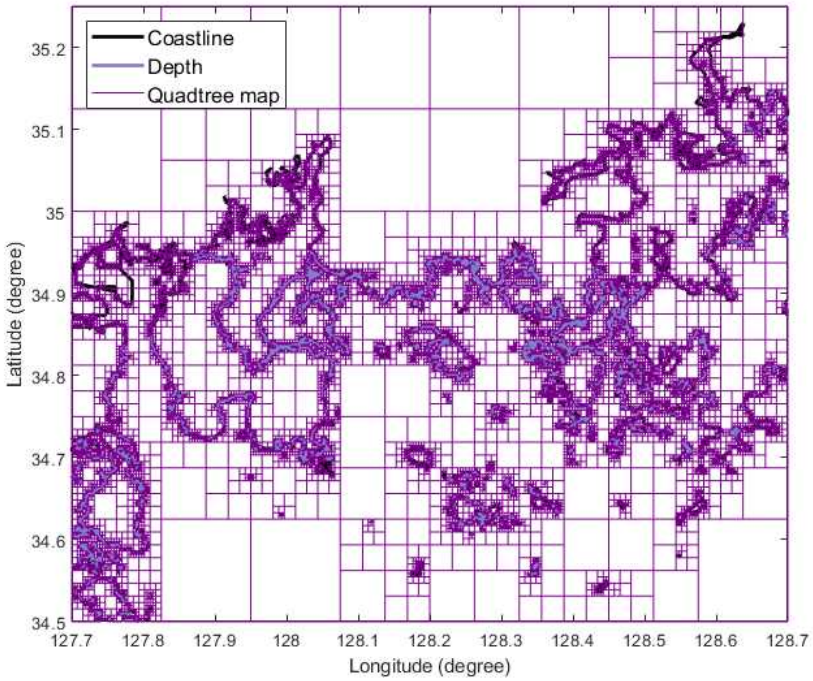

Figure 9-43 PMR quadtree generation results in south sea

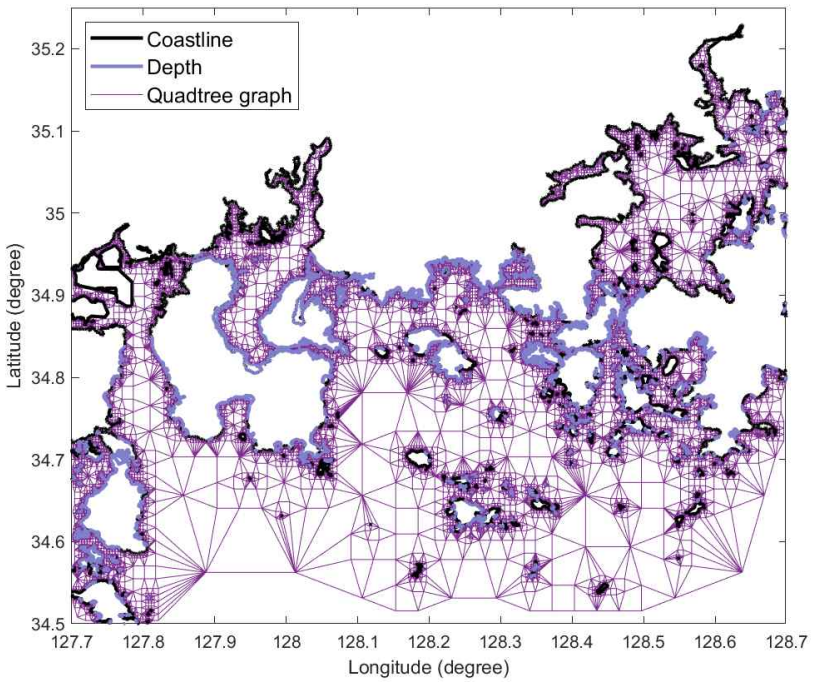

Figure 9-44 Quadtree based graph in south sea

서해안의 도초에서 발무기항을 가는 항로에 대해서 최단 거리 항로와 최소 연료 소모량 항로를 탐색하였다. 선박이 항해를 시작하는 시간은 한국표준시로 2020년 12월 23일 오전 0시이며, KMA와 KHOA 데이터를 사용하였다. 항해 시간이 짧기 때문에 기상 정보는 선박의 출항 시각 정보를 그대로 사용한다. 앞서 생성한 쿼드 트리 그래프에 출항지와 입항지 위치를 그래프에 삽입한다. 출항지와 입항지에 대해 Dijkstra 알고리즘을 수행하여 최단 거리 항로와 최소 연료 소모량 항로를 계산한다. 그 결과를 Figure 9-45에 나타냈으며, 최단 거리 항로와 최소 연료 소모량 항로가 다르게 나온 것을 확인하였다. 또한, 그림에서 볼 수 있듯이 두 항로 모두 불필요한 점이 많이 생성된 것을 확인할 수 있다. 불필요한 점이 많으면 항로의 거리가 길어지며, 선박의 실제 항로와 많이 다를 수 있으므로 불필요한 점을 제거해야 한다. 불필요한 점을 제거하기 위해 가시성 그래프를 사용하였다. 가시성 그래프의 간선 가중치는 최단 거리 항로는 거리에서는 거리로, 최소 연료 소모량 항로에서는 연료 소모량으로 설정하였다. 각각 가시성 그래프를 적용한 결과를 Figure 9-46에 나타냈다.

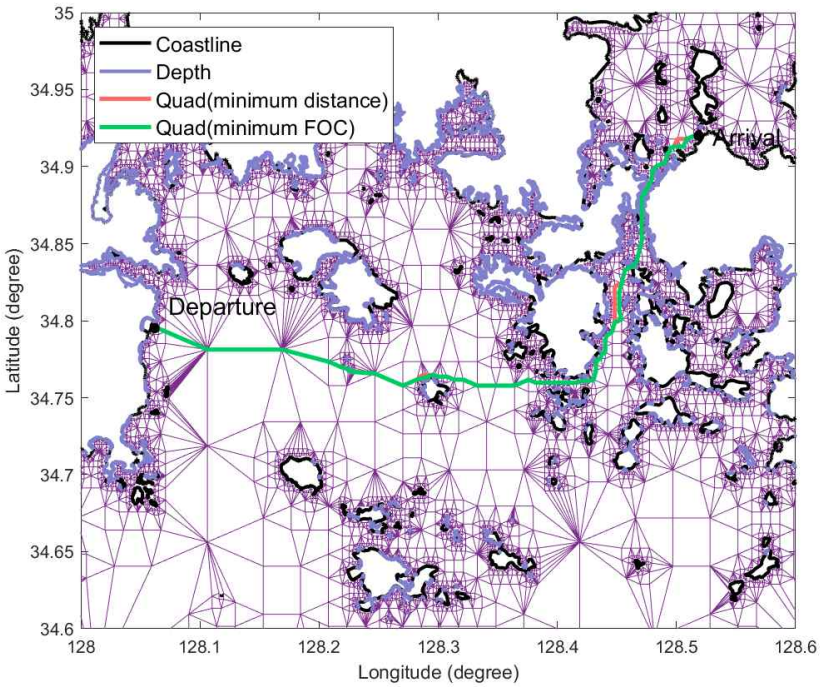

Figure 9-45 Minimum distance and FOC quadtree path

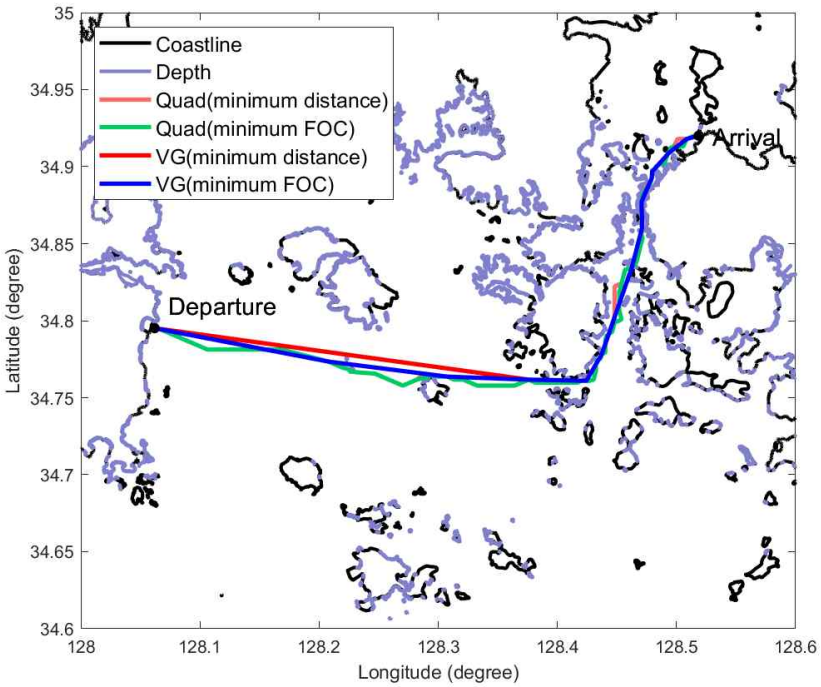

Figure 9-46 Minimum distance and FOC path on visibility graph

Table 9-13 Comparison of minimum distance and FOC quad path

Quad	Distance (km)	ETA (s)	FOC (L)	Number of waypoints	Computation time (s)
Minimum distance	**57.34 (0)**	**12,284.35 (0)**	89.95 (+0.15)	157	10.47
Minimum FOC	57.40 (+0.06)	12,296.40 (+12.05)	**89.80 (0)**	163	12.78

Table 9-14 Comparison of minimum distance and FOC visibility path

Visibility	Distance (km)	ETA (s)	FOC (L)	Number of waypoints	Computation time (s)
Minimum distance	**53.97 (0)**	**11,542.74 (0)**	85.37 (+0.19)	19	1.24
Minimum FOC	54.00 (+0.03)	11,550.63 (+7.89)	**85.18 (0)**	20	1.37

최단 거리 항로와 최소 연료 소모량 항로 계산 결과를 Table 9-13과 Table 9-14에 나타냈다. 최단 거리 항로에서는 목적 함수를 이동 거리로 설정했기 때문에 최소 연료 소모량 항로보다 거리와 항해 시간이 짧았으나 연료 소모량은 0.19L 더 많았다. 쿼드 트리 그래프 기반 항로와 가시성 그래프 기반 항로를 비교했을 때, 웨이포인트 개수는 각각 157, 163개에서 19, 20개로 크게 감소한 것을 확인하였다. 계산 시간의 경우 최단 거리 항로는 거리만 계산하면 되지만, 최소 연료 소모량 항로는 연료 소모량을 계산해야 하기 때문에 시간이 오래 걸린 것을 확인하였다. 최단 거리 항로에서 쿼드 트리 기반 항로와 가시성 그래프 기반 항로의 거리와 연료 소모량

을 비교했을 때 각각 5.88%, 5.92% 감소한 것을 확인하였다. 최소 연료 소모량 항로에서 쿼드 트리 기반 항로와 가시성 그래프 기반 항로의 거리와 연료 소모량을 비교했을 때 각각 5.09%, 5.14% 감소한 것을 확인하였다. 시간 - 단위시간 당 연료 소모량 그래프는 Figure 9-47에 나타냈으며, 강조한 부분에서 최단 거리 항로의 시간 당 연료 소모율이 크게 나타난 것을 확인하였다. 최단 거리 항로와 최소 연료 소모량 항로의 속력, 부가 저항, 기상 정보는 Figure 9-48에 나타냈다.

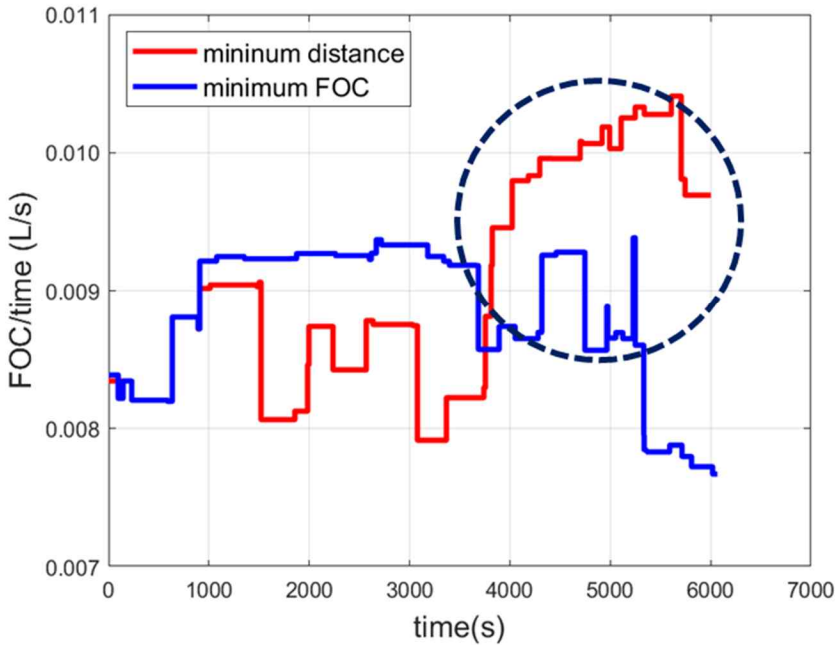

Figure 9-47 Comparison of FOC per time

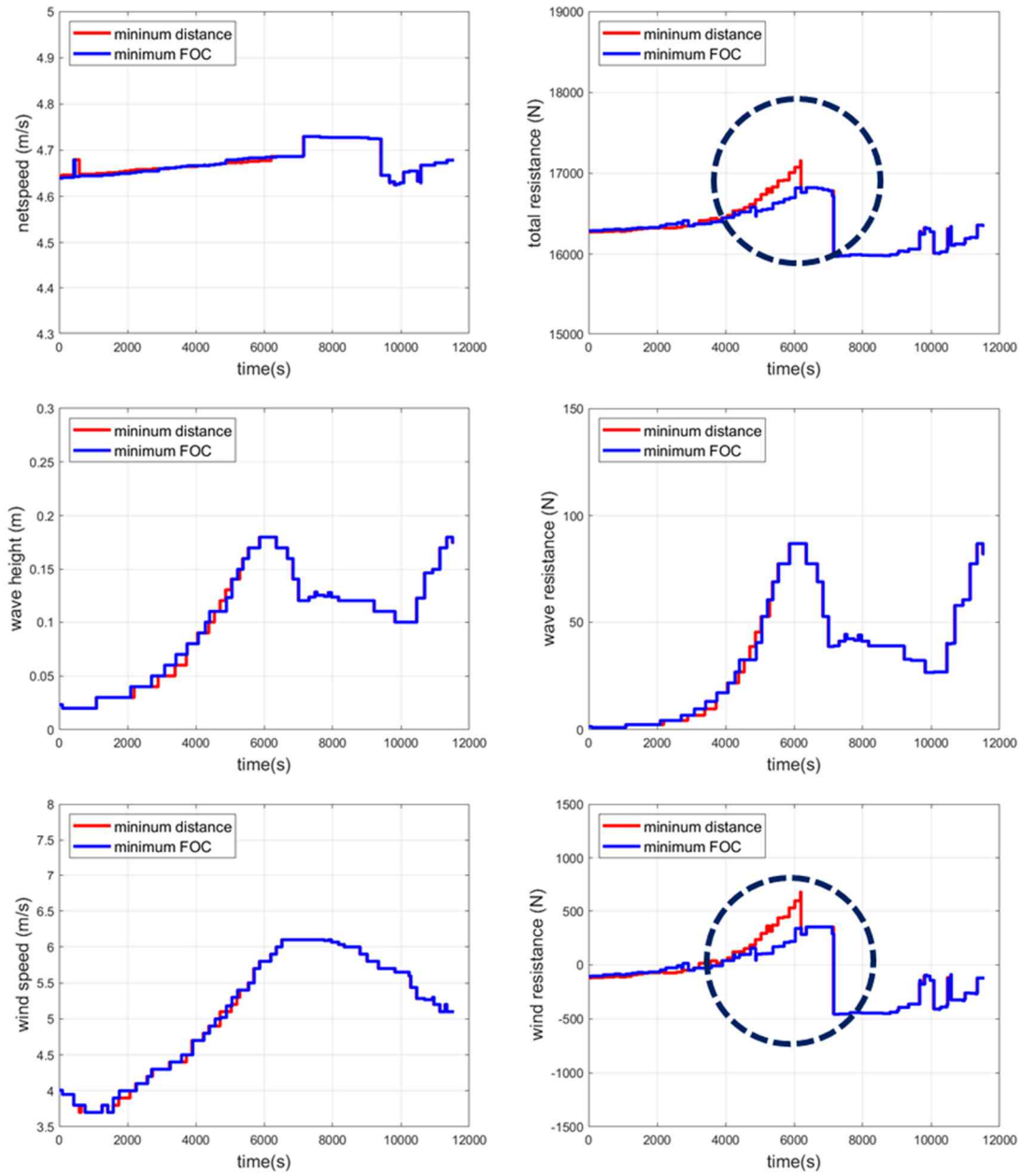

Figure 9-48 Comparison between the results of minimum distance path and minimum FOC path

9.5.4 Second case in south sea of Korea

쿼드 트리를 이용한 그래프는 9.5.3항에서 생성한 그래프를 동일하게 사용하였다. 노량항에서 거제도 대포항을 가는 항로에 대해서 최단 거리 항로와 최소 연료 소모량 항로를 탐색하였다. 선박이 항해를 시작하는 시간은 한국표준시로 2020년 12월 23일 오전 0시이며, KMA와 KHOA 데이터를 사용하였다. 항해 시간이 짧기 때문에 기상 정보는 선박의 출항 시각 정보를 그대로 사용한다. 앞서 생성한 쿼드 트리 그래프에 출항지와 입항지 위치를 그래프에 삽입한다. 출항지와 입항지에 대해 Dijkstra 알고리즘을 수행하여 최단 거리 항로와 최소 연료 소모량 항로를 계산한다. 그 결과를 Figure 9-49에 나타냈으며, 최단 거리 항로와 최소 연료 소모량 항로가 다르게 나온 것을 확인하였다. 또한, 그림에서 볼 수 있듯이 두 항로 모두 불필요한 점이 많이 생성된 것을 확인할 수 있다. 불필요한 점이 많으면 항로의 거리가 길어지며, 선박의 실제 항로와 많이 다를 수 있으므로 불필요한 점을 제거해야 한다. 불필요한 점을 제거하기 위해 가시성 그래프를 사용하였다. 가시성 그래프의 간선 가중치는 최단 거리 항로는 거리에서는 거리로, 최소 연료 소모량 항로에서는 연료 소모량으로 설정하였다. 가시성 그래프 기반 항로는 Figure 9-50에 나타냈다.

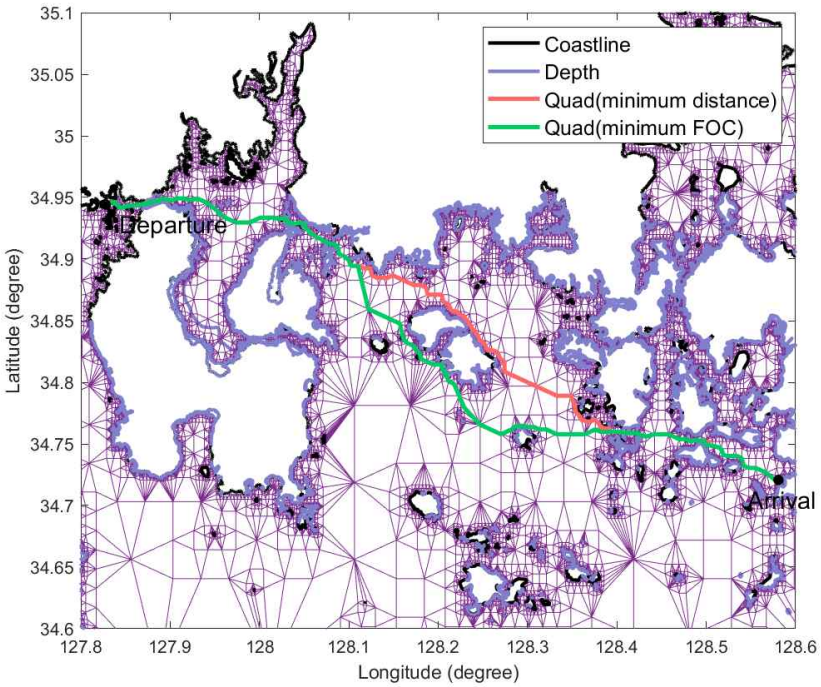

Figure 9-49 Minimum distance and FOC quadtree path

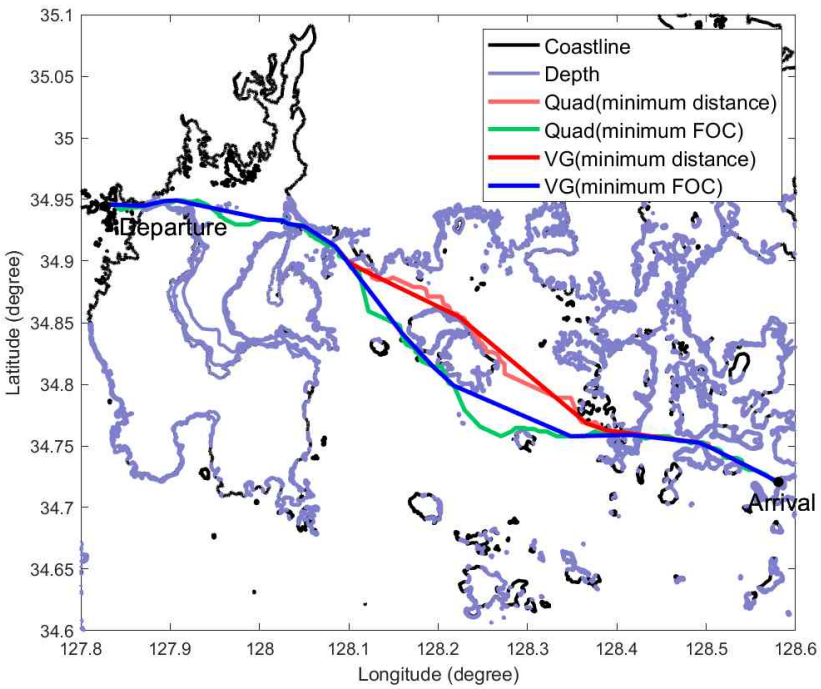

Figure 9-50 Minimum distance and FOC path on visibility graph

Table 9-15 Comparison of minimum distance and FOC quad path

Quad	Distance (km)	ETA (s)	FOC (L)	Number of waypoints	Computation time (s)
Minimum distance	**82.68 (0)**	**19885.75 (0)**	76.01 (+0.32)	256	11.54
Minimum FOC	83.64 (+1.18)	20,156.92 (+79.07)	**75.79 (0)**	243	16.08

Table 9-16 Comparison of minimum distance and FOC visibility path

Visibility	Distance (km)	ETA (s)	FOC (L)	Number of waypoints	Computation time (s)
Minimum distance	**75.24 (0)**	**18,133.37 (0)**	69.41 (+0.02)	20	1.92
Minimum FOC	76.55 (+0.92)	18,430.40 (+179.08)	**69.39 (0)**	24	1.91

최단 거리 항로와 최소 연료 소모량 항로 계산 결과를 Table 9-15와 Table 9-16에 나타냈다. 최단 거리 항로에서는 목적 함수를 이동 거리로 설정했기 때문에 최소 연료 소모량 항로보다 거리와 항해 시간이 짧았으나 연료 소모량은 0.02L 더 많았다. 쿼드 트리 그래프 기반 항로와 가시성 그래프 기반 항로를 비교했을 때, 웨이포인트 개수는 각각 256, 243개에서 20, 24개로 크게 감소한 것을 확인하였다. 계산 시간의 경우 최단 거리 항로는 거리만 계산하면 되지만, 최소 연료 소모량 항로는 연료 소모량을 계산해야 하기 때문에 시간이 오래 걸린 것을 확인하였다. 최단 거리 항로에서 쿼드 트리 기반 항로와 가시성 그래프 기반 항로의 거리와 연료 소모량

을 비교했을 때 각각 9.15%, 8.55% 감소한 것을 확인하였다. 최소 연료 소모량 항로에서 쿼드 트리 기반 항로와 가시성 그래프 기반 항로의 거리와 연료 소모량을 비교했을 때 각각 8.56%, 8.38% 감소한 것을 확인하였다. 시간 - 단위시간 당 연료 소모량 그래프는 Figure 9-51에 나타냈으며, 강조한 부분에서 최단 거리 항로의 시간 당 연료 소모율이 크게 나타난 것을 확인하였다. 최단 거리 항로와 최소 연료 소모량 항로의 속력, 부가 저항, 기상 정보는 Figure 9-52에 나타냈다.

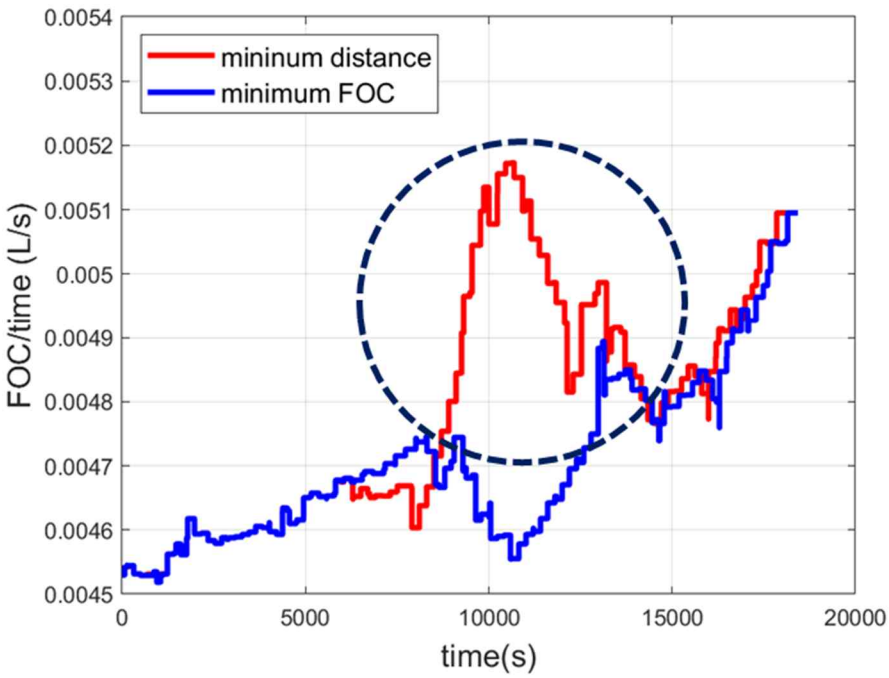

Figure 9-51 Comparison of FOC per time

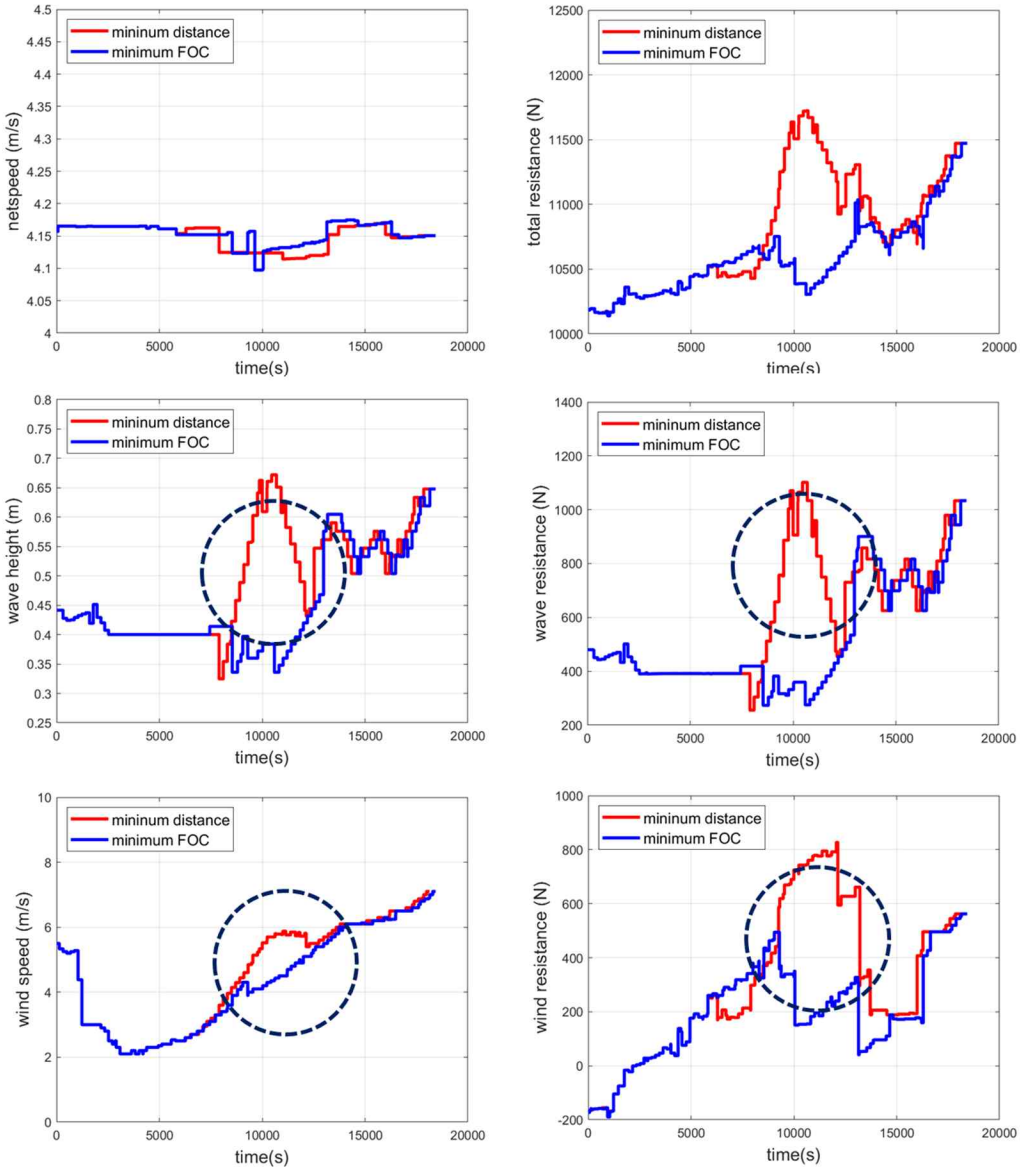

Figure 9-52 Comparison between the results of minimum distance path and minimum FOC path

9.6 Long voyage path results

본 절에서는 서해안의 인천에서 남해안의 부산까지 가는 최적 항로를 계산하고 분석한다. 장거리를 이동하기 때문에 연안 선박 중 크기가 큰 선박을 선택하였으며, 해당 선박의 제원은 과 같다. 선박이 항해를 시작하는 시간은 한국표준시로 2020년 12월 23일 12시이며, 한국 기상청과 한국 해양조사원 데이터를 사용하였다. 쿼드 트리 기반 그래프에 출발점과 도착점을 삽입하여 Dijkstra 알고리즘을 수행하였으며, 이후 가시성 그래프를 이용하여 불필요한 점을 제거하였다.

Table 9-17 Ship specification for long voyage simulation

Item	Value
Length between perpendiculars (LPP)	43 m
Breadth (B)	13 m
Draft (design)	2.3 m
Wetted surface (S)	416.3 m^2
Displacement volume	219.9 m^3
Block coefficient	0.55
Frictional resistance coefficient	0.004648
Transverse projected area above waterline	112.5 m^2
Quasi-propulsive efficiency	0.703
Transmission efficiency	0.98
Fuel consumption ratio	0.156 L/(ps·h)

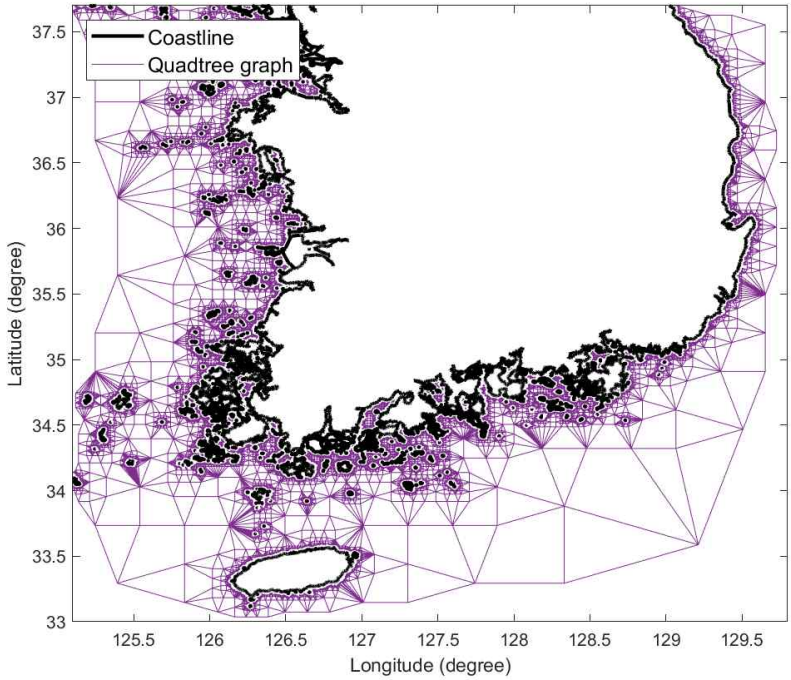

Figure 9-53 Quadtree graph for long voyage

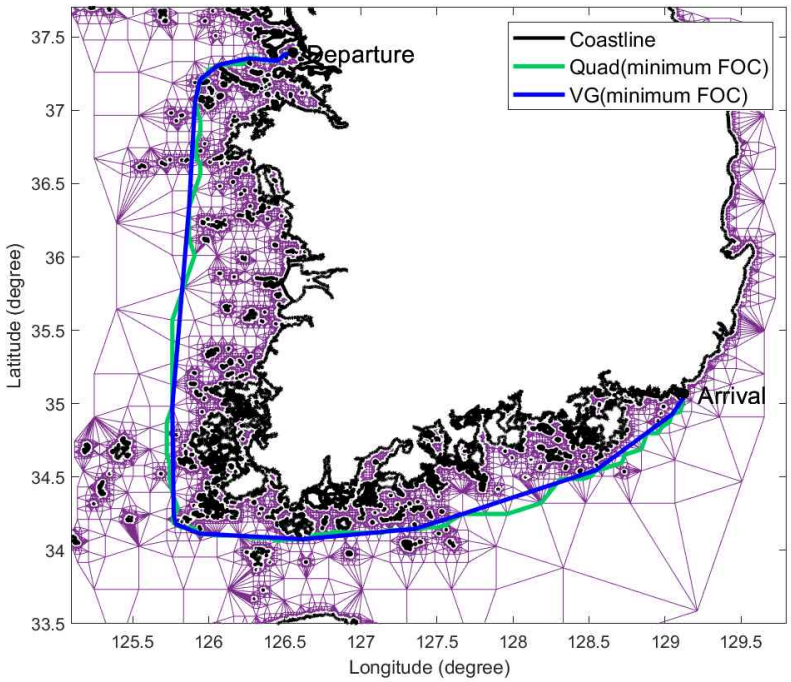

Figure 9-54 Quadtree and VG minimum FOC path in long voyage

Table 9-18 Comparison of minimum FOC quad path and visibility path

	Distance (km)	ETA (s)	FOC (L)	Number of waypoints	Computation time (s)
Minimum FOC quad	786.52	17.67	10,757	338	30.17
Minimum FOC VG	744.01	16.71	10,180	26	9.11

최소 연료 소모량 항로 계산 결과를 Table 9-18에 나타냈다. 현재 가지고 있는 기상 정보로는 연료 소모량을 최소로 한 항로와 최단 거리 항로의 차이가 유의미하지 않아 연료 소모량을 최소로 한 항로만 나타냈다. 쿼드트리 그래프 기반 항로와 가시성 그래프 기반 항로를 비교했을 때, 웨이포인트 개수는 각각 338개에서 26개로 크게 감소한 것을 확인하였다. 계산 시간의 경우 앞의 절에서 단거리 항로를 계산할 때는 약 20초가 걸렸으나 장거리 항로를 계산할 때는 39.28초가 걸렸다. 계산 시간 증가 이유는 항로의 거리가 길어지면서 장애물 증가, 기상 정보 계산, 연료 소모량 계산 등이 있다. 최소 연료 소모량 항로에서 쿼드 트리 기반 항로와 가시성 그래프 기반 항로의 거리와 연료 소모량을 비교했을 때 각각 5.40%, 5.36% 감소한 것을 확인하였다. 시간 - 단위시간 당 연료 소모량 그래프는 Figure 9-55에 나타냈으며, 강조한 부분에서 최단 거리 항로의 시간 당 연료 소모율이 크게 나타난 것을 확인하였다. 최단 거리 항로와 최소 연료 소모량 항로의 속력, 부가 저항, 기상 정보는 Figure 9-56에 나타냈다.

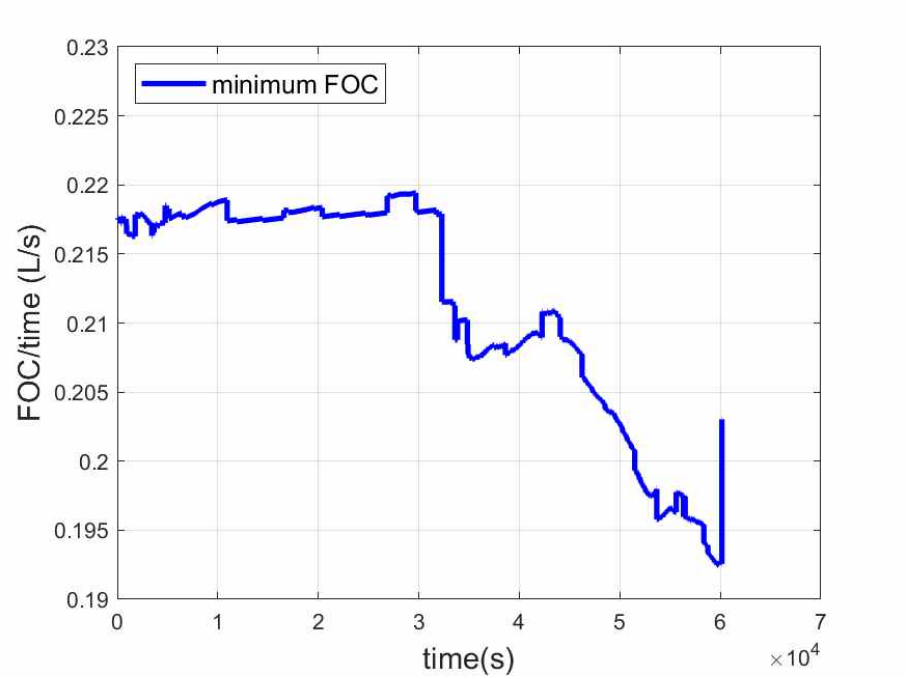

Figure 9-55 FOC per time in long voyage

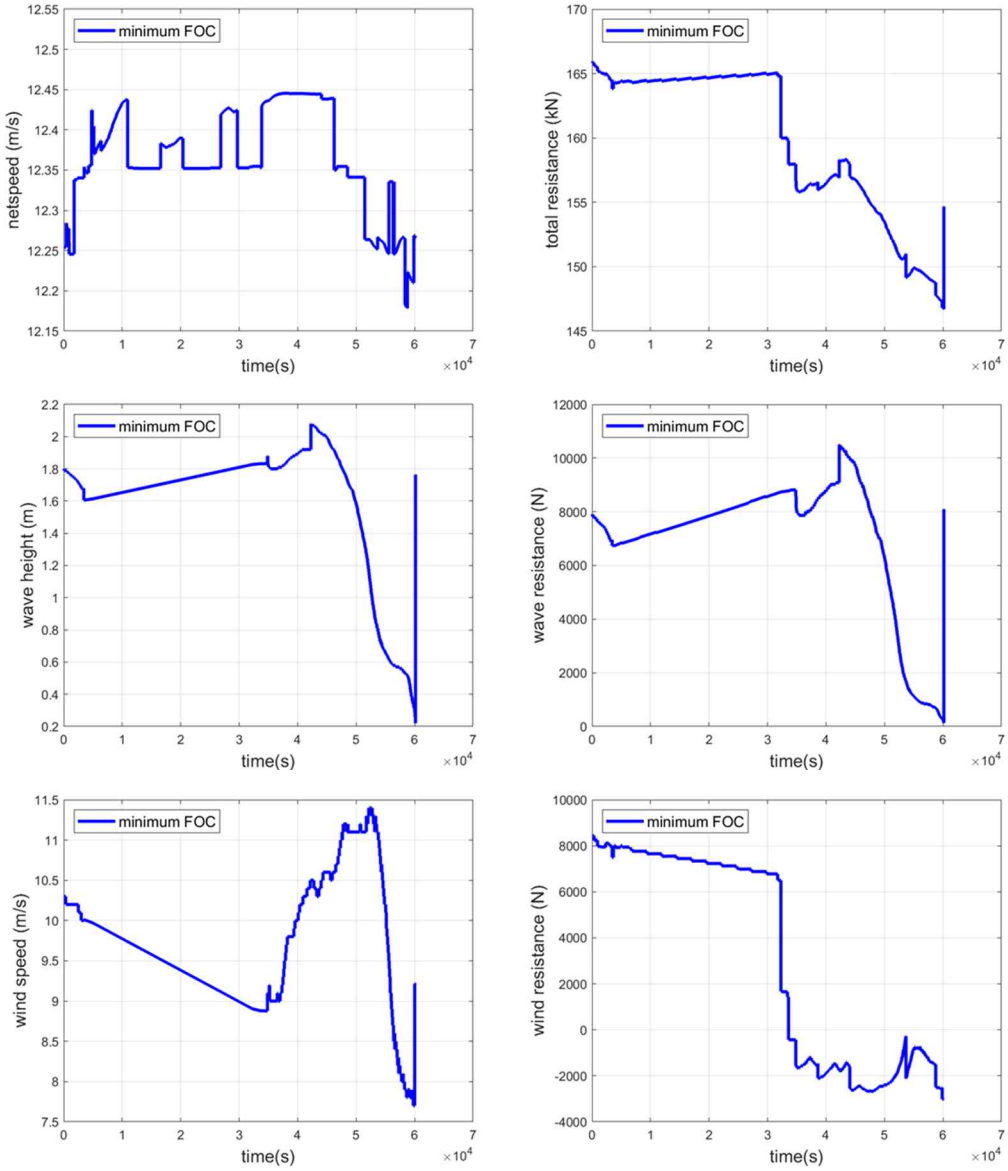

Figure 9-56 The results of minimum FOC path in long voyage

10 Conclusion and future works

본 연구에서는 해안선 확장 알고리즘, 쿼드 트리, 가시성 그래프 및 Dijkstra 알고리즘을 통합하여 복잡한 해양 환경에 대한 선박의 항로 계획 문제를 해결했다. 오프셋을 사용한 해안선 확장 알고리즘은 선박을 지형으로부터 일정 거리를 유지시키기 위해 적용되었으며, PMR 쿼드 트리는 복잡한 해양 환경을 표현하는 데 필요한 계산 시간과 메모리 소비를 줄이는데 사용된다. 이 접근법에서 쿼드 트리 구조는 쿼드 트리 기반 그래프로 변환되고 Dijkstra 알고리즘은 최단 경로를 찾는 데 사용된다. 쿼드 트리 최단 경로에서 최적의 경로를 찾기 위해 획득 한 경로 웨이 포인트를 기반으로 가시성 그래프가 구성되며, 그 후 가시성 그래프에서 Dijkstra 알고리즘을 이용하여 최단 경로를 탐색한다. 가시성 그래프 구성 시 사용되는 가시성 알고리즘은 쿼드 트리 구조를 사용하여 개선되었다. 한국의 남해안과 서해안에 대한 쿼드 트리 기반 그래프와 가시성 그래프의 경로 길이를 비교해 보면 가시성 그래프가 쿼드 트리 기반 그래프에서 얻은 경로의 품질을 향상시키는 것을 확인할 수 있었다. 한국의 5 개 사례를 비교하여 쿼드 트리 기반 그래프에 비해 가시성 그래프에서 향상된 경로 품질을 확인했다. 또한 쿼드 트리와 가시성 그래프 접근 방식의 통합으로 일반 가시성 그래프를 사용할 때보다 약 6 배 빠른 것을 확인했다. 이 연구에서 제안된 알고리즘은 복잡한 해양 환경에 대해 메모리 사용량을 줄이고 계산 시간을 크게 감소시켰다.

그리고 기상정보를 적용하여 연료 소모량이 최소가 되는 항로를 계산하였다. 사용한 기상정보는 바람, 파랑, 조류, 조위 정보이다. 조위 정보는 수심에 반영하여 선박이 지나갈 수 없는 영역을 판단하기 위해 사용하였다. 선박의 부가 저항을 계산하기 위해 파랑과 바람 정보를 사용하였으며, 선박의 대수 속력을 계산하기 위해 조류를 반영하였다. 이러한 정보를 이용하여 유효 동력을 계산하였으며, 이를 기반으로 연료 소모량을 계산하였다. 연료 소모량을 기반으로 연안 항로를 탐색하였으며, 한국의 서해안과 남해안에 적용하였다. 시뮬레이션 결과에서 최소 연료 소모량 항로는 최단 거리 항로에 비해 연료 소모량이 최대 약 1.79% 감소한 것을 확인하였다. 또한 최단 거리 항로와 형상이 다른 최소 연료 소모량 항로가 계산된 것을 확인하였다. 인천과 부산에 대한 장거리 항해에 대한 시뮬레이션도 수행하여, 장거리 항해에도 본 연구가 적용이 가능하다는 것을 확인하였다.

본 논문의 연구는 복잡한 해양 환경에서 기상 정보를 이용하여 선박의 연료 소모량이 최소가 되는 항로를 계산하였으나 실제 연안 선박의 항로에 적용하여 검증하지는 못하였다. 향후 제안한 알고리즘을 이용하여 계산한 선박의 항로 계획과 실제 항해 결과를 비교하여 제안한 알고리즘의 검증을 수행할 계획이다.

References

Bhattacharya, P., Gavrilova, M.L., 2008. Roadmap-based path planning - Using the voronoi diagram for a clearance-based shortest path. IEEE Robot. Autom. Mag. 15, 58–66. https://doi.org/10.1109/MRA.2008.921540

Blum, C., 2005. Ant colony optimization: Introduction and recent trends. Phys. Life Rev. https://doi.org/10.1016/j.plrev.2005.10.001

Chen, X., McMains, S., 2005. Polygon offsetting by computing winding numbers, in: Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference - DETC2005. pp. 565–575. https://doi.org/10.1115/detc2005-85513

Chopard, B., Tomassini, M., 2018. Particle swarm optimization, in: Natural Computing Series. Springer Verlag, pp. 97–102. https://doi.org/10.1007/978-3-319-93073-2_6

Daniel, K., Nash, A., Koenig, S., Felner, A., 2010. Theta*: Any-angle path planning on grids. J. Artif. Intell. Res. 39, 533–579. https://doi.org/10.1613/jair.2994

Dijkstra, E.W., 1959. A Note on Two Problems in Connexion with Graphs. Numer. Math. 1, 269–271.

Ferguson, D., Stentz, A., 2006. Using interpolation to improve path planning: The Field D* algorithm. J. F. Robot. 23, 79–101. https://doi.org/10.1002/rob.20109

Ghosh, S.K., Mount, D.M., 1991. Output-sensitive algorithm for computing visibility graphs. SIAM J. Comput. 20, 888–910. https://doi.org/10.1137/0220055

Hart, P.E., Nilsson, N.J., Raphael, B., 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Trans. Syst. Sci. Cybern. 4, 100–107. https://doi.org/10.1109/TSSC.1968.300136

Huang, G., Feng, B., Wang, Z., Wang, D., 2014. Quadtree routing for hybrid satellite/terrestrial networks, in: 2014 6th International Conference on Wireless Communications and Signal Processing, WCSP 2014. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/WCSP.2014.6992204

ISO, 2015. Ships and marine technology – Guidelines for the assessment of speed and power performance by analysis of speed trial data.

Jaillet, L., Cortés, J., Siméon, T., 2010. Sampling-based path planning on configuration-space costmaps. IEEE Trans. Robot. 26, 635–646. https://doi.org/10.1109/TRO.2010.2049527

Kallmann, M., 2010. Navigation queries from triangular meshes, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer, Berlin, Heidelberg, pp. 230–241. https://doi.org/10.1007/978-3-642-16958-8_22

Kapoor, S., Maheshwari, S.N., 2000. Efficiently constructing the visibility graph of a simple polygon with obstacles. SIAM J. Comput. 30, 847–871. https://doi.org/10.1137/S0097539795253591

Karaman, S., Frazzoli, E., 2011a. Sampling-based Algorithms for Optimal Motion Planning. Int. J. Rob. Res. 30, 1–76.

Karaman, S., Frazzoli, E., 2011b. Sampling-based algorithms for optimal motion planning. Int. J. Rob. Res. 30, 846–894. https://doi.org/10.1177/0278364911406761

Kim, B., Kim, T.W., 2017. Weather routing for offshore transportation using genetic algorithm. Appl. Ocean Res. 63, 262–275. https://doi.org/10.1016/j.apor.2017.01.015

Kim, J., Kim, M., Kim, D., 2011. Variants of the quantized visibility graph for efficient path planning. Adv. Robot. 25, 2341–2360. https://doi.org/10.1163/016918611X603855

Lee, S.M., Roh, M. Il, Kim, K.S., Jung, H., Park, J.J., 2018.

Method for a simultaneous determination of the path and the speed for ship route planning problems. Ocean Eng. 157, 301–312. https://doi.org/10.1016/j.oceaneng.2018.03.068

Li, P.F., Wang, H.B., He, D.Q., 2018. Ship weather routing based on improved ant colony optimization algorithm, in: Proceedings – 2018 IEEE Industrial Cyber–Physical Systems, ICPS 2018. Institute of Electrical and Electronics Engineers Inc., pp. 310–315. https://doi.org/10.1109/ICPHYS.2018.8387677

Ma, D., Ma, W., Jin, S., Ma, X., 2020. Method for simultaneously optimizing ship route and speed with emission control areas. Ocean Eng. 202, 107170. https://doi.org/10.1016/J.OCEANENG.2020.107170

Magid, E., Lavrenov, R., Afanasyev, I., 2017. Voronoi–based trajectory optimization for UGV path planning, in: 2017 International Conference on Mechanical, System and Control Engineering, ICMSC 2017. Institute of Electrical and Electronics Engineers Inc., pp. 383–387. https://doi.org/10.1109/ICMSC.2017.7959506

Nieuwenhuisen, D., Overmars, M.H., 2004. Useful cycles in probabilistic roadmap graphs, in: Proceedings – IEEE International Conference on Robotics and Automation. Institute of Electrical and Electronics Engineers Inc., pp. 446–452.

https://doi.org/10.1109/robot.2004.1307190

Niu, H., Savvaris, A., Tsourdos, A., Ji, Z., 2019. Voronoi-Visibility Roadmap-based Path Planning Algorithm for Unmanned Surface Vehicles. J. Navig. 72, 850–874. https://doi.org/10.1017/S0373463318001005

Noreen, I., Khan, A., Habib, Z., 2016. A Comparison of RRT, RRT* and RRT*-Smart Path Planning Algorithms. IJCSNS Int. J. Comput. Sci. Netw. Secur. 16, 20–27.

Organization, I.M., 2021. E-navigation [WWW Document]. URL https://www.imo.org/en/OurWork/Safety/Pages/eNavigation.aspx (accessed 4.23.21).

Overmars, M.H., Welzl, E., 1988. New methods for computing visibility graphs, in: Proceedings of the 4th Annual Symposium on Computational Geometry, SCG 1988. Association for Computing Machinery, Inc, New York, New York, USA, pp. 164–171. https://doi.org/10.1145/73393.73410

Samet, H., 1989. Applications of Spatial Data Structures: Computer Graphics, Image Processing and Gis [WWW Document]. URL https://www.amazon.com/Applications-Spatial-Data-Structures-Addison-Wesley/dp/020150300X (accessed 4.14.21).

Shah, B.C., Gupta, S.K., 2020. Long-Distance Path Planning for

Unmanned Surface Vehicles in Complex Marine Environment. IEEE J. Ocean. Eng. https://doi.org/10.1109/JOE.2019.2909508

Souissi, O., Benatitallah, R., Duvivier, D., Artiba, A., Belanger, N., Feyzeau, P., 2013. Path planning: A 2013 survey, in: Proceedings of 2013 International Conference on Industrial Engineering and Systems Management. pp. 1–8.

Tak, H.-T., Park, C.-G., Lee, S.-C., 2019. Improvement of RRT*-Smart Algorithm for Optimal Path Planning and Application of the Algorithm in 2 & 3-Dimension Environment. J. Korean Soc. Aviat. and Aeronaut. 27, 1–8. https://doi.org/10.12985/ksaa.2019.27.2.001

Tsou, M.-C., 2010. Integration of a Geographic Information System and Evolutionary Computation for Automatic Routing in Coastal Navigation. J. Navig. 63, 323. https://doi.org/10.1017/S0373463309990385

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

Abstract

Coastal path planning algorithm using quadtree and visibility graph

Wonhee Lee

Department of Naval Architecture and Ocean Engineering

The Graduate School

Seoul National University

With the introduction of e-Navigation technology by the International Maritime Organization, it has been recommended for ships to exchange data with maritime centers. For this purpose, the necessary information for ships is electronically exchanged for information exchange, and this information includes route information. Therefore, the vessel must make

a voyage plan and transmit it to the maritime center. Maritime center must check that there is no problem in the voyage plan and send the voyage plan to the vessel. Since weather routing system is established for ships sailing the ocean, it is easy to exchange route data between ships and maritime center. However, it is difficult to exchange route data for coastal ships because the navigator decide the route manually using their professional knowledge. In order to compensate the manual route planning along with the digitalization of the route, the demand for an automatic route system in the coast has increased.

Coastal ships include passenger ships, fishing boats, and yachts for various purposes. Vessels must calculate the route to their destination to achieve their objectives. The purpose of this study is to compute a safe and efficient route within a coast with a complex environment. It is necessary to secure the safety of the vessel by providing a route plan in consideration of the changing marine environment to the vessel in operation before departure or in real time. In order to determine a route, information such as shoreline, water depth, weather, and tide is required. Based on this information, many studies on route planning of ships

sailing the ocean have been developed, but most of them are focused on reducing fuel consumption and sailing time in open sea. Because the coastline is complex and there are many obstacles in the coast, it is difficult to apply the existing research in coastal environment. In addition, since there are many areas with a large tide difference in Korea, the environment changes frequently during navigation. Therefore, it is necessary to calculate the route within a short time to consider frequent environment change.

In this paper, we propose a novel route planning algorithm for complex offshore environments by integrating the advantages of quadtree and visibility graph. A quadtree is used to improve the inefficient environment representation method such as the uniform grid. The graph is constructed using quadtree and the shortest path is computed on the quadtree based graph. Using the waypoints of the previously computed route, a visibility graph is generated to calculate the optimal route. To find the optimal path, Dijkstra algorithm is used at this time. The proposed algorithm can perform route planning from any angle and establishes a safe route plan by maintaining a certain distance from the

coastline. In addition, the fuel consumption is computed in consideration of the weather information according to the departure time of the vessel, and the route with the minimum fuel consumption is planned. To calculate the fuel consumption, the braking power must be computed. The braking power can be calculated from the effective power, which is composed of the ship's resistance and speed. The resistance of the vessel is composed of resistance in still water, additional wind resistance, and additional wave resistance, and the net speed of the vessel. At this time, the additional resistance and the speed of the ship are computed based on ISO 15016. For validation of proposed algorithm, the minimum fuel consumption route and the shortest distance route were compared and evaluated. For performance evaluation, the algorithm proposed in this study was applied to the west and south sea of Korea. It was confirmed that the topology of the minimum fuel consumption route was different from the shortest route due to the influence of the weather information. Overall, it was confirmed that the proposed algorithm can generate the optimal route in complex environment with a short time.

Keywords: Coastal path planning, Polygon Map Random (PMR) Quadtree, Visibility Graph, Fuel Oil Consumption, Complex Marine Environment

Student Number : 2015-21172