'93 한국자동제어학술회의논문집 (1993.10.20~22)

이동로보트의 경로계획

°표종훈, 한민홍
고려대학교 공과대학 산업공학과

Path Planning for Modile Robot Navigation

°Jong-Hoon Pyo, Min-Hong Han
Department of Industrial Engineering, Korea University

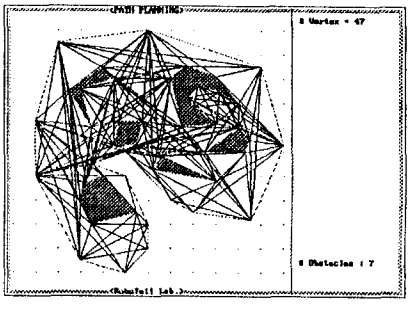
ABSTRACT

This paper discusses an approach to real-time path-planning of mobile robot navigating amidst multiple obstacles. Given an environment with the coordinates of known obstacles, the moving area of a mobile robot is divided into many patches of triangles with small edge length, in order to ensure a path better than those reported in the literature. After finding a minimum-distance path between start and goal point, the path is revised to minimize the number of turns and total path-length by two-step path-revision and path-smoothing.

1. 연구의 소개

이동로보트의 기본적인 기능은 주어진 목적지까지 빠른 시간안에 안전하게 도착하는 것이다. 이를 위해서는 이동로보트 스스로가 장애물과 충돌하지 않고 빠른 시간안에 목적지에 도착할 수 있는 최적경로를 실시간으로 찾을 수 있어야 한다. 위치좌표가 이미 알려진 고정된 장애물들을 고려할 때, 최적경로를 얻는 방법은 장애물의 꼭지점을 노드로, 꼭지점들을 서로 연결하는 선분중에서 장애물과 교차하지 않는 것을 아크로 간주하여 네트워크를 구성한 후, Dynamic Programming이나 Dijkstra Algorithm을 이용하여 네트워크로부터 경로를 찾는 것이다. 그러나 꼭지점의 갯수가 많은 경우에는 네트워크의 크기가 커지게 되어 해를 얻기까지 많은 시간이 소비되므로 실시간경로계획은 사실상 어렵다. <그림 1>은 7개의 장애물과 47개의 꼭지점을 갖는 경로계획문제에서 최적경로를 얻기 위한 네트워크이다. N개의 노드가 있을 때 하나의 노드에 연결될 수 있는 아크의 최대 갯수는 (N-1)개이므로 노드의 갯수가 증가하면 실시간처리는 불가능하다.

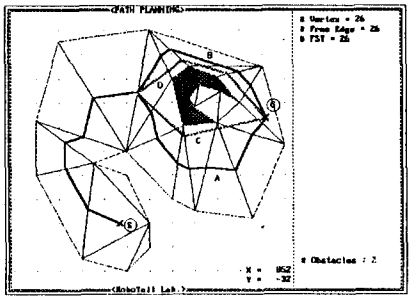
최적경로는 아니더라도 그에 근접한 좋은 경로를 실시간으로 얻기 위해서는 새로운 구조로 네트워크를 구성해야 한다. 삼각분할(Triangulation)은 장애물구역을 제외한 자유구역을 여러개의 삼각형(FST : Free Space Triangle)으로 분할할 수 있다. 그리고 분할된 삼각형들을 노드로 간주하는 네트워크를 구성하면 하나의 노드에 연결된 최대 아크의 수가 3개로 제한되므로, 노드의 갯수가 증가하더라도 빠른 시간안에 해를 얻을 수 있다. 구역분할[1][2][6][7]에 이용되는 삼각분할법에는 몇가지가 있지만 가장 기본이 되는 것은 Voronoi diagram의 쌍대원리(Duality Relation)에 기초한 Delaunay Triangulation[4]이다. 하지만 이 방법은 기하학으로부터 출발한 개념이라서 이해하기가 쉽지 않고 알고리즘으로의 구현이 까다로우며 분할속도가 느리다. Kai Wing Tang과 Ray Jarvis[1]가 개발한 삼각분할법은 분할속도는 빠르지만 폭이 좁고 길이가 긴 삼각형을 형성하므로 주행경로가 길어지는 단점이 있다. 왜냐하면 동일한 운행구역이 주어질 때 가능한 한 길이가 짧은 변을 갖는 삼각형으로 분할하는 것이 최적경로에 근접할 수 있는 기본이기 때문이다. 변의 길이가 짧은 삼각형을 만들수록 전체 운행구역은 더욱 많은 수의 삼각형들로 분할된다. 노드의 갯수가 증가함에 따라 네트워크의 탐색에 소비되는 시간도 증가하지만 그에 연결된 아



<그림 1> 최적경로를 얻기위한 네트워크

크의 갯수가 적기 때문에 그 시간의 증분은 주행거리의 단축효과에 비교해 볼 때 무시될 수 있을만큼 적다.

이동로보트의 경로계획에 관한 대부분의 연구들에서는 분할된 삼각형을 이루는 변의 중점을 따라 경로가 진행한다[1][3]. 그 이유는 첫째, 이동로보트가 주행중에 주위의 장애물들로부터 먼 거리를 유지할 수 있고 둘째, 분할된 삼각형을 노드로 구성하는 네트워크(FSTN : Free Space Triangle Network)로부터 짧은 시간안에 경로를 찾을 수 있기 때문이다. 하지만 이러한 경로정의는 긴 주행거리를 갖는 경로를 만들게 되며, 선택된 경로는 단지 네트워크상에서만 최적일 뿐 현실적인 최적경로와는 많은 차이가 있다. 따라서 경로상의 방향전환점에 대하여 Visibility Test[6] - 두 점을 연결한 선분이 장애물과 교차하는가를 검사하는 것 - 를 이용한 경로개선을 실시해 보면, 경로개선을 실시한 이후에 그 결과가 뒤바뀌는 경우가 생긴다. 예를 들어, 운행구역내에 장애물이 존재하는 <그림 2>와 같은 경우를 생각해 보자. 경로A와 경로B는 네트워크상의 경로이고 경로C와 경로D는 각각 경로A와 경로C를 개선시킨 경로이다. Length(A) > Length(B)이므로 경로B가 선택되지만 경로개선을 실시해 보면 Length(C) < Length(D)이므로 결국 경로B를 선택한 이동로보트는 경로C보다 더욱 거리가 긴 경로D를 따라 주행하는 오류를 범한다. 또한, 경로개선을 실시한 후 그 선택결과가 바뀌지 않았더라도 <그림 2>의 네가지 경로는 모두 좋은 경로라고 볼 수 없다. 결론적으로, 최적경로에 근접한 좋은 경로를 얻기 위해서는 삼각형의 변을 따라 진행하는 경로구축방법을 활용하여야 된다.



<그림 2> 변의 중점을 통과하는 경로

경로개선을 실시하는 이유는 선택된 경로가 현실적인 최적경로를 만들 수 없는 불완전한 네트워크로부터 찾아진 것이기 때문에 경로를 구성하는 방향전환점중에는 경로에서 삭제되더라도 장애물회피에 전혀 문제가 되지 않는 점이 존재하기 때문이다. 경로개선은 경로상의 모든 방향전환점들에 대하여 Visibility Test를 실시한 후 그 결과에 의하여 삭제가능한 점들을 경로상에서 제거함으로써 이루어진다.

본 연구는 앞에서 언급되었던 기존의 연구들에서 나타난 문제점들을 보완하여 최적경로에 근접한 좋은 경로를 실시간으로 얻기 위하여 다음과 같은 사항들을 다루고자 한다.

(1) 주어진 운행구역을 가장 길이가 짧은 변을 갖는 삼각형들로 분할하는 방법을 소개하고,
(2) 경로개선후에 최적경로에 근접한 경로를 얻을 수 있도록 분할된 삼각형의 변을 따라 경로를 진행시키고,
(3) 경로개선을 실시하여 경로를 단축하는 동시에, 이동로보트의 유연한 방향전환과 주행시간단축을 위하여 경로를 Smoothing 하는 방법을 소개한다.

데이타입력과 분할 Edge 추출 및 네트워크 구축의 세과정은 장애물위치의 변화가 없다면 이동로보트의 운행전에 단 한번만 수행된다. 역상으로 표시된 경로의 선택과 개선은 새로운 목적지가 주어질 때마다 수행된다. 실제로 본 알고리즘을 실행시키면 삼각분할과정에서 소비되는 시간이 총실행시간의 70%이상을 차지하며, 만일 장애물의 꼭지점 갯수가 증가하면 그 비율은 더욱 커질 것이다. 그러므로 본 알고리즘은 장애물위치의 변화가 빈번하지 않는 한 실시간으로 주행경로를 계획할 수 있다.

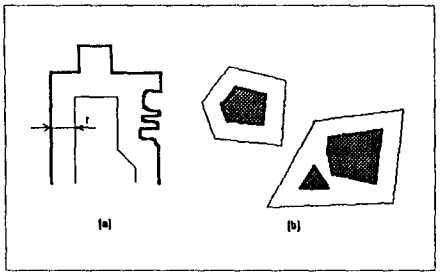
2. 분할 Edge의 추출

장애물좌표를 입력할 때는 모든 실제좌표값에 (이동로보트의 반경+α)를 더한다.(단, α는 안전계수) 안전계수의 값이 클수록 주행하는 로보트와 장애물과의 간격이 커지게 되어 안전주행을 할 수 있지만 실제운행구역이 좁아지므로 경로가 길어지는 결과를 가져온다.

● 좌표 입력

운행구역의 외곽좌표는 외곽선을 이루는 꼭지점의 실제좌표값에 <그림 3>의 (a)와 같이 운행구역의 안쪽으로 (로보트의 반경+α)를 더한 좌표로 입력한다. 로보트가 진입할 수 없는 좁은 구역은 제거하고 외곽선의 요철이 심한 곳은 단순화시킴으로써 꼭지점의 증가로 인한 분할 Edge 추출에 소비되는 시간을 절약할 수 있다.

장애물좌표는 장애물을 이루는 꼭지점의 실제좌표값에 장애물의 바깥쪽으로 (로보트의 반경+α)를 더한 좌표를 입



<그림 3> 좌표 입력

력한다. 근접한 장애물사이의 간격이 로보트의 지름보다 좁은 경우에는 <그림 3>의 (b)와 같이 두개의 장애물을 하나의 장애물로 간주하여 좌표를 입력한다.

● 기본데이타의 종류 및 구조

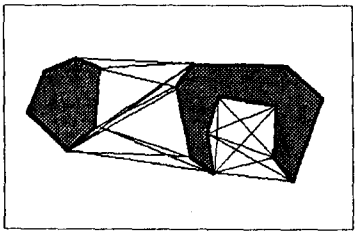
꼭지점, 변, 삼각형이 기본데이타를 형성한다. 장애물이나 꼭지점의 수가 달라지더라도 유연성있게 대처하기 위하여 입력되는 데이타는 동적메모리할당기법[8]을 이용하여 저장된다.

각 장애물의 꼭지점 입력순서는 반시계방향이어야 한다[5]. 만일 반시계방향이 아니라면, 꼭지점의 좌표로부터 삼각분할에 필요한 정보를 얻을 수 없기 때문이다.

연결가능한 모든 꼭지점들 사이에 만들어지는 선분중에서 장애물의 외곽선을 이루는 변을 S_EG(Solid Edge), 그 나머지 선분들을 NS_EG(Non-Solid Edge)로 나타내면 NS_EG중에서 삼각형을 형성하는 변이 분할 Edge가 된다. 꼭지점, 변, 삼각형간의 관계는 Explicit Edges Mesh구조[5]로 연결된다.

● 분할 Edge의 추출

NS_EG는 <그림 4>와 같이 일반적으로 두개의 서로 다른 장애물들 사이에서 만들어지는데 장애물이 오목한 모양일 때는 하나의 장애물에서도 만들어진다.



<그림 4> Non-Solid Edge(NS_EG)

NS_EG는 복잡하게 서로를 가로지른다. 가장 길이가 짧은 분할 Edge들의 집합을 만들어야만 크기가 작은 삼각형이 보다 많이 만들어지게 된다. NS_EG중에서 길이가 가장 짧은 것부터 FST의 변이 될 수 있는 후보로 채택하기 위하여 NS_EG를 크기순으로 정렬시킨다. 정렬에는 Bubble Sort Algorithm을 사용하였다.

분할 Edge를 만드는 과정은 길이가 가능한 한 짧고 서로 교차하지 않으면서도 삼각형을 이루는 변의 집합을 찾는 것이다. 첫번째의 NS_EG는 가장 짧기 때문에 처음 실시하는 교차테스트의 비교기준이 되므로 무조건 분할 Edge가 된다. 차례로 교차테스트를 받는 특정한 NS_EG는 자신보다 앞서 만들어지는 모든 분할 Edge의 어느 하나와도 교차하지 않으면 그 자신이 분할 Edge가 되어 자신의 뒤에 오는 NS_EG에 대하여 교차테스트의 비교기준이 된다. 이 과정에서 만들어지는 분할 Edge는 반드시 FST의 변으로 사용된다. 두 변이 서로 교차하는가를 테스트하는 방법은 [부록2]를 참고하시오.

3. Edge 네트워크 구성

S_EG와 분할 Edge를 합해서 A_EG(Active Edge)라 하면 경로탐색은 A_EG를 따라서 진행한다. FST가 필요한 이유는 출발지와 목적지가 입력되면, 이들 점들이 각각 어느 삼각형내에 위치하는가를 알아야 하기 때문이다. 그 삼각형들을 찾으면 그의 꼭지점들과 출발지, 목적지를 연결하여 새로운 A_EG를 만든 후 A_EG리스트에 추가한다.

A_EG = 분할 Edge + S_EG = FST의 변

● 운행구역의 삼각분할

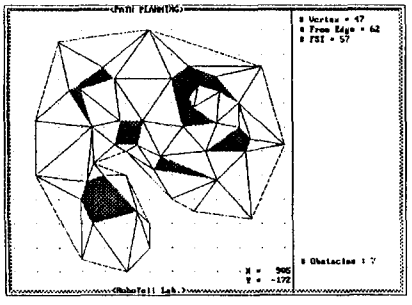
A_EG리스트중에서 길이가 짧은 분할 Edge를 우선순위로 하여 그 변의 두 꼭지점과 동일한 꼭지점을 갖는 두개의 서로 다른 변을 찾아서 두변의 나머지 꼭지점들이 일치하면 삼각형을 이룬다고 결정하고 FST리스트에 추가한다. 그리고 만일 동일한 삼각형이 이미 FST리스트에 존재하면 현재의 삼각형을 무시하고 다음 변을 고려한다. 모든 분할 Edge들을 한번씩 고려한 후에는 FST 링크드리스트가 만들어지고, 운행구역은 이들 삼각형들로 분할된다.

● 네트워크 구축

한 변은 두개의 끝점을 갖는다. 입력된 모든 꼭지점은 네트워크의 노드로, A_EG는 각 노드를 연결하는 아크가 된다. FST리스트가 만들어진 후, 출발지를 포함하는 삼각형 T_S와 목적지를 포함하는 삼각형 T_O를 찾는다. 특정한 점이 한 삼각형내에 위치하는지를 결정하기 위하여는 [부록3]을

참조하시오. 출발지를 S, 목적지를 G, 삼각형 T_S의 세 점을 각각 V_{S1}, V_{S2}, V_{S3}라 하고, 삼각형 T_G의 세 점을 각각 V_{G1}, V_{G2}, V_{G3}라 하면, 변 E(S, V_{S1}), E(S, V_{S2}), E(S, V_{S3}), E(G, V_{G1}), E(G, V_{G2}), E(G, V_{G3})를 만들어 A_EG리스트에 추가한다. 이로써 네트워크의 노드는 2개 증가하고 아크는 6개 증가한다.

<그림 5>은 <그림 1>에서 다룬 문제를 간략화한 네트워크이다. 삼각분할한 결과 분할 Edge는 62개가 추출되었고, 운행구역은 57개의 삼각형으로 분할되었다. <그림 1>에 비교하여 볼 때 노드를 연결하는 아크의 갯수가 대폭 감소한 것을 볼 수 있다. 우리는 <그림 1>대신에 <그림 5>의 네트워크를 이용하여 경로를 구한다.



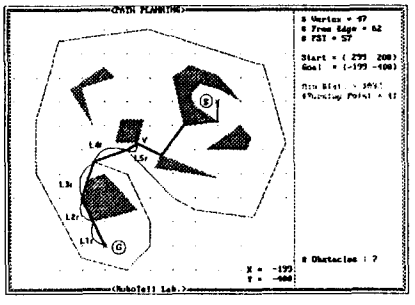
<그림 5> 간략화된 네트워크

4. 경로의 선택 및 개선

Branch & Bound Method[9]를 이용하여 <그림 5>의 네트워크로부터 경로를 찾는다. 선택된 경로는 현실적인 최적경로는 아니므로 경로개선을 실시한다.

● 경로의 선택

경로의 탐색은 목적지로부터 이루어진다. 출발지로부터 탐색을 시작하여도 동일한 결과를 얻는다.



<그림 6> 선택된 경로

<그림 6>에서 꼭지점 V의 거리는 목적지로부터 V에 이르는 경로상에 있는 A_EG들의 길이를 합한 (L1r + L2r + L3r + L4r + L5r)이다. 꼭지점 V까지 목적지로부터 경로 r을 따라 P_r개의 A_EG를 거쳐서 도착했다면 거리는 다음과 같다. V의 거리는 경로r에 따라 달라진다.

$$V\text{의 거리}(D_w) = \sum_{i=1}^{P_r} L_{ir}$$

* L_{ir} ≡ 경로 r상에서 목적지로부터 i번째 A_EG의 길이

<그림 6>은 <그림 5>의 네트워크에서 경로를 탐색한 결과를 나타낸다. 출발지의 좌표는 (299, 200), 목적지의 좌표는 (-199, -400)이다. 굵은 실선으로 표시된 것이 선택된 경로이며 경로거리는 10.92m이고 방향전환점의 수는 11개이다. 경로는 A_EG를 따라 진행하고 있다. 하나의 A_EG를 지날 때마다 방향전환점의 수가 하나씩 증가한다. 이동로보트가 주행중에 방향전환점에서 정지한 후 새로운 방향으로 회전해야 한다면 방향전환점의 수가 많아질수록 로보트의 평균주행속도는 낮아진다. 만일 경로상에서 방향전환점의 수를 줄일 수 있다면, 주행거리와 주행시간을 단축할 수 있을 것이다.

● 경로의 개선

i) 1차 경로개선 - 불필요한 방향전환점의 제거

<그림 6>에 나타난 경로를 자세히 살펴보자. 어떠한 방향전환점은 그 점의 앞뒤에 위치하는 방향전환점들을 연결함으로써 경로상에서 삭제될 수 있다. 삭제후에도 장애물 회피에 전혀 문제가 없을 뿐 아니라 오히려 경로를 단축하는 효과를 가져다 준다.

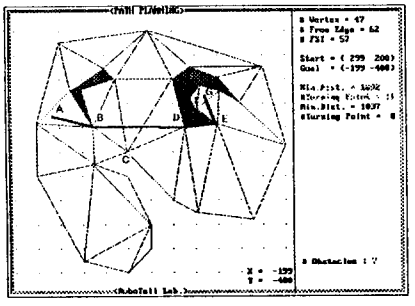
1차 경로개선을 실시하는 순서는 다음과 같다. (단, 경로개선은 출발지로부터 목적지로 진행하면서 실시한다. 역방향으로 실시하여도 동일한 결과를 얻는다.)

(가) 경로상의 모든 방향전환점들의 쌍에 대하여 Visibility Test를 실시하여 경로를 가장 크게 단축할 수 있는 쌍을 찾는다.

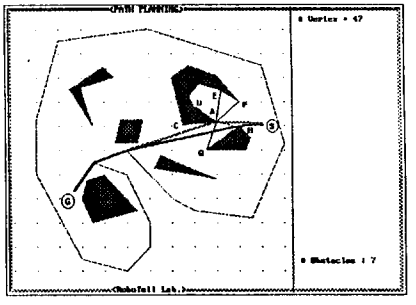
(나) (가)에서 찾아진 쌍을 실제로 연결하여 새로운 경로를 만든다.

(다) 더 이상의 경로단축이 없을 때까지 (가)(나)를 반복한다.

<그림 7>은 1차 경로개선을 실시하는 예이다. 네트워크상에서 선택된 경로 A-B-C-D-E-F-G가 있을 때, Visibility Test를 실시한 결과 점 B와 D를 연결하는 것이 가장 큰 경



〈그림 7〉 1차 경로개선의 예



〈그림 8〉 2차 경로개선의 예

로단축을 가져오므로 그들을 최우선순위로 연결하여 새로운 경로를 만든다. 새로운 경로에 대하여 다시 Visibility Test를 실시하여 점 E와 G를 연결하면 더 이상 단축가능한 구간이 없으므로 종료한다. 개선된 경로는 굵은 실선으로 표시된 A-B-D-E-G 이다.

(ii) 2차 경로개선 - 방향전환점의 이동

1차 경로개선이 끝난 후 경로상에 남아 있는 방향전환점들은 이동로보트의 장애물회피를 위하여 반드시 필요한 점들이다. 이들 방향전환점들 중에는 〈그림 8〉의 점 A와 같이 직접적인 방향전환의 책임이 없는 점들도 있다. 〈그림 8〉에서 더 좋은 경로가 만들어지려면 점 A가 방향전환의 책임을 점 B로 전가함으로써 점 A는 경로로부터 탈출하고 점 B는 경로안으로 들어가야 한다.

2차 경로개선을 실시하는 순서는 다음과 같다.

(가) 경로상에 있는 연속된 세 방향전환점을 V1, V2, V3라 하였을 때, V2와 변으로 연결된 점들 중에서 V2 대신 경로에 들어 갔을 때 가장 큰 경로의 단축을 가져올 수 있으며, 장애물회피 에도 문제가 없는 점을 선택한다.

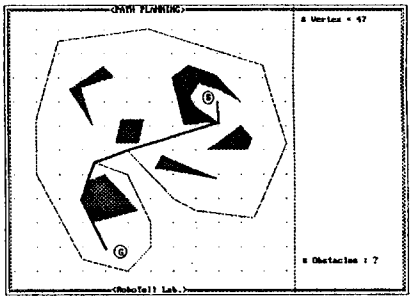
(나) (가)에서 선택된 점으로 V2를 대체함으로써 새로운 경로를 만든다.

(다) 경로상에 있는 모든 연속된 세점에 대하여 (가)(나)를 반복한다.

〈그림 8〉는 2차 경로개선을 실시하는 예이다. 점 A와 변으로 연결된 점 B, C, D, E, F, G를 점 A 대신에 하나씩 경로에 넣어 만들어지는 새로운 경로를 고려한다. 점 B를 선택하면 경로의 길이도 짧아지고 점 B를 통과하는 새로운 경로가 장애물과 교차하지도 않으므로 점 B로서 점 A를 대체한다. 굵은 실선으로 표시된 것이 개선된 경로이다.

〈그림 9〉은 〈그림 6〉에 대해서 1차 경로개선을 실시한 결과를 나타낸다. 1차 경로개선만으로 좋은 경로가 만들어졌으므로 2차 경로개선을 실시하여도 같은 결과를 얻는다. 주행거리는 10.92m에서 10.37m로 0.55m가 단축되었고, 방향전환점의 수는 11개에서 8개로 3개가 감소되었다.

경로개선으로 얻을 수 있는 주행시간의 단축은 이동로보트의 주행메카니즘에 따라 조금씩은 다를 수 있겠으나 개선전에 비해 훨씬 좋은 결과를 가져다 준다.



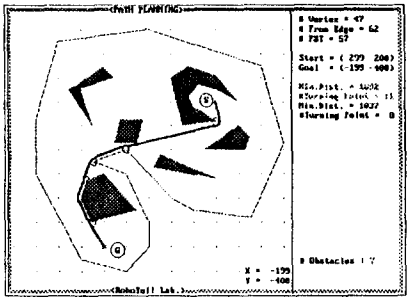
〈그림 9〉 〈그림 9〉에 대한 경로개선 결과

〈그림 9〉에 나타난 경로를 살펴보면 경로의 많은 부분이 장애물을 이루는 S_EG로 이루어져 있음을 알 수 있다. 이동로보트가 경로를 따라 이 구역을 주행할 때에는 장애물과 매우 가까이 접근하게 된다. 이동로보트가 경로상의 좌표를 따라 한치의 오차도 없이 주행한다면 로보트와 장애물 사이에는 적어도 안전계수 α만큼의 간격이 유지된다. 그러나, 실제로 바퀴의 미끄러짐등에 의한 주행오차가 누적되면 장애물과의 충돌위험이 있으므로 이동로보트가 장애물에 가까이 접근하는 것은 바람직하지 않다. 또, 로보트가 방향전환점에서 회전할 때 방향전환점을 중심으로 반지름 R을 갖는 원의 호를 따라 회전한다면 정지하지 않고서도 부드럽게 방향을 바꿀 수 있으므로 주행시간을 단축할 수 있다.

● 경로의 Smoothing

경로의 Smoothing과정은 방향전환 호(Arc)를 연결하는 것이다. 방향전환 호의 반지름이 클수록 원활한 방향전환이 가능하지만 중심에서 멀어지기 때문에 주위의 장애물과 충돌할 가능성이 높아지므로 유의해야 한다. 각 방향전환점에서 만들어지는 호의 반지름을 R로 고정하면 호의 연결이 쉬워진다.

<그림 10>는 <그림 9>의 경로를 smoothing하여 만든 최종경로이다.



<그림 10> Smoothing을 마친 최종경로

5. 결론

이동로보트의 경로계획문제에서 최적경로를 실시간으로 얻기는 어렵다. 그러므로 실시간처리를 위해서는 원문제를 구역분할을 통하여 빠른 시간내에 풀 수 있는 문제로 변형한 후, 변형문제의 해를 구한다. 여기서 얻어진 경로는 원문제에서의 최적경로와 많은 차이가 있으므로 경로개선을 실시하여 최적경로에 근접한 경로를 얻는다. 실제로 출발지와 목적지를 바꾸어가며 실험을 반복한 결과 90% 이상의 경우에 최적경로를 얻었으며 나머지 경우는 이에 근접한 경로를 나타냈다.

최적경로에 근접한 좋은 경로를 얻기 위한 조건은 운행구역을 가능한 한 작은 삼각형들로 분할하고, 분할된 삼각형의 변을 따라서 경로를 진행시키는 것이다. 경로개선을 실시하면 경로거리는 평균 6% 단축이 가능하고, 방향전환점의 갯수는 평균 31% 줄어든다.

Norton Benchmark Test에서 CPU속도 72.1MHz를 나타내는 80486 PC를 사용하여 본 알고리즘을 실행시키면 <그림 1>에서 다룬 경로계획문제에서 <그림 10>의 최종경로를 2-3초내에 얻을 수 있다. 하지만 이동로보트의 실제운행전에 수행되는 분할 Edge의 추출에서 소비되는 시간을 감하면 새로운 목적지가 주어질 때마다 경로를 찾는 시간은 1초이내이다.

<참고 문헌>

[1] Kai Wing Tang and Ray Jarvis, "Collision-Free Path Finding Amongst Polygon Obstacles Using Efficient Free Space Triangulation", Second International Conference on Automation, Robotics, and Computer Vision, Vol.3 of 3, RO-11.1.

[2] F. Wallner, T. C. Lueth and F. Langinieux, "Fast Local Path Planning for a Mobile Robot", Second International Conference on Automation, Robotics, and Computer Vision, Vol.3 of 3, RO-10.5.

[3] W. S. Ko, L. D. Seneviratne, S. W. E. Earles, and W. C. Ko, "Implementation of Robot Collision-Free Path Planning in a Maze Using the Triangulation Algorithm", Second International Conference on Automation, Robotics, and Computer Vision, Vol.3 of 3, RO-10.7.

[4] Leonidas Guibas and Jorge Stolfi, "Primitives for the Manipulation of General Subdivision and the Computation of Voronoi Diagrams", ACM Transactions on Graphics, Vol.4, No.2, April 1985, Pages 74-123.

[5] 신 영수, 김 현석, "3차원 그래픽 - C언어로 배우는 이론과 알고리즘", 가남사.

[6] T. C. Woo and S. Y. Shin, "A Linear Time Algorithm for Triangulating a Point-Visible Polygon", ACM Transactions on Graphics, Vol.4, No.1, January 1985, Pages 60-70.

[7] Hakyoung Chung, Yong Seok Choi, Jang Gyu Lee, "Path Planning for a Mobile Robot with Grid Type World Model", Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems, Pages 439-444.

[8] Christopher J.Van Wyk, "Data Structures and C Programs", Addison Wesley.

[9] Billy E.Gillett, "Introduction to Operations Research - A Computer-Oriented Algorithmic Approach", McGraw-Hill.