2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

지능형 항해를 위한 응용 폴리선 항로계획

박정선, 김창민, 김용기
경상대학교 컴퓨터과학과
e-mail : kong@ailab.gsnu.ac.kr

Route Planning Applying the Poly-line for Intelligent Navigation

Jeong-Seon Park, Chang-Min Kim, Yong-Gi Kim

요 약

선박 운항시 운항소요인원을 최소화하기 위해서는 선박의 자동화 및 지능화가 필요하다. 특히 선박의 지능화된 운항을 위해서는 항해전문가의 지식을 저장, 관리하여 그것을 기반으로 항해전문가의 임무를 대신 수행하는 지능형 항해시스템(intelligent navigation system)이 요구된다. 지능형 항해시스템은 GPS, 전자해도, 항로계획, 항로감시, 항로관리, 경로계획, 경로감시, 경로관리, 유도, 제어 등의 기능으로 구성된다. 본 논문에서는 지능형 항해시스템의 자동화된 항로계획을 위해 폴리선기능을 응용한 항로계획(RPAP, Route Planning Applying the Poly-line)기법을 제안한다. 제안된 기법은 대표적 항로계획 기법인 A*탐색기법과 비교·평가하여 그 효용성을 증명한다.

1. 서론

최근 선박업계에서는 인력부족 현상을 해결하기 위해 선박의 자동화 및 지능화에 관한 연구가 활발히 진행되고 있다. 선박의 자동화 및 지능화를 위해서는 지능형 항해시스템(*intelligent navigation system*)이 요구되는데, 지능형 항해시스템은 항해전문가의 지식을 저장, 관리하여 그것을 기반으로 항해전문가의 임무를 대신할 수 있는 항해 시스템을 의미한다.

일반적으로 항해시스템은 항해계획(navigation planning)기능과 유도 및 제어(guidance and control) 기능으로 구성된다. 항해계획은 세부적으로 항로계획(route planning)과 경로계획(path planning)으로 구분하는데, 항로계획은 선박이 출발지점부터 목표지점까지 운항하게될 모든 항로를 경유점(waypoint)을 이용하여 계획하는 기능을 의미하고 경로계획은 항로계획시 결정된 경유점 정보를 이용하여 경유점과 경유점 사이의 운항 경로를 계획하는 기능을 의미한다.

선진국을 중심으로 활발히 진행되고 있는 선박의 항해시스템에 관한 연구는 다양한 기능을 보유하고 신뢰도가 높은 ECDIS(Electronic Chart Display Information System)의 개발을 목표로 이루어지고 있다. 현재까지 각국에서 개발된 ECDIS는 GPS, 전자해도, 항로계획, 항로감시, 항로관리, 경로계획, 경로감시, 경로관리, 유도, 제어 기능 등을 보유하는데, 이들 다양한 기능 중 항로계획은 항해전문가의 경유점 입력에만 의존하여 계획되고 시스템은 단지 입력된 정보만을 관리하고 있는 상황이다.

항로계획은 선박의 운항전 또는 선박이 항로를 크게 이탈했을 시에 항해전문가의 오랜 항해지식을 기반하여 최단거리와 안전함을 목표로 이루어진다. 항로계획의 자동화를 위해서는 효과적인 검색기법의 적용이 가장 필요하다. 이를 위해 여러 가지 인공지능적 기법들이 응용되고 있는데, 대표적인 기법으로 격자좌표 기반 A*탐색기법을 이용한 항로계획 기법[1][2]이 있다. 이 기법은 구현이 간단하다는 장점이 있으나 격자의 크기에 따라 원래 물체의 왜곡 정도가 달라져 항로계획 결과가 다르게 나타나는 등의 단점이 있다. 본 논문에서는 전자해도 상에 존재하는 장애물의 모양을 왜곡시키지 않고 최적의 항로를 계획하는 폴리선기능을 응용한 항로계획(RPAP, Route Planning Applying the Poly-line) 기법을 제안한다.

제 2절에서는 격자기반 A*탐색기법을 이용한 항로계획기법에 관해 다룬다. 제 3절에서는 본 연구에서 제안하는 폴리선기능을 응용한 항로계획기법에 관하여 살펴보고 제 4절에서는 A*탐색기법을 이용한 항로계획기법과 제안하는 폴리선기능을 응용한 항로계획기법을 비교 평가하며 마지막으로 제 5절에서는 결론

및 향후 연구과제에 대해 살펴본다.

2. 격자좌표기반 A*탐색기법을 이용한 항로계획 기법

항로계획 연구분야에는 유전자알고리즘[3][4], 퍼지 관계곱[5][6], A* 및 휴리스틱 탐색기법 [1][2][7][8]등 다양한 인공지능적 탐색기법들이 이용되고 있다. 본 절에서는 기존의 여러 탐색기법들 중 최단거리 확보가 증명되어 가장 많이 응용되고 있는 격자좌표 기반 A*탐색기법을 이용한 항로계획 기법에 대해 살펴본다.

A*탐색기법을 이용하여 목표지점을 찾아가는 방법은 기본적으로 그림 1에서 보는 것과 같이 출발지점 S로부터 현재지점 P까지의 거리(*Distance(S to P)*)와 현재지점 P로부터 목표지점 G까지의 거리(*Distance(P to G)*)의 합을 구하고, 그 값이 최소가 되는 지점을 최우선으로 탐색해 가는 방법이다.

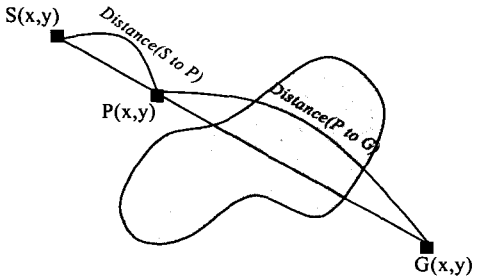

그림 1. A*탐색기법을 이용한 항로계획

격자좌표 기반 A*탐색기법을 이용한 항로계획은 일정 크기의 격자를 기반으로 출발지점부터 목표지점 사이에 후보노드를 생성하고, 그들 중 최적의 평가함수(evaluate function) 값을 갖는 노드를 탐색하여 항로를 계획한다. 이때 평가함수는 선박의 현재방향, 장애물로부터의 거리, 목표지점과의 거리 등 항로탐색시 고려되어야 할 사항을 수치로 계산하는 함수이다.

격자좌표를 기반으로 하는 항로계획에서는 그림 2와 같이 전자해도상의 모든 장애물의 실제 크기나 모양이 격자의 크기에 맞추어 왜곡됨으로 해서 계획된 항로가 설계된 격자의 크기에 의해 영향을 크게 받는다는 문제가 발생한다. 그리고 그림 3과 같이 현재의 노드에서 인접한 8방향의 노드들만을 고려하여 탐색하므로 최적항로를 탐색하지 못하는 문제도 발생한다[2].

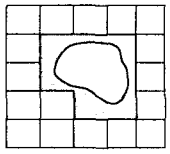

그림 2. 장애물 확장(격자기반)

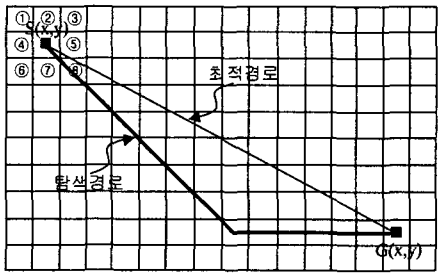

그림 3. 격자좌표기반 A*탐색기법 적용시 항로계획의 문제점

위에서 언급한 문제 이외에도 격자좌표기반 A*탐색기법을 이용하여 항로계획시 격자의 크기가 너무 작게 설정되었거나 골이 깊은 장애물을 만나는 경우 불필요한 노드의 탐색이 많아져 메모리 부족현상이 발생하고 수행시간이 연장된다는 문제가 발생한다[2].

격자좌표기반 A*탐색기법을 이용한 항로계획의 단점을 보완하기 위해서는 오래 경험을 쌓아 숙달된 시스템 설계자에 의해 일반 장애물의 왜곡 정도가 가장 적고 수행시간이 적당한 크기의 격자가 결정되어야 한다. 또한 골이 깊은 장애물을 탐색하게될 경우를 예방하기 위하여 탐색 이전에 장애물의 골을 제거해주는 등의 선처리 작업이 필요하다.

3. 폴리선기능을 응용한 항로계획 기법

본 절에서는 제안하는 폴리선기능을 응용한 항로계획(RPAP, Route Planning Applying the Poly-line) 기법의 전처리단계와 알고리즘에 관해 살펴본다.

3.1 전처리단계

본 논문에서 제안하는 RPAP기법을 적용하여 항로계획을 수행하기 위해서는 세 가지 작업이 먼저 수행되어야 한다.

첫째, 전자해도 상에 벡터형으로 표현된 객체(object)들을 장애물과 비장애물로만 구분한다. 이때 자료구조는 국제수로기구(IHO, International Hydrographic Organization)에서 정의한 전자해도 표준 S-57[9]을 따른다.

둘째, 선박 운항시 안전한 항로를 확보하기 위해 그림 4와 같이 장애물의 크기를 안전거리만큼 확장한다.

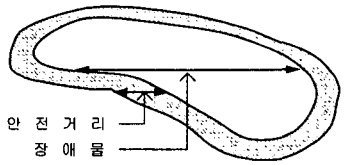

그림 4. 장애물과의 안전거리 확보

셋째, 현재 대상이 되는 전자해도에 대해 화면 외부는 모두 장애물로 인식한다고 가정한다.

3.2 폴리선기능을 응용한 항로계획기법 알고리즘

제안하는 RPAP기법은 그림 5에서 보이는 것과 같이 전자해도 상에 표현된 장애물을 회피하여 항로를 계획하며 아래와 같은 알고리즘을 가진다.

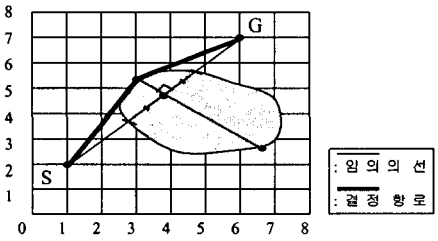

그림 5. RPAP기법에서의 장애물 회피

```
PROGRAM route_planning_applying_ploy-line;
TYPE
  count = 1 .. 1000; {count of obstacles or waypoints}
  g_coordinate = record  {general coordinate}
      x, y : real
    end;
  o_coordinate = record  {obstacle's coordinate}
      x[count], y[count] : real;
   end;
  route_DB_array : packed array [count] of g_coordinate;
  obstacle_array : packed array[count] of o_coordinate;
VAR
  S, G, s, g : g_coordinate;      obstacles : obstacle_array;
  route_DB : route_DB_array;      are_obstacles : boolean;
  i, j, obstacle_dir : integer;  d_c_obstacles:obstacle_array;
FUNCTION are_there_obstacles(s, g : g_coordinate)
                                          : boolean;
  {check on the line l if there is(are) obstacle(s) then
   return true otherwise return false.}
FUNCTION d_c_obstacles(obstacles) : integer;
  {d_c_obstacles=decide_consider_obstacles}
  {decide to consider-obstacles from obstacles' list for
   line l and return the list of d_c_obstacles that is
   sorted by degree of near to s(start point).}
FUNCTION extension(middle : g_coordinate,
                   one_obstacle : o_coordinate) : g_obstacle;
  {extend lines from the obstacle's middle point vertically.
   return the one coordinate have short line's length that
   is chose between the two coordinates that escape from
   the obstacle.}
FUNCTION obstacle_free(s, g : g_coordinate) : g_coordinate;
  {call are_there_obstacles() and return new waypoint that
   return-value from extension(). }
FUNCTION obstacle_free(s, g : g_coordinate)
                                          : g_coordinate;
VAR
o_start, o_end, o_middle, o_position : g_coordinate;

BEGIN
  o_start := one x, y value of d_c_obstacles[1];
  o_end := one x, y value of d_c_obstacles[1];
  o_middle.x := (o_start.x + o_end.x)/2;
  o_middle.y := (o_start.y + o_end.y)/2;
  o_position := extension(o_middle, d_c_obstacle[1]);
  are_obstacles := are_there_obstacles(s, o_position);
  if are_obstacles = 0 then
    obstacle_free := o_position
                     {return the value to call position)
  else
    o_position := obstacle_free(s, o_position);
END

BEGIN
  read(S); read(G);
  i = 1; s := S; g := G;
  route_DB[i] := s;
  while 1
    do begin
      i := i +1;
      are_obstacles := are_there_obstacles(s, g);
      if are_obstacles = 0 then begin
          route_DB[i] := g
          break
        end;
      else begin
          d_c_obstacles := d_c_obstacles(obstacles)
          o_free := obstacle_free(s, g)
          route_DB[i] := o_free
          s := o_free
          g := G;
        end;
    end
  end;
END.
```

4. 평가

본 절에서는 격자좌표기반 A*탐색기법과 본 논문에서 제안하는 RPAP기법을 적용하여 좌표상의 특정 출발지점에서 목표지점까지의 항로를 계획하고 그 결과를 비교하으로써 두 기법의 효용성을 평가하다

우선 두 모형의 비교 평가를 위하여서는 공통 기준이 필요하다. 먼저 RPAP기법을 위한 X, Y좌표를 표현하고, 이와 동일한 크기로 A*탐색기법을 위한 격자를 표현하였다. 이것은 전자해도 상에 위도와 경도로 표현되는 한 점을 기준으로 격자와 좌표를 표현한 것이라고 볼 수 있다. 또한 A*탐색 기법의 격자크기에 따른 결과의 상이함을 평가하기 위해 기준 격자의 크기를 2배, 3배로 확대하고 탐색경로 상에 존재하는 장애물들을 격자의 크기에 따라 표현하였다. 본 평가에서 거리는 모두 RPAP기법의 좌표를 기반으로 계산하였다.

본 평가에서 제안하는 RPAP기법은 전자해도 상에

존재하는 장애물의 모양을 왜곡시키지 않으므로 해서 격자좌표기반 A*탐색기법을 적용한 항로계획보다 뛰어난 운항경비를 가지는 항로를 산출하였다. 또한 격자의 크기에 따라 결과가 다르게 나타나는 A*탐색기법에 반해 항상 일정한 항로를 계획한다는 것을 알 수 있었다. 표 1은 본 논문에서 제안하는 RPAP기법과 격자좌표기반 A*탐색기법을 세 가지 크기의 격자에 적용하여 항로계획한 결과를 보인 것이다. 그림 6과 7은 각각 동일한 크기의 장애물을 보유한 전자해도 상에 출발지점과 목표지점을 부여하여 항로계획한 결과를 표현한 것이다.

	적용 조건 (격자 크기)	총 항로 길이 합	경유점 수
항로 #1	RPAP	13.06	4
	A* (기준격자)	24.30	7
	A* (기준격자의 2배)	25.89	6
	A* (기준격자의 3배)	24.65	7
항로 #2	RPAP	20.33	6
	A* (기준격자)	22.31	8
	A* (기준격자의 2배)	23.06	8
	A* (기준격자의 3배)	23.79	8
비고		(소수점 두자리 반올림)	

표 3. 항로계획 결과 비교 예

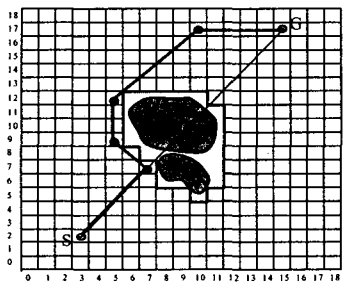

그림 6. 항로계획 결과(A*탐색기법 적용)

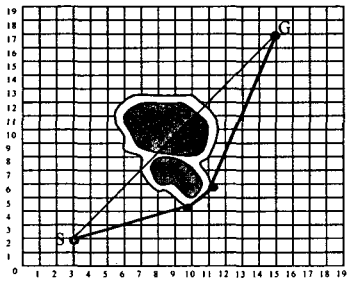

그림 7. 항로계획결과 (RPAP기법 적용)

5. 결론 및 향후과제

본 논문에서 제안하는 RPAP기법을 적용한 항로계획은 기존에 널리 쓰이는 격자기반 A*탐색기법을 적용한 항로계획 보다 뛰어난 운항경비의 항로를 산출하였고 또한 그 결과가 항상 일정하게 나타나는 장점을 가진다.

본 연구에 이어 향후에 이루어져야 할 과제는 다음과 같다. 첫째, 실제 전자해도 상에 본 기법을 적용하여 보다 신뢰성 높은 결과를 산출하도록 유도하여야 할 것이다. 둘째, 전자해도 상의 장애물인식에 대한 다양한 연구가 이루어져야 할 것이며 셋째, 계획된 항로의 데이터베이스를 효율적으로 관리하여 항해시스템 내의 타 기능에 효율적으로 이용되도록 하여야 할 것이다.

참 고 문 헌

[1] Robert J. Szczerba, Robust Algorithm for Real-Time Route Planning, IEEE Transcations on Aerospace and Electronic Systems Vol. 36, No. 3 July 2000.

[2] 하희천, 전자해도를 이용한 최적항로결정시스템에 관한 연구, 한국해양대학교, 1997, 2.

[3] Andreas C. Nearchou, Adaptive Navigation of Autonomous Vehicles Using Evolutionary Algorithms, Artificial Intelligence in Engineering 13, 1999.

[4] Sunshil J. Louis, Multiple Vehicle Routing With Time Windows Using Genetic Algorithms, IEEE, 1999.

[5] 민종수, 김창민, 김용기 자율무인잠수정의 휴리스틱 경로탐색을 위한 자세 기반 상대적 격자모형, 한국정보처리학회 춘계 논문집, 1999

[6] 이영일, 김창민, 김용기, 퍼지관계곱 기반 평가함수를 이용한 개선된 AUV의 항행 휴리스틱 탐색기법, 한국정보처리학회 춘계 논문집, 1999

[7] Laban Kallgren, Real- time replanning of mission routes based upon threates, Master thesis, Linkoping University, 5th of January 2001.

[8] Seow Meng Ong, A Mission Planning Expert System with Three-Dimensional Path Optimization for the NPS Model 2 Autonomous Underwater Vehicle, Naval Postgraduate School, Monterey, California June 1990.

[9] S-57 Maintenance document, IHO, 2000.