

전자해도를 이용한 최적항로결정시스템에 관한 연구

하 희 천¹⁾, 柳 吉 洙²⁾

A Study on the Optimal Navigation Route Decision System Using Electronic Navigational Chart

Hee-Cheon Ha, Keel-Soo Rhyu

Abstract

One of the tasks of maritime navigation is to decide upon the optimal navigation route that minimizes a vessels travel time and fuel consumption. In the past, a navigator had to plot their navigation routes manually on seacharts using a ruler and a compass, but today faster plotting of seacharts has appeared owing to the development of electronic technique, so the navigation route can easily be plotted on ECDIS(Electronic Chart Display Information System) employing a mouse or a keyboard. Recently, problems have occurred as the decision of a navigation route on ECDIS requires the navigator to have expert knowledge of maritime navigation as well as computer systems.

Due to this situation, this paper has tried to develop a system that decides the optimal navigation route automatically simply by inputting the start point and destination point. The system decides the navigation route according to three criteria; 1) Utilizing the A* searching method for the optimal navigation route. 2) Applying the weighting function for an obstacle, that is an island, an land, etc. 3) Connecting of the possible direct path partly for fast searching.

The system displayed well an optimal navigation route on ECDIS about the Pusan coast as the target.

1) 한국해양대학교 대학원 전자통신공학과 석사과정, 컴퓨터공학 전공

2) 한국해양대학교 자동화·정보공학부 교수

1. 서론

컴퓨터의 기억장치 용량의 증가와 처리속도의 고속화에 힘입어 해도상의 각종 정보들을 전자데이터화하여 종합적인 항로구축이 가능한 전자해도(ECDIS: Electronic Chart Display Information System)^{[1]-[4]}가 등장하게 되었다. 전자해도는 종이 해도에 비해 원하는 구역이나 정보를 자유로이 확대하여 볼 수 있고, 수정보완이 용이하며 휴대,저장이 편리하다. 또한 첨단위성장비인 GPS(Global Positioning System)^{[5],[6]}와 결합하여 선박의 위치를 플로팅할 수 있게 되었다. 이러한 장점 때문에 전자해도의 사용은 날로 증가추세에 있다. 또한 선박의 운항에 있어서 최적항로를 결정하는 것은 항행시간과 연료의 소비를 줄이는 중요한 요인중의 하나이다. 이 때문에 최적항로를 결정하기 위해 종래에는 항해사들이 항해분야의 전문적인 지식을 기반으로 하여 종이 해도위에서 자와 컴퍼스를 이용하여 항로를 작성하였지만,^{[7],[8]} 오늘날에는 항로의 결정도 전자해도상에서 마우스나 키보드를 사용하여 쉽게 작성할 수 있게 되었다. 그러나 항해사가 전자해도상에서 항로를 결정하기 위해서는 항해분야 뿐만 아니라 컴퓨터분야에도 전문적인 지식을 가져야 한다는 새로운 문제점이 야기되었다.

본 논문에서는 이러한 배경으로부터 전자해도상에서 출발지점과 목표지점을 입력하면 자동으로 최적항로를 결정해 주는 시스템의 개발을 목표로 하여 최적항로를 결정하기 위해 세 가지 방법을 제안하였다. 첫째로 가장 최적인 탐색방법이라고 할 수 있는 A*탐색방법^{[12]-[16]}을 전자해도상에서 적용할 수 있도록 거리와 탐색깊이를 이용하여 평가함수를 만들고 이 함수에 의해 최적항로를 구하는 방법을 제안한다. 둘째로 A*탐색방법에 나타난 문제점들을 보완한 장애물가중치 적용법을 이용하여 최적항로를 구하는 방법을 제안한다. 셋째로 부분적으로 직선연결이 가능한 위치에서는 장애물가중치 적용법을 이용하지 않고 부분직선경로 연결법을 사용하여 탐색시간을 더욱 줄이도록 하는 방법을 제안한다.

이 세 가지 방법의 타당성을 입증하기 위해 Turbo-C 언어를 이용하여 시스템을 구현하였으며, 부산근해를 대상으로 하는 전자해도상에서 시스템을 실행시켜 본 결과 최적항로를 잘 결정해 주는 것을 확인할 수 있었다.

2. 전자해도와 탐색데이터구조

2.1 전자해도의 종류와 특징

오늘날 컴퓨터나 전자기술의 발달에 따라 해도정보를 브라운관상에 표시하고 여기에 레이더영상을 중첩시키거나 또는 다른 선위측정장치로부터의 정보를 입력시켜 브라운관상의 해도에 자동적으로 선위가 표시되게 하는 장치들이 개발되어 선박항해에 유용하게 이용되고 있다. IMO(International Maritime Organization)는 이러한 장치를 전자해도, 즉 ECDIS(Electronic Chart Display System)라 부르고 국제적인 기준을 작성하여 항해의 안전을 도모하는 작업을 진행하고 있다. 이러한 전자해도의 장점으로는 표시의 확장성, 표시구역의 선택성, 표시정보의 선택성, 표시의 식별성, 수정보완의 용이성, 휴대의 편리성 등을 들 수 있으며 급후 더욱 빨리 보급될 것으로 여겨진다.^{[1],[4]} 전자해도는 그 제작에 있어 크게 벡터(Vector)방식과 레스터(Raster)방식으로 나눌 수 있다.^{[1],[17]} 벡터방식이란 원하는 지역의 데이터를 마우스나 디지털타이저를 이용하여 입력한 그림의 형태에서 가장자리에 관한 정보만을 추출하여 저장하는 방법으로 한 점과 다음 점과의 위치 및 방향을 포함한 각종정보들로 이루어져 있다. 레스터방식이란 스캐너를 이용하여 그림의 전체를 이미지파일로 저장하고 화면의 출력을 이미지의 저장순으로 출력하는 것을 말한다. 본 논문에서 연구하고자 하는 내용은 전자해도상에서 두 점간의 유효항로를 결정하기 위한 것으로서, 이를 위해서는 화면상의 각 픽셀에 대한 정보가 필요하게 된다. 따라서 본 논문에서는 벡터방식의 전자해도 데이터를 레스터방식의 데이터로 변환한 전자해도데이터를 사용하고 있다.

2.2 탐색을 위한 데이터구조의 형태

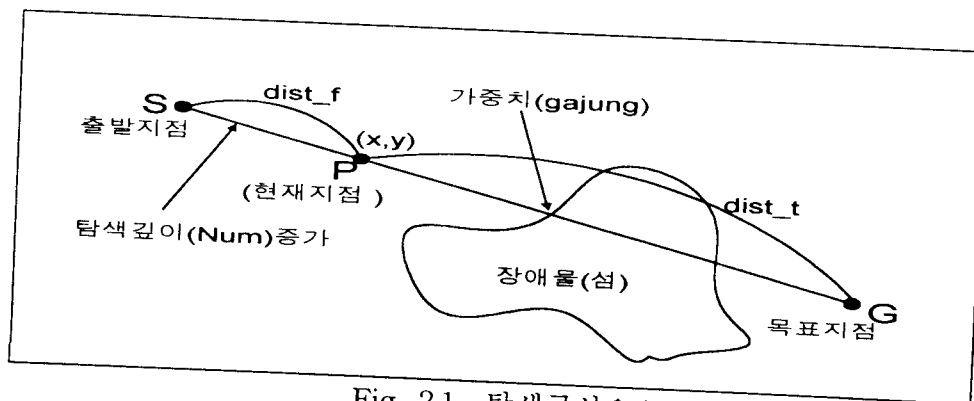


Fig. 2.1 탐색구성요소

탐색을 위해 필요한 탐색구성요소는 Fig. 2.1과 같다. 해도 화면의 가로와 세로좌표를 x, y로 한다. 그리고 출발지점에서 현재지점까지의 거리를 나타내는 변수(dist_f)가 필요하며, 현재지점에서 목표지점까지의 거리를 나타내는 변수(dist_t)가 필요하다. 또 탐색을 수행함에 있어 탐색을 몇 번째 하고 있는가 하는 탐색의 깊이를 나타내는 변수(num)가 필요하다. 그리고 각종 장애물, 즉 방파제, 섬, 암초, 수심선 등을 만났을 경우 가중치를 적용할 변수(gajung)가 필요하다. 이러한 조건들을 고려하여 탐색후보 한 점이 가져야 할 정보를 나타내는 구조체를 Fig. 2.2와 같이 선언하였고, 탐색해 온 경로를 저장할 변수의 구조체를 Fig. 2.3, 최종 최적경로의 해가 가져야 할 구조체를 Fig. 2.4와 같이 선언하였다. 이상과 같은 데이터구조를 가지고 경로를 탐색해 나가기 위해서는 출발지점으로부터 목표지점까지의 탐색이 진행되는 동안 계속 목표지점을 향하여 접근할 수 있어야 한다. 이를 위해서는 출발지점으로부터 현재지점까지 탐색해 온 거리와 현재지점으로부터 목표지점까지 남은 거리를 평가 함수로 이용할 필요가 있다. 이러한 탐색에 가장 유사하게 접근할 수 있는 방법의 하나로 본 논문에서는 A*탐색방법을 응용하여 이용하기로 한다.

```

struct HUBO_DBO {
    int x;          /* x좌표 */
    int y;          /* y좌표 */
    int num;        /* 탐색깊이 */
    int gajung;     /* 가중치 */
    long dist_f;   /* 발견함수=출발지점에서 현재지점까지의 거리 */
    long dist_t;   /* 거리함수=현재지점에서 목표지점까지의 거리 */
}hubo_list[1800]; /* 후보들을 저장할 메모리 */

```

Fig. 2.2 탐색후보 한 점이 가지는 구조체

```

struct HUBO_DB1 {
    int x;          /* 탐색후보의 x좌표 */
    int y;          /* 탐색후보의 y좌표 */
    int num;        /* 탐색후보의 탐색깊이 */
} bt_stack[1800]; /* 저장할 메모리 */

```

Fig. 2.3 탐색해 온 경로를 저장할 변수의 구조체

- ③ 바다가 아닌 육지나 해안선 등의 장애물을 나타내는 점들은 후보에서 제외시킨다.
- ④ 각 점들이 이미 후보로서 생성된 경우에는 제외시킨다.
- ⑤ 배는 항해를 할 때에 섬이나 육지해안선 등에 붙어서 다닐 수는 없기 때문에 해안선이나 장애물로부터 일정거리를 유지하는 것으로 한다.

3.2 A*탐색방법의 적용과 문제점

전자해도상에서 A*탐색방법을 이용하여 목표지점을 찾아가는 방법은 Fig. 3.2에서 보는 것과 같이 현재지점을 P라고 했을 때 이 지점까지의 탐색 깊이(dist_f)와 이 지점으로부터 목표지점까지의 거리(dist_t)와의 합이 최소로 되는 값을 최우선 후보로 선정하여 탐색해 가는 방법이다.

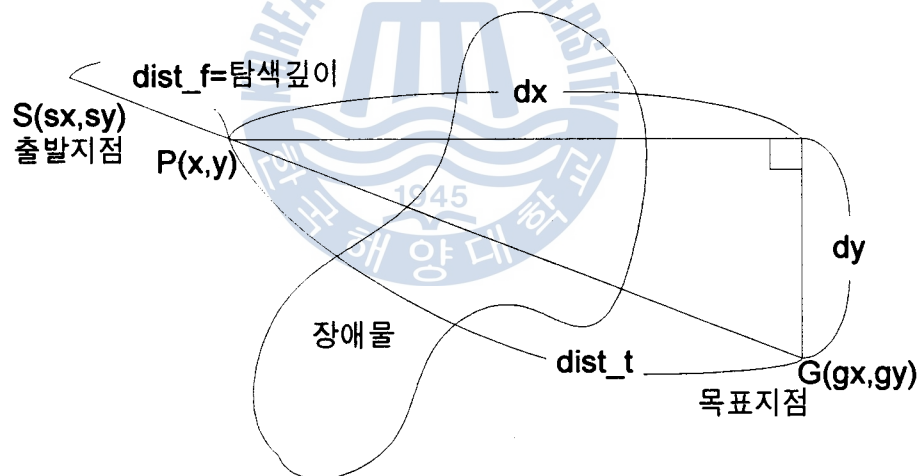


Fig. 3.2 A*탐색방법에 의한 탐색구조

즉, $f(x) = g(x) + h(x)$
 $f(x)$: 평가함수
 $g(x)$: 탐색깊이(dist_f)
 $h(x)$: 목표지점까지의 거리(dist_t)

로 하여 후보중 $f(x)$ 의 값이 작은 순으로 탐색을 해나가는 것이다. A*탐색방법을 전자해도상에 적용하여 보면 Fig. 3.3과 같은 꼴이 깊은 장애물을 만났

을 경우 너무 많은 후보의 생성으로 인하여 메모리 부족현상을 초래한다.

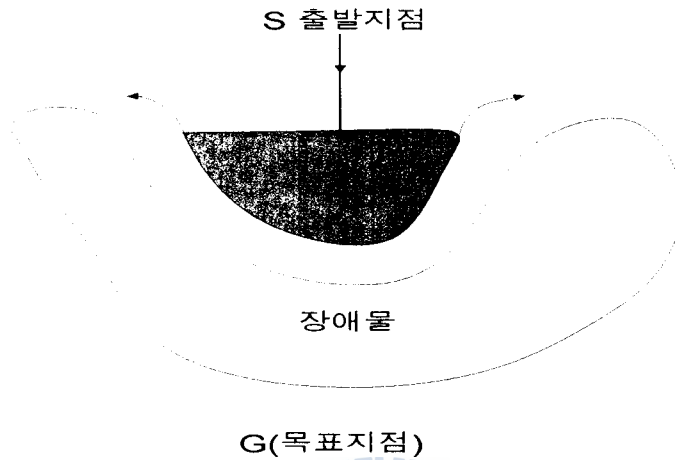


Fig. 3.3 골이 깊은 장애물을 만났을 경우의 후보생성

4. 발견적 탐색방법의 도입

4.1 장애물가중치 적용법

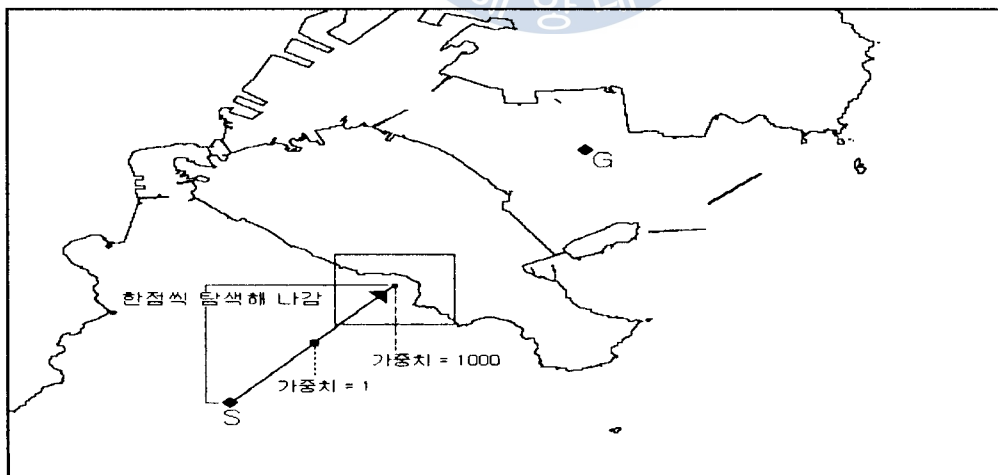


Fig. 4.1 장애물가중치 적용법

A*탐색방법만으로는 탐색을 수행함에 있어서 문제가 발생하는 것을 알 수

있었다. 이러한 문제를 해결하기 위해 장애물가중치 적용법을 도입하고자 한다. 즉, Fig. 4.1과 같이 장애물에 접근하지 않았을 때에는 첫 번째 방법과 동일한 A*탐색방법으로 탐색을 하지만 장애물로부터 일정거리까지 근접하게 되면 가중치를 크게 주어(=1000) 목표지점까지의 거리가 짧은 점보다 장애물에 가까운 점을 우선하여 탐색하게 하면 해안선으로부터 일정거리를 유지하면서 탐색하게 된다. 즉

$$f(x) = \alpha * (g(x) + h(x))$$

α : 장애물가중치

($\alpha = 1000$: 장애물로부터 일정거리에 도달한 탐색후보

$\alpha = 1$: 그 외의 모든 후보)

이렇게 하여 탐색을 하면 일단 골이 깊은 섬과 같은 장애물을 만났을 경우에는 이 골 안의 후보를 모두 탐색하는 것이 아니라 골의 해안선으로부터 일정 거리를 유지하는 지점들만이 탐색되고 골을 빠져나갈 수 있기 때문에 탐색후보가 많이 발생하는 것을 해결할 수 있다. 그러나 장애물가중치 적용법에 있어서도 문제점은 발생한다. 즉 Fig. 4.2와 같은 경우에는 장애물 주위를 필요 이상으로 탐색하게되어 탐색시간과 메모리를 많이 소요하게 된다.

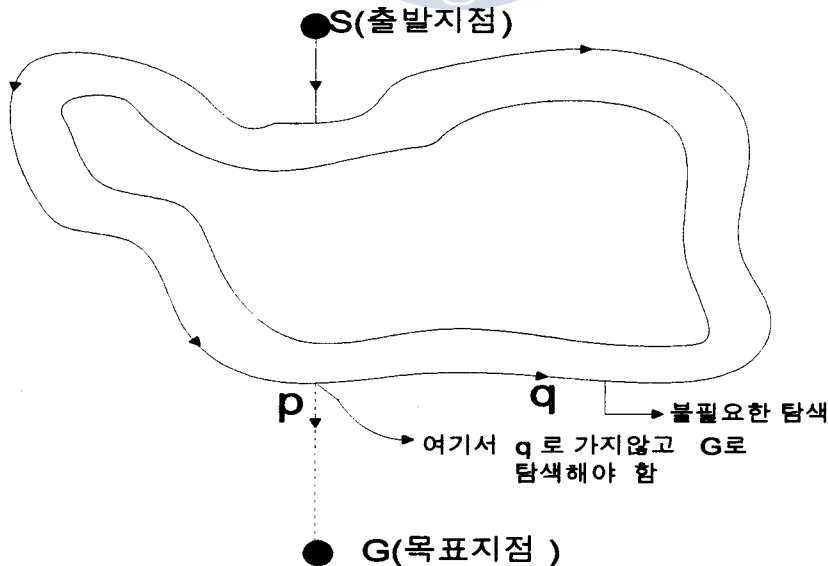


Fig. 4.2 장애물가중치 적용시의 불필요한 탐색발생

4.2 부분직선경로 연결법

장애물가중치 적용법만으로는 장애물 둘레를 필요이상으로 탐색하는 불필요한 탐색이 이루어지는 동시에 장애물이 매우 큰 경우에는 메모리의 부족을 일으킬 수가 있다. 그래서 이러한 문제점들을 해결하기 위하여 탐색의 후보 선정시 먼저 그 후보의 현재위치에서 목표지점까지 직선을 그어 이 직선이 장애물을 만나지 않는 최대지점까지는 바로 부분직선경로로 선정하는 것이다. 이러한 방법을 부분직선경로 연결법이라고 한다. Fig. 4.3과 같은 그림에서 S-P1, P2-P3, P4-G 구간은 목표지점을 향해 직선을 그었을 때 아무런 장애물이 없는 구간이다. 따라서 이 구간에서는 A*탐색법이나 장애물가중치를 적용법에 의한 탐색을 수행하지 않고 부분적으로 직선을 그어 이 직선상에 있는 점들을 탐색해로 간주하기로 한다.



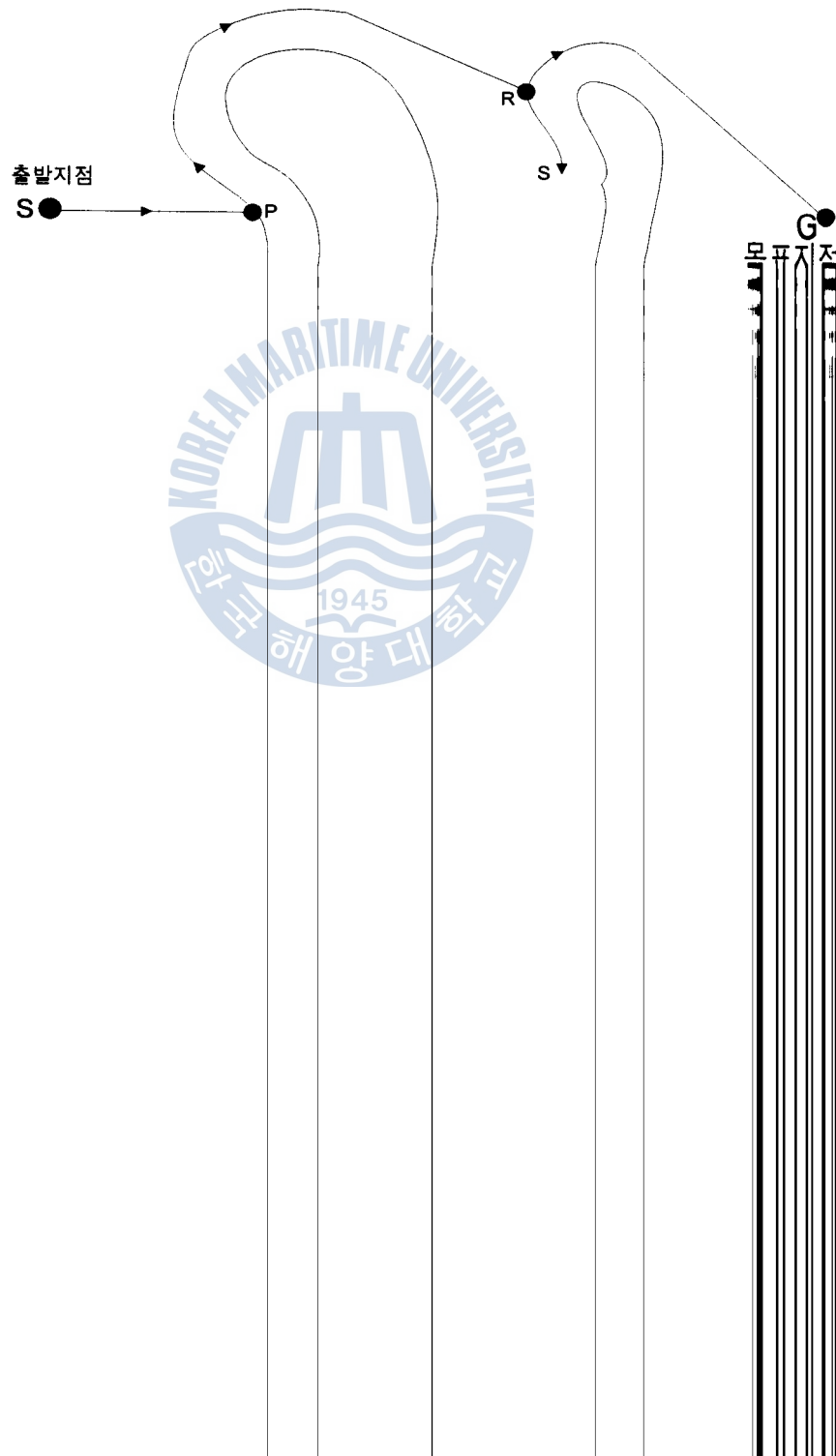
Fig. 4.3 장애물을 만나지 않는 최대지점까지의 부분직선경로 연결법

5. 최적항로의 도출

탐색리스트로부터 최적항로를 도출하기 위해서는 다음과 같은 두 단계의 과정이 필요하다.

R-S와 같이 불필요한 탐색경로를 제거한 경로, 즉 S로부터 G까지 하나의 선으로 이어지는 경로를 도출한다.

② 최적항로의 도출 : 유효탐색경로로부터 최적항로를 도출한다.



4.2 부분직선경로 연결법

장애물가중치 적용법만으로는 장애물 둘레를 필요이상으로 탐색하는 불필요한 탐색이 이루어지는 동시에 장애물이 매우 큰 경우에는 메모리의 부족을 일으킬 수가 있다. 그래서 이러한 문제점들을 해결하기 위하여 탐색의 후보 선정시 먼저 그 후보의 현재위치에서 목표지점까지 직선을 그어 이 직선이 장애물을 만나지 않는 최대지점까지는 바로 부분직선경로로 선정하는 것이다. 이러한 방법을 부분직선경로 연결법이라고 한다. Fig. 4.3과 같은 그림에서 S-P1, P2-P3, P4-G 구간은 목표지점을 향해 직선을 그었을 때 아무런 장애물이 없는 구간이다. 따라서 이 구간에서는 A*탐색법이나 장애물가중치를 적용법에 의한 탐색을 수행하지 않고 부분적으로 직선을 그어 이 직선상에 있는 점들을 탐색해로 간주하기로 한다.

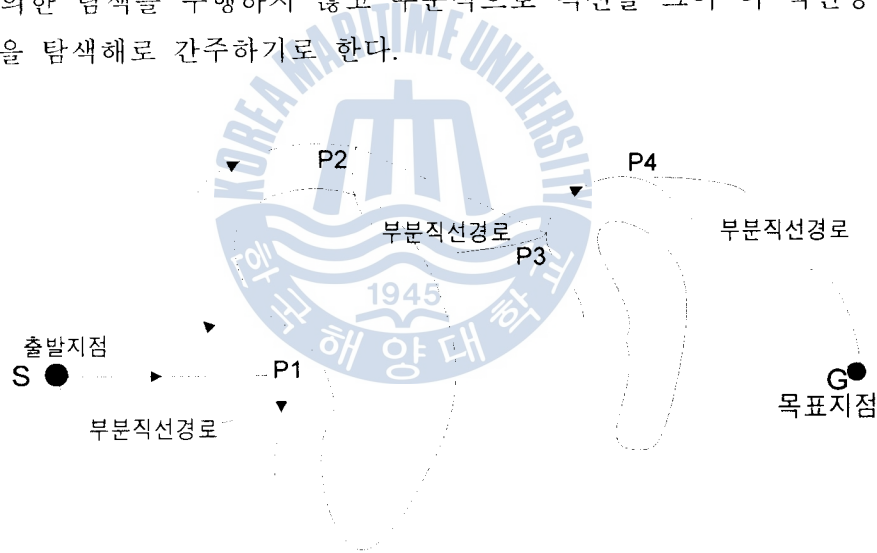


Fig. 4.3 장애물을 만나지 않는 최대지점까지의 부분직선경로 연결법

5. 최적항로의 도출

탐색리스트로부터 최적항로를 도출하기 위해서는 다음과 같은 두 단계의 과정이 필요하다.

- ① 유효탐색경로의 도출 : 탐색리스트 중에서 Fig. 5.1에 나타난 P-Q, 또는

Fig. 5.2와 같이 탐색깊이를 계산하여 탐색지점을 저장할 때 같이 저장하는 것이다. 그림에서 탐색리스트는 다음과 같이 저장된다.

((0 S), (1 a), (2 b), (3 c), (4 d), (3 e), (5 f), (4 g), (6 h), (5 i), (7 j),
(8 k), (9 l), (10 m), (11 G))

따라서 P방향으로 진행한 부분이 해를 발견한 경우이고 Q방향으로 진행한 부분이 해를 발견하지 못한 부분이므로 분기점으로부터 Q방향으로 진행한 탐색 경로는 다음과 같이 해서 제거시킬 수 있다.

- ① G에서 P방향으로 탐색리스트의 값을 역으로 탐색해 나간다.
- ② 탐색리스트에서 탐색깊이의 순서가 차례로 1씩 감소하는 지를 조사하여, 감소하면 다음 값을 조사한다. 즉 직전에 조사한 값의 탐색깊이가 n이라고 하면, 현재 조사중인 값의 탐색깊이가 n-1인가를 조사하여 같으면 계속적으로 반복한다.
- ③ 만일 n-1과 다른 m이라고 하면, 탐색깊이가 n-1과 같은 값이 나타날 때까지 조사한 값을 제거한다. 단 탐색깊이가 n-1이라 해도 직전에 조사한 값의 탐색깊이가 n이었다면 역시 제거한다.
- ④ ②와 ③의 작업을 탐색리스트의 출발지점까지 반복한다.

이 결과 탐색리스트에는 유효탐색경로상의 탐색위치의 값만이 저장된다. Fig. 5.2의 예에서 ①~④의 단계를 적용시킨 결과 다음과 같은 탐색리스트로 수정된다.

((0 S), (1 a), (2 b), (3 c), (4 d), (5 f), (6 h), (7 j), (8 k), (9 l), (10 m),
(11 G))

5.2 최적항로

Fig. 5.3을 보면 탐색리스트로부터 불필요한 탐색경로를 제거한 경로와 최적항로와는 상당한 차이가 있는 것을 알 수 있다. 이러한 결과는 장애물가중치 적용법을 이용한 탐색방법에 기인한 것이다. 따라서 유효탐색경로로부터 최적

항로를 도출해야 한다. 최적항로도출 방법의 하나로 본 논문에서는 Fig. 5.3과 같은 방법을 이용하고 있다. 유효탐색경로로 이루어진 탐색리스트에서 역으로

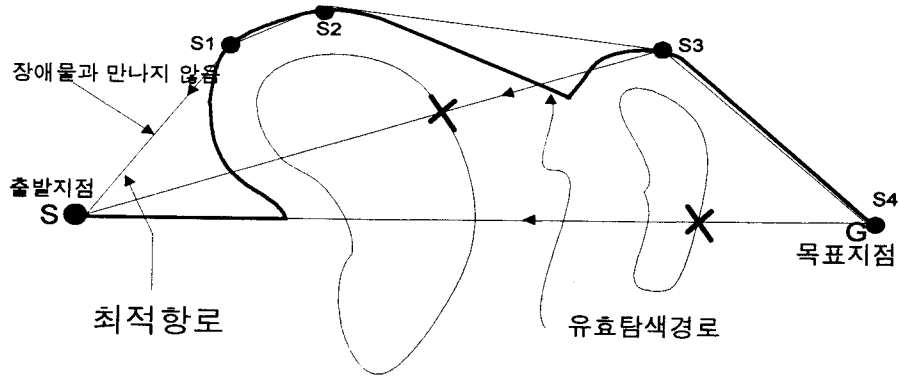


Fig. 5.3 유효탐색경로로부터 최적항로의 도출방법

목표지점으로부터 출발지점 S로 직선을 그어 장애물과 만나지 않는 가를 조사한다. 만나게 되면 목표지점으로부터 출발지점의 다음 탐색지점까지 직선으로 그어 다시 장애물과 만나는 지를 조사한다. 이런 조사를 반복하여 장애물과 만나지 않는 점이 발견되면, 목표지점으로부터 발견된 점까지를 직선으로 연결한다. 이후 발견된 점으로부터 출발지점과를 직선으로 연결하는 조사를 수행하고, 이런 과정을 반복 수행한다. Fig. 5.3에서는 우선 S4에서 S로 직선을 그어보고 장애물과 마주치므로 S의 다음 탐색지점과 S4를 그어보며, 장애물과 마주치지 않는 탐색지점이 발견될 때까지 이런 반복을 계속한다. 즉 S3 지점까지 반복하게 된다. 이번에는 S3지점에서 출발지점으로 직선을 그어보고 다시 장애물과 마주치지 않는 S2지점이 발견될 때까지 반복한 후, 다시 S2와 S에 대하여 직선을 그어본다. 이렇게 해서 최종적으로 S-S1-S2-S3-S4라는 직선경로가 얻어지게 되며, 이 경로가 최적항로로 된다.

6. 시스템 구현 및 고찰

6.1 시스템의 구현

본 시스템의 환경은 다음과 같이 구성하였다. 프로그램은 MS-DOS환경에서 컴파일 했다. 입력장치로서 입력이 간편한 마우스를 선택하였고, 실행속도

를 빠르게 하기 위하여 PC의 CPU는 586-Pentium(120Mhz)을 사용하였으며, PC의 메모리는 8MB이다. 화면의 해상도는 640×480 16Color VGA모드를 사용하였으며 프로그램 컴파일러는 TC(Turbo-C) 2.0을 사용하였다. TC의 메모리모델은 Large Model로 하였다.

6.2 검토 및 고찰

표 6.1은 동일한 출발지점과 목표지점에 대하여 A*탐색법, 장애물가중치 적용법, 장애물가중치 적용법에 부분직선경로 연결법을 도입한 방법을 전자해도 상에서 각각 실행하여 본 것이다. 탐색에 소요되는 시간, 후보의 수, 탐색해운 경로를 저장하는 탐색리스트의 수를 비교하여 볼 때 장애물가중치 적용법에 부분직선경로 연결법을 도입한 방법이 가장 우수하다는 것을 알 수 있다.

표 6.1 세 가지 탐색방법에 대한 실행결과 비교표

출발지점(S) 목표지점(G)	탐색방법 (결과그림)	후보수	탐색리스트 수	탐색시간
S:(150, 290) G:(288, 84)	방법① (Fig. 6.1)	1113	2000	탐색실패
	방법② (Fig. 6.2)	2023	1461	26초
	방법③ (Fig. 6.3)	999	868	22초
S:(72, 247) G:(377, 251)	방법①	823	1479	20초
	방법②	881	608	10초
	방법③	461	482	6초
S:(381, 377) G:(261, 112)	방법①	1117	2000	탐색실패
	방법②	1453	1075	19초
	방법③	873	751	14초

(방법①: A*탐색방법. 방법②: 장애물가중치 적용법.
 방법③: 장애물가중치 적용법 + 부분직선경로 연결법.)

Fig. 6.4는 장애물가중치 적용법과 부분직선경로 연결법에 의한 탐색으로 실제탐색경로와 최적항로사이에는 상당한 차이가 있지만, 최적항로 도출알고리즘을 적용하여 최적항로를 도출해 낸 좋은 결과를 보여주고 있다.

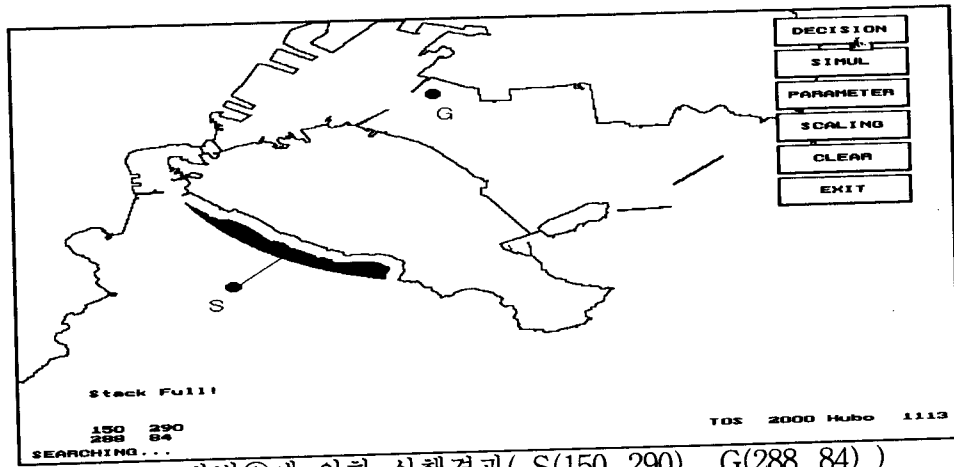


Fig. 6.1 방법①에 의한 실행결과(S(150, 290), G(288, 84))

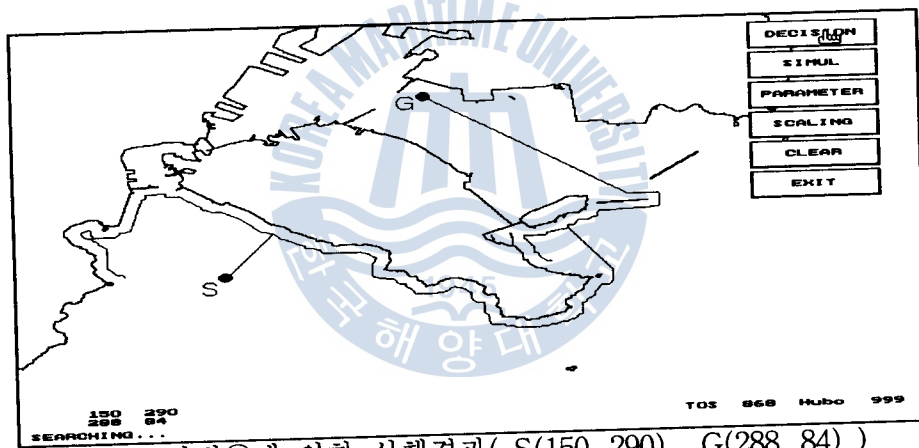


Fig. 6.2 방법②에 의한 실행결과(S(150, 290), G(288, 84))

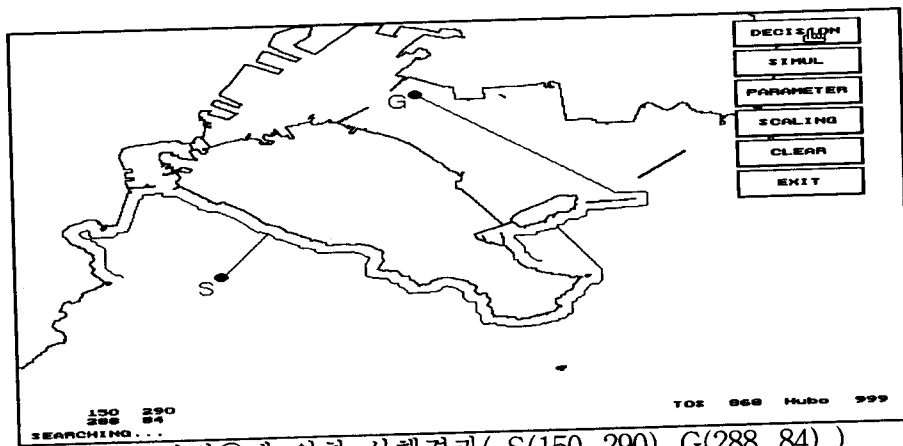


Fig. 6.3 방법③에 의한 실행결과(S(150, 290), G(288, 84))

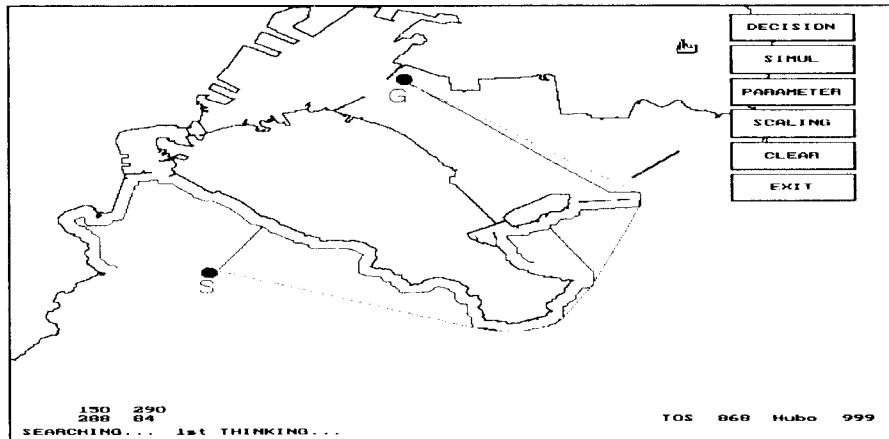


Fig. 6.4 최적항로의 도출 결과

7. 결론

본 논문에서는 전자해도상에서 출발지점과 목표지점을 입력하면 최적항로를 결정할 수 있는 세 가지 방법, 즉 ①A*탐색방법을 이용한 항로결정알고리즘, ②장애물가중치 적용법을 이용한 항로결정알고리즘, ③부분직선경로 연결법을 도입한 항로결정알고리즘을 제안하였고 시스템을 구현하였다. 시스템의 실행 결과 다음과 같은 결론을 얻었다.

- 1) 전자해도상에서 각 픽셀을 비용으로 환산하고, 해도에 관한 경험적인 지식을 이용함으로써 A*탐색법을 최적항로를 결정하는 시스템에 적용할 수 있었다.
- 2) A*탐색법을 적용하면 메모리의 부족과 탐색시간이 많이 걸린다는 것을 알 수 있었으며, 이를 보완하기 위해 장애물가중치 적용법을 이용함으로써 탐색시간을 줄일 수 있었다.
- 3) 부분직선경로 연결법을 도입하여 부분적으로 직선연결이 가능한 위치에서는 장애물가중치 적용법을 이용하지 않고 직선경로로 이어 줌으로써 탐색시간을 더욱 줄일 수 있었다.

그러나 최적항로결정은 해도 한 장을 대상으로 하였기 때문에 축척이 다른 여러 장의 지도가 있을 때에도 항로를 결정할 수 있는 방법, 또한 변침점에서

의 각도가 너무 클 경우 변침점을 늘려 각도를 완만하게 함으로써 실제 항해가 가능한 항로를 결정해주는 방법 등에 관해서는 금후 더욱 검토되어야 할 사항이라고 사료된다.

참 고 문 헌

- [1] Adam. J. Kerr, "INTERNATIONAL PERSPECTIVES ON ECDIS", 전자시대와 수로국의 발전전략 국제심포지움 자료집 pp9-22., 1996.
- [2] Takahiro SATO. PhD, "HYDROGRAPHY IN THE AGE OF ELECTRONIC NAVIGATIONAL CHART", 전자시대와 수로국의 발전전략 국제심포지움 자료집 pp37-43., 1996.
- [3] Arthur G. Gaines, "US electronic chart display and information system(ECDIS) test bed project : overview and update", 전자시대와 수로국의 발전전략 국제심포지움 자료집 pp53-70., 1996.
- [4] 서상현, "우리나라 전자해도 구축방안", 전자시대와 수로국의 발전전략 국제심포지움 자료집 pp85-91., 1996.
- [5] Bradford W. Pakinson, James J. Spiker Jr., "Global Positioning System Theory and Application, vol 1,2", AIAA, Inc., 1996
- [6] Alfred Leick, "GPS Satellite Surveying", Second Edition, John Wiley & Sons, Inc., 1994.
- [7] 윤여정, "천문항해학", 한국해양대학교 해사도서출판부, 1973.
- [8] 윤여정, "지문항해학", 한국해양대학교 해사도서출판부, 1973.
- [9] 김세환, 박상현, 윤상준, 이상정, "차량항법을 위한 GPS/DR 통합 칼만 필터", '96 GPS WORKSHOP의 논문집, 1996.
- [10] 김욱, 이장규, 김현수, "차량항법장치에서의 지도매칭법을 이용한 향상된 위치 결정", '96 GPS WORKSHOP의 논문집, 1996.
- [11] 조현정, "Auto Navigation Simulation System", 비트프로젝트10호, BIT 바아이티출판, pp13-31, 1996.
- [12] ながお まこと, "知識と推論", 岩波書店, 1988.
- [13] P.H. Winston, "인공지능", 도서출판 生能, 1990.

- [14] 김재희, “인공지능의 기법과 응용”, 교학사, 1991.
- [15] 유석인, “인공지능 원론”, 교학사, 1988.
- [16] 최종수, “인공지능의 세계”, 방한출판사, 1987.
- [17] 김은형, “GIS Lecture II”, 1994.



