韓國 OA學會 論文誌
第4卷 第1號, 1999. 3.

99-4-1-2-3

A* 알고리즘을 이용한 최적항로결정에 관한 연구

정 정 수*, 류 길 수**

A Study on The Optimal Navigation Route Decision using A*Algorithm

Jung-Jung Su*, Rhyu-Keel Soo**

요 약

선박에서 항해시간을 줄이고 연료소비를 절감하기 위하여 최적항로의 결정하는 작업은 선박운항의 가장 중요한 요소이다. 최근, 보다 빠르고, 정확한 최적항로를 결정하기 위해 전자해도시스템이 개발되고 있다. 본 논문은 이러한 시스템에 탐색의 알고리즘 중 최선의 탐색해를 제공하는 A*알고리즘을 전자해도 시스템에 적용하였다. 그러나 A*알고리즘의 적용은 과도한 탐색시간과 많은 메모리를 요구하는 문제점이 있다는 사실이 발견되었고 이러한 문제점을 해결하기 위해 장애물을 발견했을 때 탐색 후보 선정에 가중치를 부여하는 보다 개선된 알고리즘의 필요성을 제안하고자 한다.

Abstract

One of the tasks of maritime navigation is to decide upon the optimal navigation route that minimizes a vessals travel time and fuel consumption. Recently, ECDIS(Electronic Chart Display Information System) is used to decide the optimal navigation route and have expert knowledge of maritime navigation.

In this paper, the system use A*algorithm for optimal navigation route on ECDIS. But some problems is discovered in this situation. it requires many memory device and searching time.

So this paper has tried to develope a advanced algorithm system that decides the optimal navigation route.

* 동명정보대학교 컴퓨터공학과 조교수
** 한국해양대학교 자동화·정보공학부 교수
논문접수 : 98. 12. 28. 심사완료 : 99. 2. 13.

Ⅰ. 서 론

지금까지의 선박운항은 해도위에서 목적지까지의 항로를 계산하고 작도하여 운항계획을 잡아 왔는데 전자기술의 발달로 이러한 해도를 전자 데이타화하여 해도의 검색과 각종 정보들을 한데 모아 종합적인 항로설정을 위한 시스템으로 개발되고 있다.

이러한 배경으로 본 논문에서는 효율적인 항행경로를 빠르게 구하기 위해 이를 PC상에서 구현하여 보았다. 최적항행경로를 구하기 위한 탐색법으로는 A*탐색법을 적용하였다. 실제로 A*탐색법은 출발지점에서 현재지점 P까지 이르는 최소의 값 g(p)와 현재지점에서 목적지점까지를 최소로하는 값 h(p), 이 둘을 합한 g(p)+h(p)를 가장 최소로 하는 값을 최우선으로 하여 탐색해 가는 방법이다. 그러나 이러한 방법 그대로를 전자지도상에 적용하여 보면 PC상에서 많은 메모리를 요구함과 동시에 탐색시간이 상당히 많이 걸린다는 것을 알 수 있었다. 그래서 실험에서는 해안선이나 섬 등의 장애물로부터 일정거리에 도달하면 이들 둘레를 먼저 탐색해 갈 수 있도록 가중치를 적용하는 것과 같은 개선된 알고리즘의 필요성을 제안하고자 한다.

Ⅱ. 탐색을 위한 데이터구조의 형태

본 논문에서는 전자해도상에서 출발지점과 목적지점을 입력한 상태에서 항로를 탐색하기 위해 필요한 구성요소로서 Fig. 1과 같은 요소들을 이용하기로 한다.

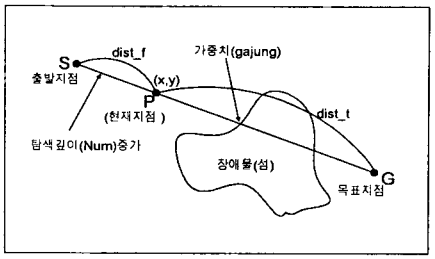


Fig. 1 탐색구성요소

해도 화면의 가로를 x좌표로 하고 세로를 y좌표로 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 몇번째 하고 있는가 하는 탐색의 깊이를 나타내는 변수(num)가 필요하다. 그리고 장애물(방파제, 섬, 암초등)을 만났을 경우 가중치를 적용할 변수(gajung)가 필요하다.

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

Fig. 2 탐색후보 한 점이 가지는 구조체

이러한 조건들을 고려하여 탐색후보 한 점이 가져야 할 정보를 나타내는 구조체를 Fig. 2과 같이 선언하였다. 그리고 실제로 해를 탐색해 나갈때 탐색해온 경로상의 탐색후보들을 저장할 메모리변수가 필요하다. 이 후보들은 좌표점 x 및 y 출발지점으로부터 몇번째로 탐색했는가 하는 탐색깊이에 관한 정보만 있으면 된다. 따라서 Fig. 3과 같이 선언할 수가 있다.

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

Fig. 3 탐색해 온 경로를 저장할 변수의 구조체

탐색해온 탐색후보리스트(bt_stack)의 각 후보들은 가능한 모든 방향으로 탐색한 것이기 때문에 불필요한 탐색경로의 데이터들이 들어 있을 수 있다. 그래서 이러한 불필요한 점들을 제거한 마지막 최종항로를 구할 때에는 각 후보들의 x 및 y 좌표만 있으면 된다. 따라서 Fig. 4와 같이 선언할 수가 있다.

```
struct XY_CORD {
int x; /* x 좌표 */
int y; /* y 좌표 */
} filt_stack[1000];/* 저장할 메모리 변수 */
```

Fig. 4 최종 최적경로의 해가 가져야 할 구조체

이상과 같은 데이터구조를 가지고 경로를 탐색해 나가기 위해서는 출발지점으로부터 목표지점까지의 탐색이 진행되는 동안 계속 목표지점을 향하여 접근할 수 있어야 한다. 이를 위해서는 출발지점으로부터 현재지점까지 탐색해온 거리와 현재지점으로부터 목표지점까지 남은 거리를 평가함수로 이용할 필요가 있다. 이러한 탐색에 가장 유사하게 접근할 수 있는 방법의 하나로 본 논문에서는 A*탐색법을 응용하여 이용하기로 한다.

Ⅲ. A*탐색방법의 적용

탐색에는 횡형탐색(breadth-first-search), 종형탐색(deapth-first-search), 최소비용탐색, 등산법 등 여러가지가 있다. 여기에서는 비용을 고려한 탐색들을 살펴보고, 가장 최적인 탐색방법이라고 할 수 있는 A* 탐색방법을 전자해도상에 적용하는 방법에 대하여 기술한다.

1. Cost를 고려한 탐색

탐색에는 일반적으로 많은 시간이 걸린다. 어떤 상태로부터 다음 상태로 이동하는 상태가 p개 있다고 한다면 n회 이것을 적용하는 것에 따라 pn개의 상태가 만들어 진다. 그러므로 횡형탐색을 n단의 깊이까지 수행하는 경우에 저장해야 할 탐색항목의 수는

$$1 + p + p2 + \cdots + pn = \frac{p^{n+1}-1}{p-1}$$

로 된다. p=3, n = 10이라 하면 이것은 $\frac{1}{2}(3^{11}-1) \approx 105$이라는 큰 수로 되어버린다. 이것을 종형탐색으로 하는 경우, n단의 깊이까지 수행한 때의 스택의 깊이는 기껏해야

$$1 + (p-1) \times n + 1 = n(p-1) + 2$$

이하로 되어 p, n이 클 때의 횡형탐색에 비해서 탐색항목의 수가 대단히 작은 것을 알 수가 있다. 그러나 어느 쪽의 경우도 실용적인 측면에서는 결점이 있다. 종형탐색은 해가 없는 방향으로 무한히 전진하여 해에 도달할 수 없다고 하는 위험성이 있고, 횡형탐색의 경우에는 방대한 양의 탐색항목을 저장하지 않으면 안된다고 하는 것이다. 그래서 주어진 문제에 관계되는 얼마간의 지식을 이용하여 탐색의 범위를 좁혀 효율적으로 해를 발견하는 방법이 검토되었다. Fig. 5에 표시한 것과 같은 도로망의 지점 A로부터 I에 이르는 최단거리를 구하는 문제를 생각해 보자.

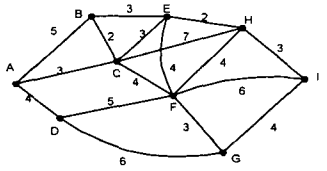


Fig. 5 최단거리탐색문제, 숫자는 경로의 길이

최단거리 탐색문제의 해법으로서 종형탐색을 취하면

목적지의 방향으로 향하여 잘 전진하고 있는 경우는 좋지만 만약 잘못된 길로 전진해버려 목적지에 도달할 수 없다고 할 때에는 곤란하다. 그러므로 횡형탐색으로 진행해 가는 방법이 안전하다. 그러나 이 예는 인접점 간의 길이가 주어져 있기 때문에 이 정보를 이용하는 것이 좋다.

그 하나의 방법은 출발지점으로부터 현재지점까지의 거리가 최소인 점을 우선적으로 받아들여 그 점으로부터 가능한 점으로 가는 것을 시도해 본다. 그 점이 이미 과거에 도달된 점이라면 거기까지의 거리가 과거에 도달했을 때의 경로의 거리와 어느 쪽이 작은가를 비교하여 작은 쪽의 값을 그 지점까지의 값으로 한다. 현재 다다른 지점이 아직 한 번도 도달되지 않은 지점이라면 거기까지의 경로의 값을 기입한다.

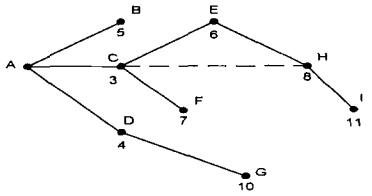


Fig. 6의 최단경로탐색으로 만들어지는 트리

Fig. 5에 대한 탐색의 결과로 Fig. 6과 같은 트리 구조가 얻어진다. 여기에서 점선의 경로는 E부터 H까지의 경로가 새롭게 발견됨으로써 제거된 탐색경로이다. 이렇게 하여 출발점으로부터 목적지까지의 최단거리를 구하는 것이 가능하다. 경로의 길이를 어떤 상태로부터 다음 상태로 이행하기 위한 비용으로 생각하면, 이 방법으로 일반 탐색 문제에 대한 해를 얻을 수가 있다. 이것을 최소경로탐색이라 부르고 있다.

비용을 고려한 탐색경로에 대한 비용의 합계를 비용함수 g(n)으로 표시하자.또 실제로 초기상태로부터 현재상태 까지 더듬어 탐색해온 경로에 의해 최적인 경로의비용함수를 $\overline{g}(n)$로 표시하는 것으로 한다. g(n)과 $\overline{g}(n)$과는 반드시 동일하게는 되지 않는다. g(n)은 이상적인 경우이기 때문에 일반적으로 $\overline{g}(n) \geq g(n)$이다. 예를 들면 Fig. 7에 표시한 망의 트리와 같은 도로를 통해서 출발지점으로부터 현재지점까지 찾아오는 경우의 비용을 생각해 본다면 $\overline{g}(n) > g(n)$의 관계가 잘 이해가 될 것이다. 그림에서 2중선은 현재까지 탐색한 경로내의 최소 비용을 주는 경로이고 이 경로에 해당하는 비용이 $\overline{g}(n)$이다 . 이 경로로 가지 않고, 아직 조사하지 않은 경로로서 $g(n) \leq \overline{g}(n)$을 주는 것이 있을지도 모른다. 탐색의 그래프가 루프를 하지 않고 트리의 구조를 하고 있을 때에는 초기상태로부터 n에 이르는 길은 하나이고, 다른 것은 없기 때문에 $g(n) = \overline{g}(n)$이다.

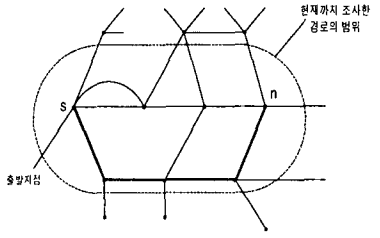


Fig. 7 비용을 고려한 탐색.

2. 목표지점까지의 예측 비용을 고려한 탐색

최단거리문제의 방법은 출발지점으로부터 현재지점까지의 비용이 최소인 것을 우선적으로 선택하여 목적지의 방향으로 탐색을 진행하는 것이었다. 그것은 목적지까지의 비용이 최소인 경로를 구하기 위해 취한 방법이다. 그러나 어떤시점으로서 비용이 최소인 점이 목적지에 가장 가까운 점이라는 보증은 없다. 최단거리문제인 Fig. 5에서 점 A로부터 출발하여 B, C, D, E, F, G, H까지의 최소 비용의 경로를 조사한 시점에서, 비용이 가장 작은 F로부터 전개의 가능성을 조사하게 되며 다음으로 목적지 I에 도달하게 된다. 그러나 실제로는 H쪽이 목적지에 훨씬 가까우며 H쪽을 우선하여 조사하는 것이 바람직하다.

이 그림에서 지금까지 생각해온 것과 같이 출발지점으로부터 현재지점까지의 비용이 최소인 경로를 우선하여 전개해 가지 않고 도달한 현재지점으로부터 목적지까지의 평가함수 값이 최소인 점으로부터 우선하여 전개하여 다음의 새로운 지점으로 전개해 가는 편이 좋다. 그러나 일반적으로 현재 지점으로부터 목적지까지의 비용은 명확하게 주어지지 않는 경우가 많다. 도로지도의 최단거리 문제의 경우에는 목적지까지의 직

선거리가 가까운 것이 하나의 목표로 되겠지만 정확한 비용은 알 수 없다. 그러나 문제에 따라서는 현재 상태로부터 목적지까지의 비용을 예측할 수가 있는 경우가 있고 이것을 이용한 탐색을 생각할 수가 있다. 이 목적지까지의 예측 비용을 h(n)으로 표시한다. 여기서 n은 현재지점을 나타내고 있다.

Fig. 8에 표시한 미로 문제에서 입구로부터 출구까지의 최단거리를 구하는 문제에 대하여 고려해 보려고 한다. 여기에서 좌표계는 입구를 원점으로하여 아랫방향으로 x축, 오른쪽 방향으로 y축을 취하는 것으로 한다. 또 출구를 (x0,y0) = (0,6)으로 하고 x,y는 현재지점을 나타내며 언제나 양의 정수값인

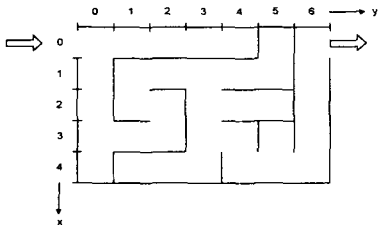


Fig. 8 미로 문제

$0 \leq x \leq 4$, $0 \leq y \leq 6$으로 가정한다. 예측 비용의 식은 다음과 같이 된다.

$$\bar{h}(x,y) = |x-x0| + |y-y0| = x - y + 6$$

이 경우의 탐색 트리는 Fig. 9에 나타낸 것과 같은 형으로 된다.

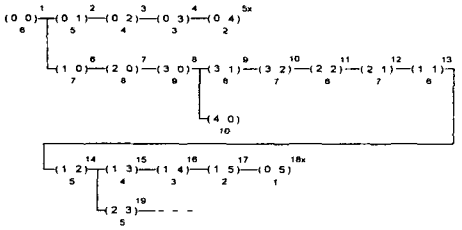


Fig. 9 $\bar{h}$를 이용한 미로문제의 탐색트리

여기에서 괄호위에 쓰인 숫자는 탐색의 순서를 표시하고 괄호아래의 숫자는 h의 값을 나타낸다. $\bar{h}(x,y)$의 값이 작은 순으로 종형탐색법으로 전개하고 있지만 이 탐색트리는 대단히 불필요한 수순을 밟고 있다. 그것은 목적지까지의 예측비용함수가 적절하지 않은 것이 원인이다. 만약 제 1의 임시 목적지를 (3,3)으로 하게 되면 예측비용함수는 $\bar{h}_3(x,y) = |x-3| + |y-3|$으로 되며, 제 2의 임시 목적지를 (4,6)으로 한다면 그때의 예측비용함수는

$\bar{h}_2(x,y) = |x-4| + |y-6|$으로 된다. 그래서 그림의 입구로부터 제 1의 임시 목적지, 제 2의 임시 목적지, 출구순으로 탐색한다고 한다면 이때의 탐색트리는 Fig. 10과 같이 되어 불필요한 탐색부분을 상당히 줄일 수 있는 것을 알 수 있다. 이 경우는 $\bar{h}_3(x,y)$, $\bar{h}_2(x,y)$, $\bar{h}(x,y)$의 값이 작은 순으로 탐색의 이동이 수행되고 있다.

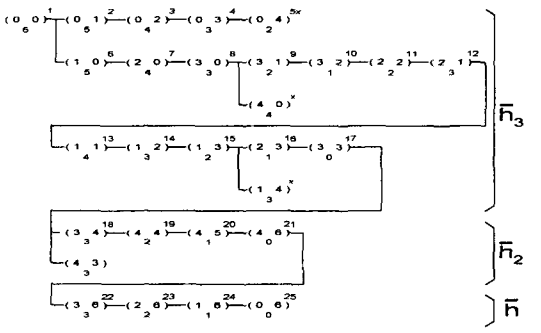


Fig. 10 중간 목적지(3,3), (4,6)을 둔 경우의 탐색트리

이와같이 문제의 해에 이르는 경로상에서 확실히 통하는 중간 위치를 알 수 있다면 문제를 2개의 부분문제로 나눌 수가 있다. 일반적으로 부분문제는 문제전체보다는 간단하기 때문에 해도 쉽게 얻어진다.

3. A* 탐색

이상에서 서술한 두 개의 방법을 조합하면 더욱 정확하게 최소비용경로를 도출할 수 있다. 즉 제1의 방법은 출발지점 S로부터 현재지점 p까지에 필요한 비용 g(p)를 최소로 하면서 목표로 향해가는 방법이고 제2의 방법은 현재지점 p로부터 목표지점까지의 평가함수 h(p)를 최소로하는 경로를 찾는 방법이므로 이들 2개의 비용을 합해서 f(p) = g(p) +h(p) 라는 함수를 최소로 하는 p를 찾으면서 탐색해 가는 방법을 이용하는 것이다. 만약 모든 점 p에 대해 g(p)와

h(p)를 정확히 알고 있다면 최적경로를 정확하게 찾아 목표지점에 도달할 수 있다. 그러나 일반적으로 g와 h는 정확히 알 수 없는 추정치이기 때문에 시행착오적인 탐색이 필요하게 된다.

출발지점 S로부터 탐색하여 탐색트리가 p지점까지 왔다고 할 때 탐색한 범위 내에서의 최소 비용은 알 수 있기 때문에 이것을 $\bar{g}(p)$로 표현하자. 다른 경로로서 거듭 작은 값을 줄 가능성이 남아 있기 때문에 $\bar{g}(p) = g(p)$는 아니다. p지점으로부터 목표지점까지의 비용은 탐색해 보지 않으면 알 수 없기 때문에 이것은 추정치 $\bar{h}(p)$이다. 그러므로 점 p를 통한 최적인 경로의 예측비용 추정치는 $\bar{f}(p) = \bar{g}(p) + \bar{h}(p)$로 쓰게 된다. 이것을 비용함수로서 탐색을 행하는 방법을 A탐색이라 한다. A탐색은 다음과 같은 특징을 가지고 있다.

① A탐색은 반드시 최적해를 주지 않는다.

② 최적해가 반드시 얻어질수 있게 하기 위해서는 $\bar{h}(p) \leq h(p)$라는 조건이 필요하며, 이것을 만족하는 A탐색을 A*탐색이라 부른다.

③ 탐색그래프가 무한한 경우에도 출발지점으로 부터 목표지점까지의 경로가 존재한다면 A*탐색은 반드시 최적인 경로를 찾아낼 수가 있다.

④ 탐색그래프 전체의 점에 대해 $h(p) \geq \bar{h}_1(p) > \bar{h}_2(p)$라면 $\bar{h}_1(p)$의 쪽이 $\bar{h}_2(p)$보다 지식이 많다고 알려져 $\bar{h}_1(p)$에 의해 탐색되어지는 점집합은 $\bar{h}_2(p)$에 의해 탐색되어지는 점집합에 포함되어 진다. 즉 탐색이 작게 끝난다.

전체의 점에 있어서 $\bar{h}_1(p) = h(p)$라면 불필요한 노드를 전개하지 않고 해를 얻을 수가 있다.

4. A*탐색방법의 전자해도상에의 적용

A*탐색방법을 전자해도상에 적용해 보고 그 문제점이 무엇인지 검토해 보기로 한다.

4.1 탐색후보의 생성

먼저 탐색이 한 지점에서 이루어질 때 이 지점으로부터 탐색해 갈 수 있는 방향은 8 방향이 된다. 즉 Fig. 11처럼 8방향으로 탐색이 이루어지기 때문에 한 점에 대하여 1, 2, 3, 4, 5, 6, 7, 8의 8개의 후보가 생성되는 것을 알 수가 있다. 본 논문에서는 후보를 생성함에 있어서 다음과 같은 제약조건을 도입하기로 한다.

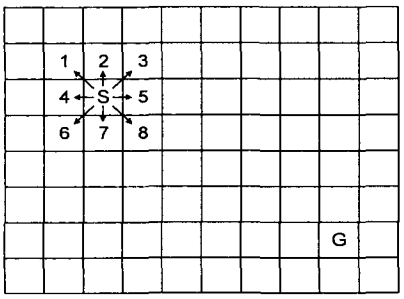
① 전자해도상의 지도를 픽셀로 나타내고 각 픽셀을 동일크기의 정사각형으로 간주한다.

② 현재지점으로부터 인접한 8방향의 픽셀(점)을 후보로 가정하며, 각 후보들까지의 비용은 동일한 것으로 한다.

③ 바다가 아닌 육지나 해안선등의 장애물을 나타내는 점들은 후보에서 제외시킨다.

④ 각 점들이 이미 후보로서 생성된 경우에는 제외시킨다.

⑤ 배는 항해를 할 때에 섬이나 육지해안선등에 붙어서 다닐 수는 없기 때문에 해안선이나 장애물로부터 일정거리를 유지하는 것으로 한다.



S:출발지점 G:목적지점
Fig. 11 8방향 탐색에 의한 8개의 후보생성

2.2 A*탐색방법의 적용과 문제점

전자해도상에서 A*탐색방법을 이용하여 목적지를 찾아가는 방법은 Fig.12에서 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.

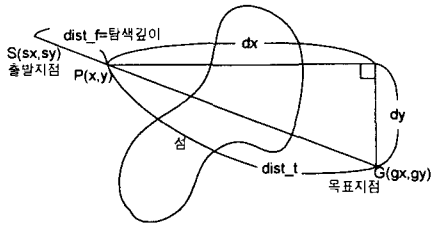


Fig. 12 A*탐색법에 의한 탐색구조

즉, $f(x) = g(x) + h(x)$

$f(x)$: 평가함수

$g(x)$: 탐색깊이(dist_f)

$h(x)$: 목적지까지의 거리(dist_t)

로 하여 후보중 f(x)의 값이 작은 순으로 탐색을 해 나가는 것이다.

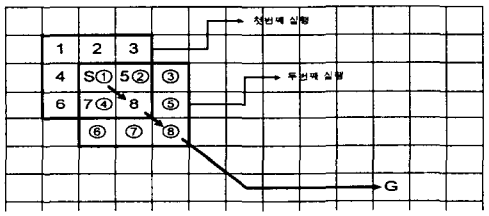


Fig. 13 A*탐색법에 의한 탐색순서

이와 같은 탐색을 Fig. 13에 적용하여 보면 탐색해 나가는 순서는 다음과 같다. S에서 볼 때 먼저 후보리스트에는 1, 2, 3, 4, 5, 6, 7, 8이 선택되지만 이들 중 G에 가장 가까운 점 8이 탐색리스트(bt_stack)에 저장되고 1, 2, 3, 4, 5, 6, 7은 G로부터 거리가 가까운 순으로(5, 7, 3, 2, 6, 4, 1)후보리스트에 저장된다. 일단 8이 최우선 후보로 선정되었으므로 다음 단계로 8에서 후보 ①, ②, ③, ④, ⑤, ⑥, ⑦, ⑧이 후보로 선정되지만 이들중 ①, ②, ④는 후보리스트에 이미 저장되어 있으므로 이들은 제외시킨다. 따라서 다음으로 가장 가까운 점은 ⑧이다. ⑧을 다시 탐색리스트에 저장시킨다. 나머지 ③, ⑤, ⑥, ⑦,은 거리가 가까운 순으로 후보리스트에 저장된다. 요약하면 다음과 같이 된다.

탐색리스트 후보리스트

첫 번째 단계 S 5, 7, 3, 2, 6, 4, 1

두 번째 단계 8, S ⑧,⑤,⑦,③,⑥, 5, 7, 3, 2, 6, 4, 1 세 번째 단계 ⑧, 8, S • • • • • • •

이렇게 해서 목적지를 찾을 경우 다음과 같은 문제점들이 발생하는 것을 알 수 있다.

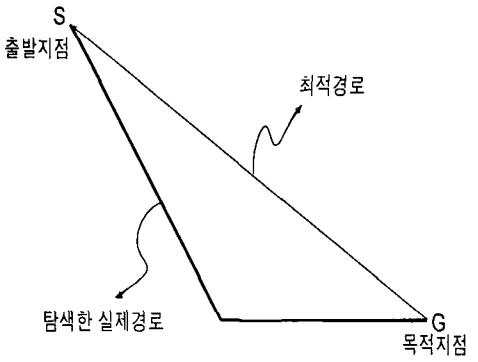


Fig. 14 탐색한 실제경로와 최적경로와의 차이발생

즉, Fig. 14에서도 알 수 있듯이 장애물이 아무 것도 없는 상태에서 출발지점 S와 목적지점 G를 선택하였을 경우 탐색한 실제경로가 최적경로와 상당한 차이가 나는 것을 알 수가 있다. 그러나 이것은 전자해도가 픽셀로 이루어져 있고 후보생성시 인접한 8방향에 대해서만 후보로 인정하기 때문에 발생하는 문제로서 전자해도에 대한 개념을 바꾸지 않는 한 해결될 수 없으므로, 본 논문에서는 고려대상에서 제외하기로 한다.

또한 Fig. 15와 같이 골이 깊은 장애물이 있는 곳에서 탐색시에 문제가 된다. 즉 Fig. 15에서 알 수 있듯이 검은 부분이 모두 후보로 선정되어 메모리의 부족을 초래할 수가 있으며, 탐색시간 또한 많이 걸린다. 이것을 확인하기 위해서 Fig. 16를 예로 들어보면 표1과 같은 순서로 탐색이 진행되어 나가는 것을 알 수 있고 골안에 있는 모든 점들이 탐색후보로 선정되므로 불필요한 탐색을 많이 하게 된다.

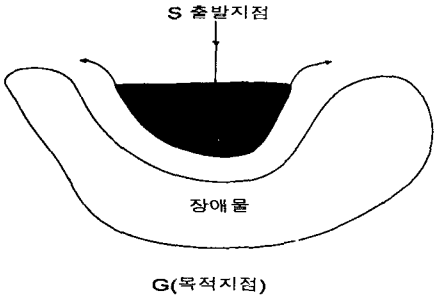


Fig 15 골이 깊은 장애물을 만났을 경우의 후보 생성

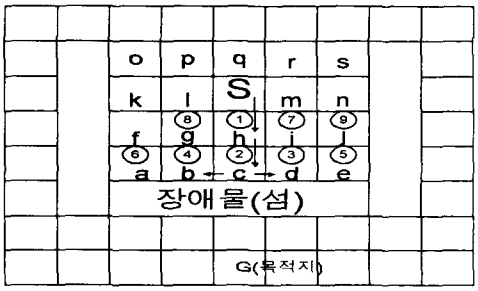


Fig. 16 깊은 골의 장애물을 만났을때의 A*탐색법

진행순서	bt_stack	hubo_list
1	h, s	c, d, b, i, g, m, l, q, r, s
2	c, h, s	d, b, i, g, m, l, q, r, s
3	d, c, h, s	b, e, i, g, j, m, l, q, r, s
4	b, d, c, h, s	e, a, i, g, j, f, m, l, q, r, s
5	e, b, d, c, h, s	a, i, g, j, f, m, l, q, r, s

표 1. Fig. 16의 탐색에 따른 탐색리스트, 후보리스트

그러므로 보다 깊은 골을 만나면 엄청난 후보가 생길 수밖에 없고 메모리 부족 현상을 유발할 수 있다. 따라서 이러한 문제를 해결하기 위하여 A*탐색방법을 개선한 다른 탐색방법이 요구된다. 즉 장애물로부터 일정거리에 도달하면 장애물 둘레를 먼저 탐색하도록 하는 가중치가 부여된 알고리즘의 개발이 필요하다.

Ⅳ. 결론

본 연구에서는 전자 해도상에서 출발지점과 목적지점을 입력하여 최적항로를 결정하기 위해 A*탐색법을 이용한 항로결정알고리즘을 사용하였으나 시간이 많이 걸리고 부분적으로 메모리 부족 현상이 발생하는 등 몇 가지 문제점이 나타났고 이를 바탕으로 다음과 같은 결론을 얻었다.

① 전자해도상에서 각 픽셀을 비용으로 환산하고, 해도 에 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 개선된 탐색 알고리즘의 개발이 필요하게 되었다.

따라서 본 연구에서 발견한 A*탐색법의 문제점을 바탕으로 최적항로를 결정하기 위해 가중치가 고려된 탐색 알고리즘에 대한 연구의 필요성이 나타났고 이에 관한 보다 자세한 검토는 금후 계속 진행 할 계획이다.

참고문헌

[1] David klein and Tim finin, "What's in a deep model?", IJCAI, 1987.

[2] Y.Ishida, "An application of qualitative reasoning to process diagnosis", IEEE, 1988

〔3〕 류길수, "열교환기 온도제어계통에 있어서의 고장진단에 관한 연구", 1990
〔4〕 류길수, "故障診斷システムにおける知識獲得と深い推 論に關する研究", 學位論文, 1989
〔5〕 Donald A. waterman, "A guide to Expert systems", Addison-Wesley Publishing Company, Inc, 1986.
〔6〕 J.Ross Quinlan, "Applications of Expert systems", Addison-wesley Publishing Company,Inc. 1987.
〔7〕 Randall davis, "Diagnostic reasoning based in structure and behavior", Artificial Intelligence, 1983.
〔8〕 Hector geffner and Judea Pearl, "An improved constraint propagation algorithm for diagnosis", IJCAI, 1987.
9) 이상윤, 김재희, "의미네트워크 변환에 의한 생성시스템의 추론 속도향상", 정보과학회논문지 vol.17, p74-p83, 1990.
10) P.H. Winston, 인공지능, 도서출판 生能, 1990.
11) 김재희, 인공지능의 기법과 응용, 교학사, 1991.

저 자 소 개

정정수
한국OA학회 논문지 제2권 제3호 참조
현재 동명정보대학교 정보공학부 컴퓨터공학과 조교수

류길수
1976 한국해양대학 기관학 공학사
1979 한국해양대학원 대학원 제어계측학 공학석사
1986 일본동경공업대학원 정보공학 공학석사
1989 일본동경공업대학원 정보공학 공학박사
현재 한국해양대학교 자동화·정보공학부 컴퓨터공학과 교수