碩士學位論文

방향성 및 동적정보를 이용한 능동적 CNS에 대한 연구

濟州大學敎 大學院

情報工學科

李 宗 憲

2001 年 12 月

방향성 및 동적정보를 이용한 능동적 CNS에 대한 연구

指導敎授 李尙俊

李 宗 憲

이 論文을 工學 碩士學位 論文으로 提出함

2001 年 12 月

李宗憲의 工學碩士學位 論文을 確認함

審査委員長 곽 호 영 印

委 員 이 상 준 印

委 員 변 상 용 印

濟州大學校 大學院

2001 年 12 月

A study on the self-adjusting CNS using the direction and dynamic information

Jong-Heon Lee

(Supervised by professor SangJoon Lee)

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF INFORMATION ENGINEERING

GRADUATE SCHOOL

CHEJU NATIONAL UNIVERSITY

2001. 12

<목차>

Summary

The path routing algorithm is the most important part in the Car Navigation System.
But most of the algorithms be designed using static information , so those algorithms don't apply the dynamic environment informations which is changed dynamically in every situation such as the weather , the traffic accidents.

In this study , we proposed the new method that applies the dynamic informations which are transmitted via the wireless communication network so that the CNS can keep the most suitable path in real time.

We simulated the proposed algorithm on the digital map which contains a two-lane and over in Jeju city.

As the result of the simulation, we got the effective result within acceptable time.

I. 서론

지속적인 도로망 확충에도 대도시의 자동차 정체현상은 날로 심화되고 있는 추세이다. 이러한 교통문제를 해결하기 위한 여러 가지 해결방법이 절실히 요구되고 있다. 자동차들의 도로 병목현상을 최소화하고, 자동차를 효율적으로 운행시켜 도로이용률을 최적화하는 기술이 필요하다.

오늘날 컴퓨터통신기술의 발전으로 고성능 장비를 저렴한 가격으로 활용할 수 있게 되고, 그 동안 산발적으로 진행됐던 디지털 지도제작, CAD시스템, 자동운송 위치시스템(AVLS), 차단관리(Fleet Management), 차량항법 시스템(CNS), 위치측정 시스템(GPS) 등을 통합한 지능형 교통 시스템(ITS)이라는 새로운 분야가 등장하게 되었다(김기섭, 1999. 전홍석, 1998). 그 중에서도 차량항법 시스템이라고 불리는 CNS(Car Navigation System)는 ITS분야 중 가장 먼저 상용화 단계에 있는 분야이다(전홍석, 1998).

이 시스템은 차량내 단말기에 전자수치지도, GPS, 항법 소프트웨어 등을 혼합한 것으로 전자지도상에 운전자의 현재 위치를 표시해 준다. 또한 최단경로, 최적경로 등의 정보 등을 제공해줌으로써 운전자가 목적지까지 편하게 갈 수 있도록 안내도 해 줄 수 있다. 여기에 통신 기술의 발전은 GPS 등의 무선통신 기술을 통해 실시간으로 이동중인 차량에 교통정보까지도 제공해 줄 수 있게 되었다. 이러한 기술은 운전자의 편리함을 제공할 뿐 아니라 더 나아가 도로교통문제의 해결 및 효율성을 향상에 도움이 될 수 있을 것이다.

차량경로계획 서비스는 차량항법 시스템과 연계하여 운전자에게 목적지까지 빠르고 쾌적한 주행경로를 제공함으로써 도로망을 효율적으로 사용한다는 측면에서 매우 중요하다. 이와 같은 시스템은 자동차 운전자뿐만이 아니라 경찰, 119, 방법순찰 등의 이동과 콜택시, 화물운송차량 이동 등을 관리하는 관제시스템에도 요구된다(남상우, 박문성, 1997).

이런 차량경로계획을 수행하기 위한 알고리즘은 가장 중요한 것이 경로탐색

(Routing)이다(김기섭, 1999. 전홍석, 1998). 최적경로는 무선 통신망을 이용해 실시간으로 시시각각 변하는 교통상황 정보를 전달받아 경로선택에 적용하는 방법이다. 갑작스런 날씨 변화에 의한 교통체증, 교통 사고에 의한 교통 통제, 이외 돌발 사고등 운전자가 미리 예측할 수 없었던 여러 가지 사고들은 교통체증 및 사고를 유발하기 쉽다. 따라서 이러한 실시간 기상정보, 도로상태, 도로정보, 사고 정보 등을 실시간을 수신하여 경로 선택에 적용함으로써 최적 경로를 찾아낼 수 있게 되는 것이다. 최적 경로를 이용한 차량항법시스템(CNS: Car Navigation System)을 이용할 경우 운전자에게 최단 시간 내에 목적지까지 도달할 수 있는 최적 경로를 제공해 주게 되어, 운전자에게는 편한 운행이 되고 도로 교통 체증 문제 및 사고도 원활히 해결해 나갈 수 있게 될 것이다(전홍석, 1998).

경로탐색은 운전자에게 현위치에서 최종목적지 까지 가장 빠르고 최적인 도로를 통하여 도달할 수 있는 경로를 알려주는 것이다.

대단히 규모가 크고 복잡한 조건을 갖는 도로망이 대상이므로 기존의 최단경로 알고리즘을 적용하는 것은 여러 문제점을 발생시킨다. 먼저, 기존 최단경로 알고리즘을 실제 도로망에 적용할 때 많은 계산 량과 메모리가 필요하기 때문에 빠른 응답시간을 요구하는 실제 시스템의 구현에는 적합하지 않다. 이 문제를 해결하기 위하여 보다 빠르고, 정확한 탐색알고리즘이 필요하다.

최단경로문제를 풀기 위한 알고리즘으로는 가장 대표적인 게 dijkstra 알고리즘과, 휴리스틱함수를 이용한 A^* 알고리즘이 있으며 dijkstra, A^*를 변형한 Bi-directional Dijkstra 와 Bi-directional A^*들이 있다(Dennis de Champeaux, 1993. E. van Dijk, 1982. Leni Sint, 1997. P.Hart, N. Nilsson, and B. Raphael, 1998).

dijkstra알고리즘은 출발지에서 인접한 노드들까지의 비용을 계산하여 가장 적은 비용의 노드로 이동하며, 이러한 과정을 목적지에 도착할 때까지 반복 수행한다. dijkstra 알고리즘은 확실히 최단경로를 탐색할 수 있다는 장점이 있으나 목적지의 방향에 관계없이 모든 방향의의 노드를 검색하게 하여 때로는 무시할 수 없을 정도의 많은 탐색 소요시간을 갖는다는 단점을 가지고 있는데 이것은 시스템의 성능을 저하시키는 요인으로 작용하고 있다.

A*는 출발지 노드에서 현재노드의 인접한 노드들에 이르는 거리뿐만 아니라 현재 노드의 인접한 노드들에서 목적지까지의 거리도 고려하여 이동노드를 선택하는 알고리즘이다. 이것은 탐색시간은 줄일 수 있으나 인접한 노드들에서 목적지까지의 거리계산을 추가로 해야하는 부담을 안고 있다. 또한 선정된 경로가 출발지에서 목적지까지의 탐색경로가 최단경로라는 보장을 할 수 없다.

그리고 Bi-directional dikstra, A* 는 탐색을 출발지에서 목적지 방향으로 탐색하는 전방탐색과 목적지에서 출발지 방향으로 탐색하는 후방탐색을 병행하여 서로의 탐색이 중간에 교차될 때 탐색을 완료하게된다(Dennis de Champeaux, 1993. Leni Sint, 1997). 이론적으로는 양방향 탐색은 단방향 탐색에 비해 전체 탐색노드 수의 절반을 줄일 수 있는 장점이 있다. 그러나 만일 양방향 탐색은 양단에서 탐색이 교차하지 않을 경우 오히려 단방향 탐색에 비해 최고 두 배의 노드를 탐색해야 하는 단점이 있다.

그렇다면 과연 지금까지 제안된 여러 가지 최단 경로 탐색 알고리즘들 중에서 차량항법 시스템의 경로 안내에 최적의 성능을 보여줄 수 있는 알고리즘은 무엇인가? 본 논문에서는 이러한 문제에 대한 해답을 얻기 위하여 차량 항법 시스템의 경로탐색을 위한 다양한 알고리즘을 시뮬레이션을 통해 성능 비교를 하였다. 시뮬레이션은 실험 결과의 실용성을 위하여 실제 디지털 도로 지도 데이터 베이스를 이용하여 진행하였다.

이 논문이 제안하는 것은 최단 경로 선택에 있어서 계산시간과 소요 메모리를 감소시킨 방향성을 갖는 dijkstra 알고리즘을 제안하여(남상우, 박문성, 1997), 탐색할 노드 수를 줄여 최단 시간 내에 최단 경로를 찾을 수 있다는 것과, 실시간 정보를 수신하였을 경우 능동적으로 경로를 수정하는 CNS에 대한 연구이다.

본 논문에서는 제안한 알고리즘을 적용하여 구현한 차량경로계획 시스템에 대하여 제주시와 신제주의 도로망을 대상으로 모의 실험을 수행하였다.

본 논문의 나머지 부분은 다음과 같이 구성되어 있다.
II장에서는 CNS에대한 전반적인 개념을 소개하고 최단경로 탐색을 위한 여

러 가지 알고리즘들의 내용 및 특성들을 상세히 살펴보고, III장에서는 각 알고리즘들을 시뮬레이션 한 결과와 실험결과를 바탕으로 각 알고리즘들의 성능을 비교 분석하며, IV장에서는 제안한 설계구조의 시스템을 구현하고 V장에서는 결론 및 향후연구를 제시한다.

II. CNS 및 최단경로탐색 알고리즘

1. CNS

Car Navigation System(차량항법시스템) 은 GPS 위성에서 받은 위치 데이터를 차량항법 시스템 본체 또는 외부의 무선 데이터망으로부터 제공되는 지리정보를 이용하여 차량의 위치를 화면에 표시하고 경로 안내를 해 주는 시스템이다.

차량항법시스템은 POI (Point Of Interest)라고 하는 위치정보 데이터를 가지고 있다. 이 POI는 호텔, 관공서, 역, 은행. 음식점, 주차장, 주유소 등 우리가 일상생활에서 접하는 모든 곳의 위치(좌표)와 해당 항목에 대한 전화번호, 주소, 등과 같은 정보를 가지고 있다. 차량항법 시스템은 사용자가 지도상에서 목적지를 찾기 힘들 경우 이러한 위치 정보를 이용하여 운전자가 가고자 하는 위치를 선택할 수 있도록 한다.

또, 차량항법 시스템은 목적지가 선택되었을 경우 운전자의 차량이 위치한 곳에서부터 목적지까지 도로 주변의 속성 (회전속성-좌회전 금지 등, 차선 수, 도로등급-국도, 일반도로, 고속도로)을 이용하여 운전자가 목적지까지 가기 위한 최적의 경로를 자동으로 계산하여 지도 위에 경로를 표시해주고, 계산된 경로를 따라 운전자가 주행하는 동안 음성(텔레메틱스)과 Graphic 등을 이용하여 경로에 대한 정보를 알려주어 운전자가 편안하게 목적지까지 갈 수 있도록 안내하는 시스템이다.

CNS는 주로 실세계 지도를 디지탈화한 디지탈 지도를 이용하여 경로탐색을 하며, 디지털 지도에는 위도와 경도 등의 위치정보가 수록되어있다. 이때 디지탈 맵 상에서의 최단, 최적 경로 탐색을 위한 다양한 알고리즘들이 사용되고 있다.

2. 최단경로 탐색 알고리즘

디지탈 맵을 이용한 다양한 경로탐색 알고리즘들에는 dijkstra, A*, Bi-Directinal dijkstra, Bi-Directional A* 등이 있으며, 그 내용 및 특성들을 간략하게 살펴보면 다음과 같다.

1) dijkstra 알고리즘

Dijkstra 알고리즘은 이해가 쉽고 각종 최단경로 문제에 광범위하게 적용될 수 있으며, 사용이 간편함으로 가장 널리 사용되고 있는 알고리즘이다(남상우, 박문성, 1997. Andrea Borella, Franco Chiaraluce, 1998).

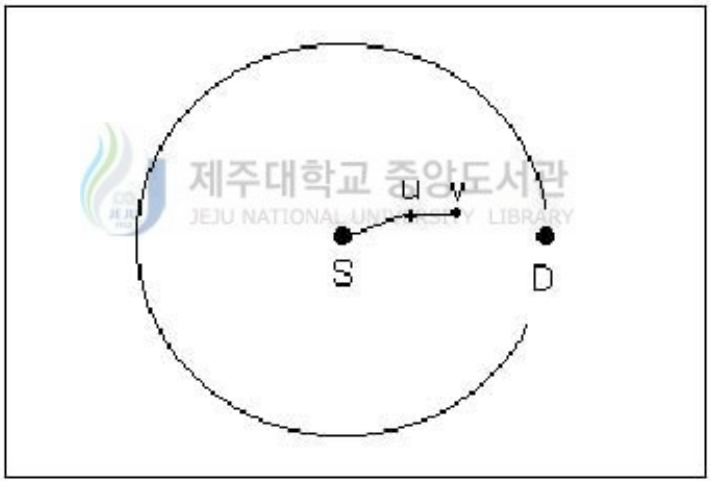

Fig 1. dijkstra's search space
(S:Starting Point D:Destination Point
u:current node y:adjacent node of u)

Dijkstra 알고리즘은 우선 초기 출발노드를 제외한 모든 노드의 표지를 ∞(무한대)로 설정한다. 그 다음 초기 출발노드에서 연결된 모든 노드의 표지에 각각의 거리를 입력한다. 이때 입력된 초기 출발노드로부터 누적거리의 값이 가장 작은 노드를 그 다음 노드로 설정하고, 그 노드에 영구표지를 표시한다.

다시 다음 노드에서 연결된 모든 노드를 찾아내어, 각각의 노드표지에 초기

출발 노드로부터의 거리의 누적 값을 입력한다. 다시 그때까지 입력된 표지값 중 가장 작은 노드를 다음 노드로 하여 위의 과정을 모든 노드에 영구표지가 될 때까지 반복한다. 식 (1)과 같이 출발지 u 노드에서 인접한 모든 v 노드까지의 비용 g를 두 노드간의 거리 L(u, g)로 계산하여 노드평가 함수 f 에 할당한 후 f의 값이 가장 작은 노드로 이동하며 이를 목적지에 도착할 때까지 반복 수행 한다.

$$f = g = L(u, v) \tag{1}$$

g는 출발 노드에서 현재 노드까지 최소 비용이다. Fig. 1에서 S는 출발지점이고 D는 목적지점이고 출발지점에서 목적지점 D를 찾고자할 때 먼저 S에 인접한 모든 노드를 검사하여 비용이 최소인 노드 u를 찾게된다. 그리고 u노드에서 인접한 모든 노드 중에서 비용이 최소인 노드 v를 선택하는 방식으로 최단 노드를 선택하게 된다. 그래서 Dijkstra 알고리즘을 사용하면 노드의 검색 반경이 Fig. 1과 같이 출발지를 중심으로 동심원을 그리며 넓어지게 된다. Dijkstra 알고리즘은 확실하게 최단 경로를 선정할 수 있는 장점이 있으나 목적지의 방향에 관계없이 모든 방향의 노드를 검색하게 되어 검색 시간이 늦어지고 탐색 공간이 많아지는 단점이 있다(전홍석, 1998).

2) A* 알고리즘

A^* 알고리즘은 목적지의 방향에 관계없이 모든 방향의 노드를 모두 검색하는 Dijkstra 알고리즘의 비효율성을 해결하기 위해 제시된 알고리즘들 중의 하나이다(전홍석, 1998. P.Hart, N. Nilsson, and B. Raphael, 1968).

A^* 알고리즘 에서는 탐색 노드의 선정에 있어서 식 (2)와 같이 Dijkstra 알고리즘에서 사용하는 출발지와의 거리 g 이외에 목적지까지 직선 거리 h를 추가하여 선정한다. g는 출발 노드에서 현재 노드까지 최소 비용이며, h는 현재 노드에서 목표 노드까지 예측된 최소 비용이다.

$$f = g + h \tag{2}$$

이로 인해 Fig. 2와 같이 목적지에서 멀어지는 노드에 대한 불필요한 계산을

생략하게 되어 전체적인 검색 속도를 증가시키고자 하는 것이다.

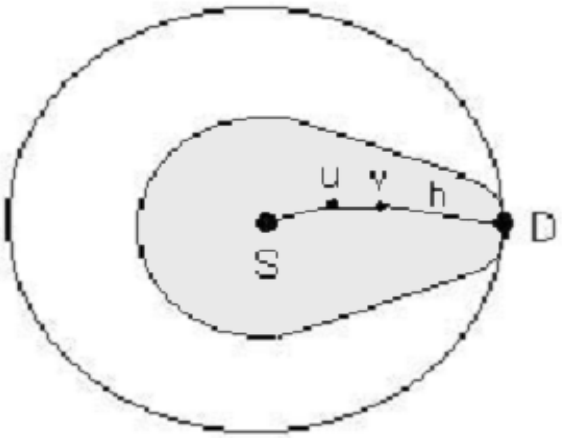

Fig. 2 A*'s search space
(S:Starting Point D:Destination Point
u : current node y : adjacent node of u)

Fig. 2에서 S는 출발지점이고 D는 목적지점이고 출발지점에서 목적지점 D를 찾고자할 때 먼저 S에 D까지의 거리를 예측하고 먼저 S에서 인접한 모든 노드들 중 D까지의 거리가 최소인 노드 u를 찾게된다. 그리고 u노드에서 인접한 모든 노드 중에서 D까지의 거리가 최소인 노드 v를 선택하는 방식으로 최단노드를 선택하게 된다. 물론 A* 알고리즘은 Dijkstra 알고리즘에 비해 h(거리)를 계산하기 위한 추가적인 계산의 부담을 가지고 있다. 또한 A* 알고리즘은 h의 사용으로 인해 선정된 경로가 최적의 경로임을 보장할 수 없게 된다(전홍석, 1998).

3) Bi-directional Dijkstra 알고리즘

Bi-directional Dijkstra 알고리즘은 Dijkstra 알고리즘의 전체 검색 반경을 줄이기 위해 출발지에서 목적지로의 검색뿐만 아니라 동시에 목적지에서 출발지로의 검색을 교대로 수행한다. 결국 최단 경로 탐색은 양단에서의 검색이 교차

되는 시점에서 종료되어지며 양방향에서 선정한 최단 경로의 합이 출발지에서 목적지까지의 최단 경로가 된다. Dijkstra 알고리즘의 검색 반경이 출발지를 중심으로 동심원을 그리며 퍼지기 때문에 Bi-directional Dijkstra 알고리즘은 이론적으로 검색 반경이 Dijkstra 알고리즘에 비해 절반으로 줄어들어 경로 탐색 시간을 줄이게 된다. 그러나 Bi-directional Dijkstra 알고리즘 역시 Dijkstra 알고리즘과 같이 목적지의 방향과 관계없이 모든 방향의 노드를 탐색하는 단점을 여전히 지니고 있다(전홍석, 1998. Leni Sint, 1997).

4) Bi-directional A^* 알고리즘

Bi-directional A^* 알고리즘은 Dijkstra 알고리즘의 단점을 해결하기 위한 A^* 알고리즘과 Bi-directional Dijkstra 알고리즘을 통합한 알고리즘이다. 즉, 출발지와 목적지에서 교대로 탐색을 수행하며 이때 탐색 노드의 선정을 위해 탐색 시작 노드부터의 거리 이외에 탐색 종료 노드까지 직선 거리 h를 함께 고려한다. 그러나 Bi-directional A^* 알고리즘은 A^* 알고리즘과 마찬가지로 h의 사용으로 인해 선정된 경로가 최단의 경로임은 보장할 수 없다(전홍석, 1998. Leni Sint, 1997).

III. 제안 알고리즘

1. 위치정보를 이용한 최적경로 탐색 알고리즘

앞에서 우리는 Dijkstra나 A*알고리즘들이 각각의 장점과 단점을 갖는 다는 것을 보았다. 제안하는 알고리즘은 정확한 탐색 결과를 제시하는 Dijkstra 알고리즘을 개선하여 검색영역을 θ각 안에 포함되는 인접 노드들만을 검색하게 함으로써 필요한 메모리양과 연산 시간을 줄이도록 하였다. 즉, 노드 중 목적지 도달에 불필요한 노드들을 탐색대상에서 사전에 제거함으로써 계산시간과 메모리의 효율적 이용이 가능해 지게 된다.

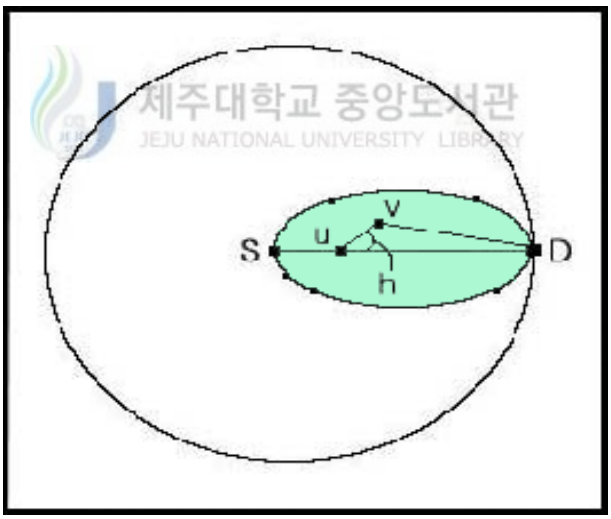

Fig. 3 Search Space for Proposed Algorithm
(S:starting point D:destination point
u:current node v:adjacent node of u
h:line(u,v) and line(u,D) of angle)

Fig. 3은 본 연구에서 제안한 알고리즘의 줄어든 검색 반경의 모습을 보여준다. S는 출발지점이고 D는 목적지점이고 출발지점에서 목적지점 D를 찾고자

할 때 먼저 직선 S와 인접노드 그리고 직선 S와D 사이의 각이 일정 범위 안에 포함하는 노드 중에서 비용이 최소인 노드 u를 찾게된다. 그리고 u노드에서 인접한 모든 노드 중에서 D까지의 이루는 각이 일정 범위(여기서는 h) 안에 포함하는 노드 중에서 비용이 최소인 노드 v를 선택하는 방식으로 최단노드를 선택하게 된다.

현재위치 u에서 목적지 D 노드까지 인접한 모든 v 노드를 찾을 때 선분 u, d 와 선분u, v의 방향 각을 이용해 탐색 대상 vertex T에 속할 수 있다. 이때 방향각 설정은 휴리스틱함수를 사용한다. 목적지의 방향과 일정 각 범위 h 안에 속하지 않는 노드는 탐색을 하지 않는다. 새로운 노드의 확장을 위한 평가함수는 식(3)과 같다.

$$f = g + h \tag{3}$$

h값은 직선(u, d)와 직선(u, v)의 삼각함수 COS 제2법칙을 이용하여 계산한다. 물론 Dijkstra 알고리즘에 비해 h(각)를 계산하기 위한 추가적인 계산의 부담을 가지고 있다.

이로 인해 Fig. 3과 같이 목적지의 방향과는 일정정도 이상 어긋나는 노드에 대한 불필요한 계산을 생략하게 되어 전체적인 검색 속도가 증가되리라고 예측할 수 있다. 식 (3)에서 최적의 성능을 보이는 h의 값은 도로의 특성에 따라 가변적이며 거리가 멀어짐에 따라 각도도 작게 잡아도 됨을 알 수 있다.

이때 최단경로 탐색을 위해 교차로 수, 차선 폭, 노선의 길이에 따라 Weight 값을 적용하였으며 초기 Weight 값을 구하는 식(4)은 다음과 같이 구할 수 있다.

$$Weight = \left(\frac{\text{노선의길이}(m)}{1000(m)}\right) + \left(1 - \left(\frac{\text{차선폭}}{\text{최대차선폭}}\right)\right) \tag{4}$$

제안알고리즘의 절차는 Fig. 4와 같다.

1. Initialization :
   T = {s} s:source vertex, w(i,j) = link cost
   L(n) = w(i,j) for n≠s : vertex s에서
   n까지의 최소비용

2. Get next vertex
   vertex j 와 인접한 vertex 중 목적지의 방향과
   ∠θ인 범위에서
   최소비용인 T에 속하지 않는 인접
   veretx x를 추가

3. Update least-cost paths
   영구 결정된 T의 link 비용에 새로 추가된
   vertex의 link 비용을 추가
   L(n) = min[L(n), L(x)+w(x,n)]

Fig. 4 Proposed Algorithm Procedure

2. 탐색 각도의 능동적 조정

dijkstra알고리즘처럼 인접한 전체 노드를 대상으로 탐색하지 않고, 기본 방향각 θ를 가지고 범위 안에 속하는 인접노드만을 대상으로 경로 탐색을 했을 경우, 특정한 도로상황에서는 목적지를 찾지 못할 경우가 발생할 수도 있다.

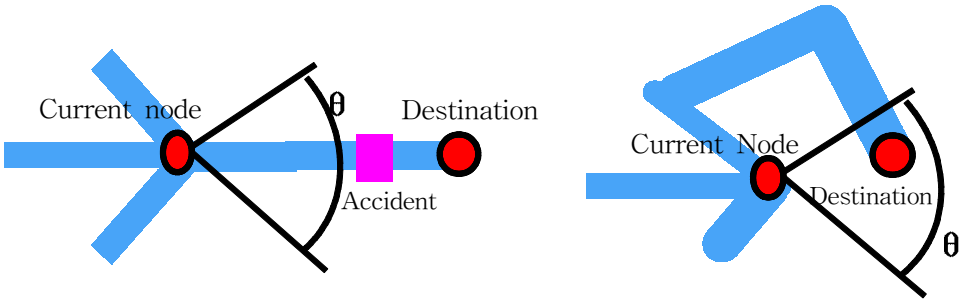

Fig. 5 Abnormal situation for searching

Fig. 5에서 왼쪽 그림은 앞 도로에서 차량 사고가 발생하여 더이상 직진 할 수 없는 상태이며, 탐색 각 θ안에 탐색할 도로가 존재하지 않는 경우를 보여주며, 두번째 그림은 도로자체의 특이성 때문에.. θ각안에 목적지까지 이르는 도로가 없는 경우를 보여준다.

이때 방향각을 일정 비율로 증가시켜 다시 목적지를 찾을 수 있도록 한다. 본 논문의 시뮬레이션에서는 방향 각을 10°씩 증가하도록 구현을 했다.

3. 실시간 동적 정보

실세계의 차량이 운행할 수 있는 도로상에는 각종 차량사고, 각종행사, 자연재해, 교통혼잡, 각종공사, 기상변화 등이 존재한다. 기존의 대부분의 CNS는 초기 경로를 설정하면 시시각각 발생하는 위와같은 사건들에 반응할 수 없는 정적인 시스템들이었다. 따라서 위와같은 각종 정보들을 Fig. 6처럼 기상청이나, 교통정보를 제공해주는 교통관제 센터로부터 무선단말기 등의 무선통신 시스템을 통하여 교통정보를 수신하여 시스템에 적용할 수 있다면 신시간으로 각종 정보를 동적으로 반영하는 능동적 CNS를 구현할 수 있으며, 운행 중인 차량은 항상 최적의 경로를 유지할 수 있게 된다.

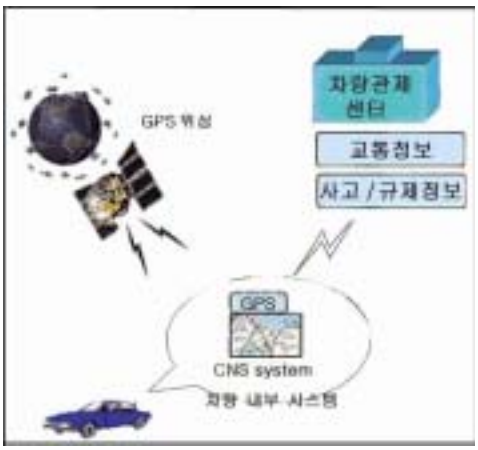

Fig. 6 Proposed system model

능동적 CNS가 받아들일 수 있는 정보는 아주 많으며, 그중에 대표적인 5가지에 대해서 각각의 경미함, 과중함에 따라 다양한 Weight를 부여할 수 있다.

차량사고, 각종행사, 자연재해, 교통혼잡, 각종공사 등의 교통정보에 따라 Weight값을 적용해서 실시간으로 최적의 경로로 갱신했다.

그리고 각각의 동적 정보는 그 사고의 심각성에 따라 상, 중, 하로 나뉘어져 있다. Table. 1은 동적 정보의 종류와 그에 따른 C값을 표로 나타낸 것이다.

Table. 1 Weights for dynamic information

동적 정보 / 사고의 심각성	차량사고	각종행사	자연재해	교통혼잡	각종공사
상	3	3	3	3	3
중	2	2	2	2	2
하	1	1	1	1	1

이러한 사건이 발생하고, 무선 통신 시스템을 통해 CNS에 수신되면 CNS는 기존의 도로 특성에 따른 Weight에 사건정보를 반영한 새로운 Weight를 계산하여 반영하게 되며, 동적 정보에 따른 새로운 Weight 값을 구하는 식(5)은 다음과 같다.

$$Weight = \text{초기}\,Weight + \left(\frac{\text{초기}\,Weight}{3} * C\right) \quad (5)$$

C는 사고의 심각성을 값을 갖는 변수

동적 정보가 발생 할 경우의 알고리즘은 Fig. 7과 같다.

```
procedure Dynamic_Information_Event(Sender: TObject);

    C=3, 2, 1 ;
Weight =초기 Weight+(초기 Weight/3 * C)
    Dijkstra(i,j) ;
    while (Destination is not found)  {
        ∠θ = ∠θ + 10°;
        Dijkstra(i,j) ;
     }
}
```

Fig. 7 Change weight with dynamic information

Fig. 8은 동적 정보가 발생했을 경우의 상황을 나타내고있는데 S지점에서 D 지점까지의 초기 제안 알고리즘으로 찾은 경로는 굵은 선으로 표시되며, 현재 차량이 진행중이다. 그리고 차량이 운행 중에 E지점에서 사고가 발생하여 사고에 대한 동적 정보를 수신 받고 그대로 가야하는가 또는 사고지점을 우회해서 가는지를 판단하게 된다. 이때 사고가 경미했을 경우는 전체 비용합이 3이 되어 우회비용보다 작다. 그리고 만약 사고가 중형일 때 는 우회해서 갈 경우 의 비용의 합은 4 그리고 그대로 가야할 경우 초기 비용의 합은 5가 되어서 u와 v 지점을 거쳐서 가는게 최적 경로가 되므로 경로를 재 수정 하게 된다.

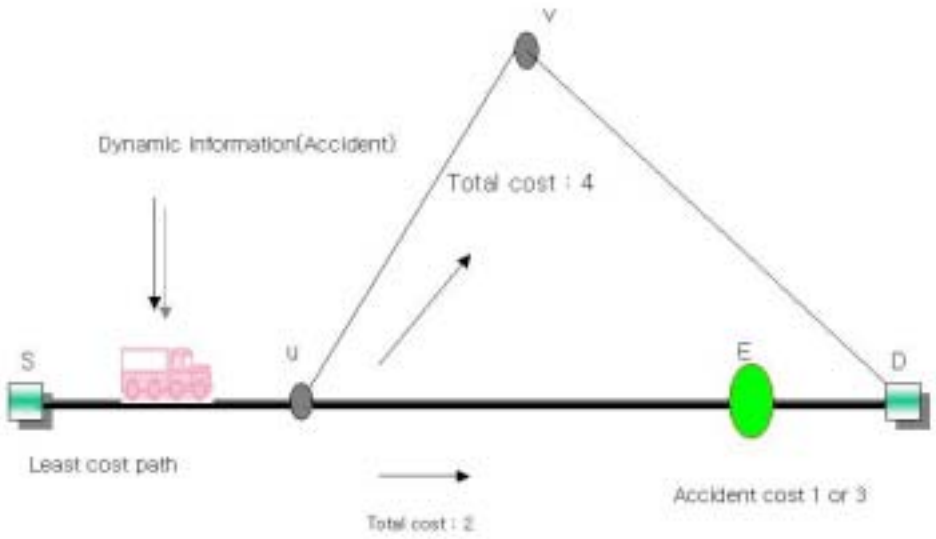

Fig. 8 Adjusting new path when an event happens

4. 전체 알고리즘

1) 기본개념

① weight및 탐색 vertext 리스트, 링크비용 초기화 :

T = {s} s:selected vertex,

w(i,j) = link cost between vertext i and j,

L(n) = 0 ; total cost for searched path

② 다음 노드 선정 :

vertex j 와 인접한 vertex 중 목적지의 방향과 ∠θ인 범위에서 최소비용인 T에 속하지 않는 인접 vertex x를 T에 추가

③ 최소 비용 경로 갱신 :

전체 비용 L(n)에 새로 추가된 T의 vertex중 최소 link 비용을 추가

L(n) = min[L(n), L(x)+w(x,n)]

다음은 제안 알고리즘 전체에 대한 소스 코드이다.

```
procedure dijkstra (a, b : integer);
var
  distance, parent : array[1..70] of integer;
    {parent는 확정표시, distance[i]는 시작점에서 i까지의 최단 거리}
  check : array[1..70] of boolean;
  i, j, min, p, maxint, ncount : integer;
   {a는 시작점(출발지점), b는 끝점(목적지점), min은 거리의 최소값}
  OP, JP, OJ, cosb, radi : double ;

begin
  chkk:=1 ;
  ncount:=0 ;
     for i:=1 to 50 do chk[i]:=0 ;
     for i := 1 to 70 do begin
       check[i] := false;              {방문상태 초기화}
       distance[i] := maxint;          {거리를 전부 최대값을 넣음}
       intp[i]:=0 ;
     end;

  parent[b] := 0;
  distance[b] := 0;        {자신에서 자신까지의 거리는 0이므로..}

  for i := 1 to 70 do begin
    min := maxint;
    for j := 1 to 70 do begin
      if (min > distance[j]) and (not check[j]) then begin
        min := distance[j];       {연결된 곳 중 최단거리인 곳을 찾음}
```

```
      p := j;                    {연결된 곳 중 최단거리인 곳을 확정}
    end;
end;

if min = maxint then break; {연결된 곳이 아예 없다면}
check[p] := true;

for j := 1 to 70 do begin
 if (e[j,p] <> 0) and (distance[j] > distance[p] + e[j,p]) then
   begin
     JX:=xyt[0,j] ;
     JY:=xyt[1,j] ;
     PX:=xyt[0,P] ;
     PY:=xyt[1,p] ;
     OP:=0 ; JP:=0 ; OJ:=0 ; cosb:=0.0 ; radi:=0.0 ;
     OP:=sqrt(((OX-PX)*(OX-PX))+((OY-PY)*(OY-PY))) ;
     OJ:=sqrt(((OX-JX)*(OX-JX))+((OY-JY)*(OY-JY))) ;
     JP:=sqrt(((JX-PX)*(JX-PX))+((JY-PY)*(JY-PY))) ;
     if OP=0 then radi:=0 else begin
        cosb:=(((OP*OP)+(JP*JP)-(OJ*OJ))/((OP*JP)+(OP*JP))) ;
        radi:=arccos(cosb)*180/3.1415926 ;
       {COS 제이법칙을 사용하여 인접 노드의 ∠∅ 각도를 구함}
     end ;
     if (radi>=0) and (radi<=strtoint(edit4.text)) then
       begin
          distance[j] := distance[p] + e[j,p];
                    {∠∅ 각도 범위에 들어있는 인접노드를 찾음}
          parent[j] := p;
          intp[ncount]:= p ;
          edit3.text:=edit3.text+inttostr(p)+' ';
          ncount:=ncount+1 ;
```

```
                cx:=ncount ;
              end ;
            end;
          end;
          end;
    panel12.Caption:='총노드수 : '+inttostr(ncount) ;
    if distance[a] = maxint then begin
  end else begin
        i := a;
        while i <> 0 do begin
        chk[chkk]:=i ;
        chkk:=chkk+1 ;  {선택된 노드들을 추출}
        i := parent[i];
       end;
    end;
end;
```

IV. 시스템 구현 및 고찰

1. 시스템 환경

구현한 시스템 환경은 다음과 같다.

- 하드웨어 : Pentium III
- 운영체제 : Windows 98
- 소프트웨어
  1) 개발언어 : Delphi 4.0
  2) GIS 엔진 : MapObjects 2.0
  3) DataBase : DbaseIII , Ms-Access
  4) Map Tool : AutoCad R14
  5) Data : 제주시와 신제주의 디지털 도로망도

제안한 시스템의 전체 구성요소들은 델파이(Delphi4.0)를 이용하여 구현하였으며 맵엔진은 ESRI사의 맵오브젝트(Mapobjects2.0)을 사용하였다. 수치지도로는 제주시와 신제주의 2차선 이상 도로를 벡터방식의 데이터를 사용하였으며 도로의 모든 정보(위치좌표,도로길이,차선폭,교차로표시,웨이트값,등...)를 Dbase와 MSACCESS에 담고 있다. Fig. 9는 제주시 디지탈 지도를 보여준다.

맵은 차량 항법 시스템에서 제주시와 신제주 지역을 표현한 것인데 노드 수가 70개이며 링크 수는 110개를 포함하고 있다. 노드는 내부적으로 번호로 구분하였다. 또한 실험의 편의를 위하여 편도 2차선 이상의 도로만을 대상으로 실험하였다.

Fig. 9 Target digital map

2. 구현된 기능

1) 시작지점과 목저짐지정의 설정

이 시뮬레이션 시스템에서는 초기의 차량의 위치와 목적지는 마우스 클릭을 통하여 임의로 지정할 수 있으며, 경로를 탐색하게 되면 탐색된 경로를 화면상에 표시해 주게 되며 차량의 진행 상황도 보여준다.

2) 지도의 줌 기능

보다 정확한 지도를 보기 위해서 지도의 확대, 축소기능을 구현했다.

3) 지도의 팬(Pan)기능

지도의 줌(Zoom)기능시 이동을 돕기 위해 팬기능을 둠으로써 바로 바로 이동이 쉽도록 했다.

4) index 기능

건물, 도로, 교차로 등의 정보를 알아볼 수 있도록 마우스 클릭시 해당 건물, 도로, 교차로 등의 정보를 표시 할 수 있도록 했다.

5) 인쇄기능

원하는 지도를 프린터 해서 볼 수 있도록 했다.

6) 차량의 현재의 위치(좌표로써 표현)

지금까지 운전자는 타 지역에 가거나 여행을 나서면 대부분이 지도에 의존했다. 이 지도책에서 축적과 주변의 지형 지물을 판단해 지도상의 위치를 가늠해야 했는데 지도 보는 방법에 익숙하지 않은 사람은 지도상의 위치를 쉽게 알지 못할 뿐 아니라 잘못된 판단으로 많은 시간과 비용(연료비)를 낭비하게 되었다.

전자지도상에 차량의 위치를 항상 표시를 해 준다.

7) 예상소요시간, 거리 표현기능

시시각각 변화되는 교통환경에 따라 결정될 링크별 운행 소요시간을 도로폭, 교차로 수, 기타 교통정보에따라 추정하여 표신하였다.

8) 최적경로발견

최적 경로 탐색 문제는 도로길이뿐만 아니라 교통 혼잡도, 교차로 수, 도로용량, 차선 수, 혼잡도로의 존재여부 등과 같은 많은 변수들을 고려하여야 함으로 더욱 복잡한 문제가 된다. 제안된 알고리즘을 이용하여 최적 경로를 선택하는데 필요한 메모리 량과 소요 시간을 줄였다.

9) 운행중 실시간 상황정보 반영

운행중 각종 사고, 행사, 기후 등의 정보를 무선통신시스템으로부터 수신하기 위한 시뮬레이션 램덤 함수를 이용하였으며, 매 10초당 한번씩 다양한 종류의 사건이 발생토록 하였으며, 사건이 발생할 때 마다 이를 수신한 시스템은 비용을 재계산하여 필요한 경우 경로를 능동적으로 수정할 수 있게 하였다.

3. 시뮬레이션 결과 및 분석

1) 초기화면

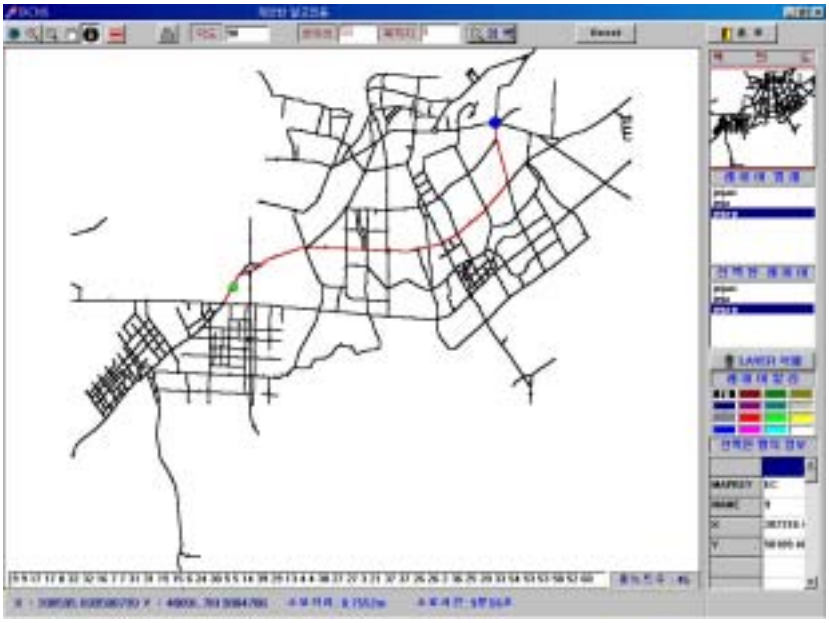

Fig 10. Simulation Screen

2) 탐색 각도의 설정

Table. 2는 각도별로 랜덤하게 500회 시뮬레이션 한 결과 h의 값이 90도일때 Dijkstra 와 A*와 유사하게 도로의 특성에 관계없이 가장 안정적으로 경로 탐색을 진행함을 알수 있다. Fig. 11은 이를 차트로 보여준다.

Table. 2 Search results for angle variation

각도	10°	20°	30°	40°	50°	60°	70°
목적지를 찾지 못한 횟수	373	352	296	231	96	49	21
평균 탐색노드수	4.5690	4.5921	6.2714	8.2886	10.4745	12.9369	16.1995
평균 탐색 비용	1.9934	2.1881	3.4376	4.4097	5.2362	5.4968	5.6663
각도	80°	90°	100°	110°	120°	A*	Dijkstra
목적지를 찾지 못한 횟수	12	6	5	0	0		
평균 탐색노드수	18.8107	21.4483	26.2588	28.0548	0	23.1976	73.4478
평균 탐색 비용	5.4155	5.3780	5.4801	5.2833	0	6.0622	5.3917

Fig. 11 Chart about search ressult for angle variation

3) dijkstra, A* 알고리즘과의 비교

Fig. 12는 탐색각을 90도로 하여 최적 경로를 탐색한 결과를 보여준다.

Fig. 12 The results of shortest path search

Table 4는 각각 Dijkstra 알고리즘과 A*알고리즘, 90도 탐색각을 갖는 제안 알고리즘으로 찾아낸 최단거리 경로 및 그에 따른 각 알고리즘의 탐색 노드 수와 비용의 합을 나타낸 것이다. Table 4에서 나타난 바와 같이 제안한 알고리즘의 탐색노드수가 가장 적은 것을 볼 수 있다.

Nodes / Alg.	Total nodes count(including duplicates)	Total node's Weight
Dijkstra	73.2442	5.3917
A*	23.1976	6.0622
Proposed	21.7662	5.4113

Table. 4 The visited node order and count, weight

Fig. 13은 500회에 걸친 실험을 통해 각 알고리즘 별로 선정된 최단 경로의 비용의 합을 나타낸 그림이다. 실험 결과에 의하면 제안 알고리즘은 Dijkstra 알고리즘에 의해 선정된 최단 경로의 길이와 거의 비슷하다는 것을 볼수 있으며, 최적성을 갖는다고 할 수 있다.

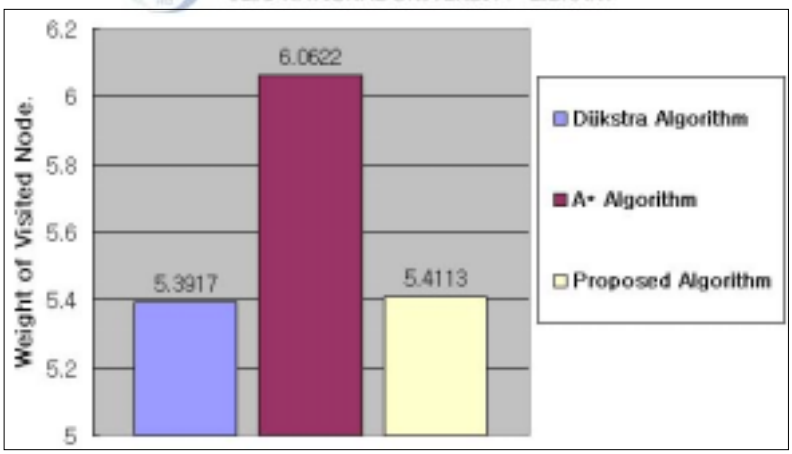

Fig. 13 Sum of searched nodes

Fig. 14는 각 알고리즘들이 전체 실험을 통해 방문한 노드 수의 총 합을 도표로 나타낸 것이다. 여기에서 보면 알수 있듯이 제안 알고리즘이 최적성을 가

지면서도, 가장 적은 탐색 노드수를 갖는다는 것을 알수 있다. 따라서 탐색 노드수가 적다는 것은 메모리 필요량과 탐색 시간을 줄일수 있음을 말한다.

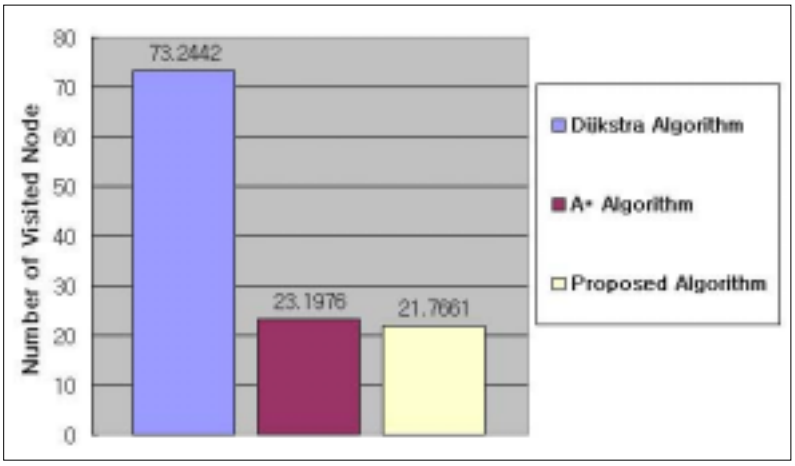

Fig. 14 Weights sum of results paths

4) 동적 정보에 대한 능동적 경로 수정

Fig. 15, Fig. 16, Fig. 17, Fig. 18, Fig. 19, Fig. 20은 차량이 운행중 동적 정보를 수신하였을 경우 능동적으로 경로를 재수정하는 과정을 보여준다. 각각의 그림에서 녹색 원은 운행중인 차량의 위치이고 파란색 원은 목적 지점이다. 그리고 빨간선은 차량이 운행할 최단 경로를 보여주고 있으며, 빨간색 원은 일정 주기적으로 발생한 사고 지점을 보여준다. Fig. 15, Fig. 18은 최초 시작 지점에서 목표지점까지의 경로를 보여주고, Fig. 16, Fig. 19는 1차 사고 발생 후 수정된 경로를 보여준다. Fig. 17, Fig. 20은 다시 2차 사고 발생 후 최종 수정된 경로를 보여주고 있다. 사고는 10초 간격으로 랜덤하게 탐색 경로 상에 발생한다. 이 그림에서 알수 있듯이 구현한 시스템은 능동적으로 동정 정보를 이용하여 최적 경로를 갱신함을 알수 있다.

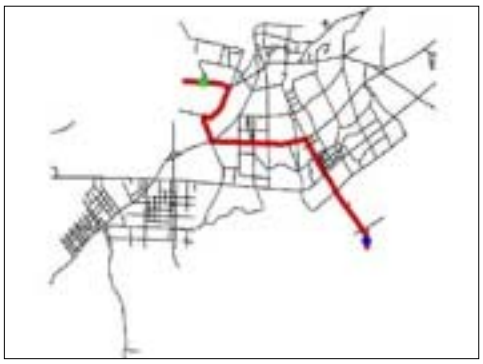

Fig. 15 first search path

Fig. 16 first event after Update Path

Fig. 17 second event after Update Path

Fig. 18 first search path

Fig. 19 first event after Update Path

Fig. 20 second event after Update Path

V. 결론 및 향후 연구

본 논문에서는 차량 항법 시스템에 적용하기 위한 빠른 경로 탐색을 할 수 있는 알고리즘을 제안하고 시뮬레이션을 통하여 그 성능을 평가하였다.

제안된 알고리즘은 디지털 도로 맵상의 위치정보를 이용하도록 고안되었으며 탐색 범위를 줄이기 위하여 출발지와 목적지간에 만나는 각 노드에서 방향각 정보를 이용하였다. 다양한 각도를 가지고 평가한 결과 90도의 탐색각을 유지하였을 경우에 dijkstra나 A*알고리즘에 유사한 최적성을 보였으며, 보다 적은 노드 수를 탐색하고도 최적 경로를 구할 수 있었다.

또한 지도상의 단순거리, 교차로와 같은 정적인 정보와 시간의 흐름에 따라 시시각각 변하는 사고, 교통량의 변화, 날씨 변화와 같은 동적인 교통상황까지도 Weight로 반영하여 능동적으로 최적 경로를 유지할 수 있는 시스템을 구현하였다.

추후 알고리즘의 좀 더 세밀한 평가를 위해 제주도 전지역 지도를 대상으로 적용하여 평가를 할 계획이며 더 많은 동적, 실시간 매개변수들을 고려한 알고리즘의 개선도 지속적으로 수행해 나갈 것이다.

그리고 대부분의 차량항법장치는 차량의 위치를 시각적으로 제공하고 있다. 이런 시간적인 제공으로 인해 운전자에게 새로운 작업을 부여함으로써 운전중 불필요한 작업으로 인한 전방주시 태만이 발생하기 때문에, 음성인식기술(텔레매틱스)을 접목할 필요도 있다.

참고문헌

김기섭, 1999, 실시간 차량경로결정을 위한 동적최단경로 알고리즘, 홍익대학교 산업공학과 석사 논문.

남상우, 박문성, 1997, 방향성을 고려한 우편경로 최적화 시스템의 최단 경로 생성 알고리즘 연구, 한국정보처리학회, 정보처리 논문지, 제4권 2호, pp. 491-498.

전홍석, 1998, 차량 항법 시스템의 경로 탐색을 위한 탐색 알고리즘들의 성능 비교, 한국정보교육학회 논문지, 제2권 2호, pp 252-259.

Andrea Borella, Franco Chiaraluce, 1998, Multicast routing in BMSNs through greedy, weighted greedy, and Dijkstra algorithm: a comparative analysis. SYBEN, pp. 444-454.

Dennis de Champeaux, 1993, Bidirectional Heuristic Search Again JACM 30(1), pp. 22-32.

E. van Dijk, 1982, Heuristic Search with Partial Node Expansion and Bi-Directional Search in Product Space, pp. 180-182.

Leni Sint, 1997, An Improved Bidirectional Heuristic Search Algorithm, JACM 24(2), pp. 177-191.

P.Hart, N. Nilsson, and B. Raphael, 1968, A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Transactions on Systems, Science and Cybernetics, SCC-4(2), pp. 100-107.

감사의 글..

겨울이 되어 춥고 움추려 드네요. 날씨 탓인지 쓸쓸함과 허탈한 마음도 있지만, 대학원이라는 하나의 과정을 마무리하여 아쉬움보다는 편안한 느낌이 드네요.
재작년 바로 이때에 대학원에 입학해서 공부라는 것을 어떻게 해야하고, 왜 해야하는지를 깨닫게 해준 시기라고 봅니다. 이제 와서 그런 것을 깨달았다는 것이 정말 사소할지 모르지만 앞으로의 제 생활에 정말 필요한 밑거름이 되지 않나 생각합니다.
우선 2년이라 기간동안 뒷바라지 해주신 부모님께 감사하다는 말을 전하고 싶습니다. 그리고 또한 묵묵히 지켜 봐준 나의 아내 그리고 내 동생들에게도 감사를 드립니다.
항상 만족스럽지 못한 제자이실 테지만 격려해주시고, 이끌어 주신 이 상준 교수님께 감사드립니다. 그리고 논문 심사에 애써주신 곽호영 교수님과 변상용 교수님께도 감사를 드립니다. 그리고 정말 열심히 하는 것이 이것이다라는 것을 보여주신 영민(올해는 꼭 결혼합서)이형, 항상 직장 생활에 지쳐있는 나의 동기 병운(병운아! 올해는 술좀 작작 먹어라)이 후배들을 잘 채경주시는 휴찬이형님, 모든일에 열심히인 은경이, 직장생활 중에서도 학업에 열중이신 아나 누나, 그리고 우리 연구실에 막내 은주와 광석이 에게 감사의 말을 전하고 싶습니다. 그리고 너무나 많은 도움을 준 대학원 동료 선,후배분들게 감사의 말 전하고 싶습니다.
이밖에 미처 감사드리지 못한, 저와 인연이 닿았던 많은 분들게 진심으로 감사드립니다.
항상 나란 존재는 다른 사람과의 관계 속에서 나 자신이 결정되나 봅니다. 저를 있게 해주신 분들은 바로 제 주위의 사람들 이었습니다.