'93 한국자동제어학술회의논문집 (1993.10.20~22)

주행 안내 시스템

°김태진, 장택준, 한민홍
고려대학교 공과대학 산업공학과

Auto-Drive-Guidance System

°Tae-Jin Kim, Taek-June Jang, Min-Hong Han
Department of Industrial Engineering, Korea University

ABSTRACT

This paper presents an Auto-Drive-Guidance System which provides a path to the destination with the shortest driving distance(or time), as well as service information such as the location of gas stations, hospitals, or police stations.

This system displays on the monitor screen the best driving path to the destination, points the current car position on the map, informs the driver of current position by voice whenever the car passes well-known places, or displays the map the driver wishes to view.

With this system, driving becomes more comfortable, and traffic jams will be greatly reduced. As a result, gasoline comsumption will be reduced and so air pollution. The system can also be applied to such areas as communication network, geographic map, and tour information.

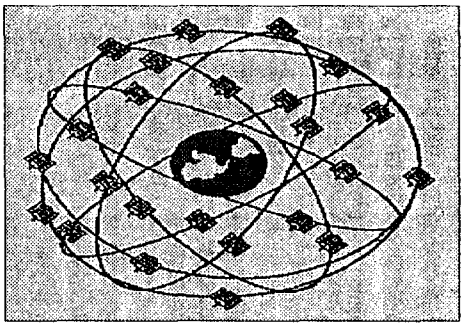
1. 서론

당신이 운전을 하여본 경험이 있다면 나는 현재 어느 위치에 있으며, 원하는 곳을 가기 위해서 어떻게 가야할 것이며, 목적지까지 어느 정도의 시간이 소요될 것인지 등이 필요함을 느꼈을 것이다. 이 밖에도 전방 교차로까지 얼마나 남았으며, 교통이 얼마나 정체되고 있고, 날씨가 어떠한 지도 궁금할 것이다. 또한 가까운 호텔이나 병원, 경찰서를 가고 싶을 때도 있을 것이다. 그래서 도로 교통 지도를 펴 보거나, 주행 중에 도로 안내 표지판을 유심히 바라보게 되고, 때로는 교통 방송도 청취하게 된다. 이런 기능을 주행중에 컴퓨터가 대신하여 알려준다면 매우 편리할 것이다.

이런 시스템을 구현하기 위해서는 현 위치및 주위의 지도 정보를 알아야 하고 도로의 체증 정도 및 도로의 상황도 알아야 한다. 현재 주위에 무엇이 위치해 있는지를 알기 위해서는 그 정보를 데이타베이스화 하여 저장해야 하며 그 정보를 빠른 시간에 검색할 수 있어야 한다. 또한 도로의 체증 정도 및 도로의 상황을 미리 알기 위해서는 교통 관제 센터와 같은 곳에서 교통 정보를 수신 분석한 후에 차량에 장착된 각 시스템에 무선방송으로 알려 줄 수 있어야 한다.

마지막으로 현재의 절대 위치를 알기 위해서는 GPS(Global Positioning System)를 이용할 수 있다. GPS는 11시간 58분마다 지구를 한 바퀴씩 돌고 있는 24개(3개의 spare)의 인공 위성을 사용하여 하루 24시간 어느 곳에서도 위치를 파악할 수 있는 시스템이다.[1] (<그림 1> 을 참조)

위도와 경도 위치는 인공 위성으로부터 GPS 수신기까지의 거리를 알면 삼각법으로 계산하여 얻을 수



<그림 1> 지구를 돌고 있는 GPS 인공 위성

있다. 이 거리는 인공위성에서 수신기까지의 걸리는 시간(300,000 km/sec)으로 계산된다. 그런데 수신시에 두 인공위성의 시간의 차이가 생기기때문에 이 시간 차이를 해결하기 위해서 세번째 인공위성이 시계 기능을 하여 두 인공위성의 수신시간 차이를 해결해 준다. 그러므로 인공위성들 중 3개가 수신되면 위도와 경도를 알 수 있고, 4개가 수신되면 위도와 경도 외에 고도까지를 알 수 있다.[2)]

GPS 시스템의 정확도는 군사용이 5m 정도로 알려져 있으나 민간에 판매되는 장비는 평균 25-100m정도이다. 이 오차로 수신 상태가 안 좋아지면 길을 안내할 때에 오류를 일으킬 우려가 있다. 이를 보정하기 위하여 절대위치를 알고 있는 다른 한 지점의 송신을 받아 오차를 수정하는 Differential System을 사용하게 되면 정확도가 원래 군사용 정도로 회복이 가능하다.

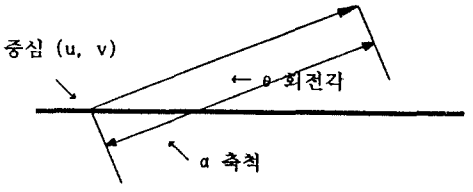
2. 주행 안내 시스템의 구성 요소

2.1. 지도

2.1.1. 지도 변환

지도를 변환시키는 요소는 화면 중심, 회전, 축척의 3가지 요소를 가진다. 화면의 중심은 전체의 지도 중에서 현재 보고 있는 중심 부분을 찾아 보여준다. 회전은 가고자 하는 방향이 항상 화면의 위를 향하도록 한다. 축척은 보고 있는 지도의 크기를 조절한다.

이는 좌표계에서 벡터 화살표로 설명할 수 있다.



< 지도 변환 벡터 화살표 >

화살표의 시작점은 중심 변환의 좌표(u, v)를 나타내고, 화살표가 가리키는 방향은 회전각(θ)을 나타내고, 화살표의 길이는 축척(α)을 나타낸다.

변환 순서는 먼저 중심 변환을 한 후에 회전과 축척 변환을 한다.

이를 수식으로 표현하면 다음과 같다.

$$x' = \alpha[\ (x-u)\cos\theta + (y-v)\sin\theta\ ]$$

$$y' = \alpha[\ -(x-u)\sin\theta + (y-v)\cos\theta\ ]$$

(x', y') : 변형된 좌표
(x, y) : 고유 좌표
(u, v) : 화면 중심 좌표
α : 축척
θ : 회전 각도

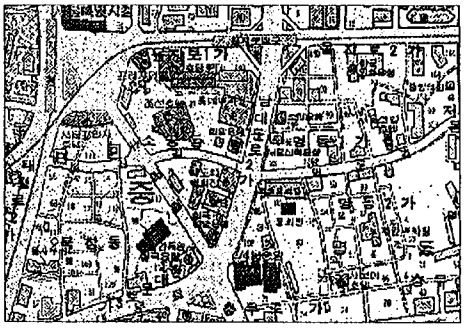
< 지도 변환 수식 >

2.1.2. 지도 저장 방식

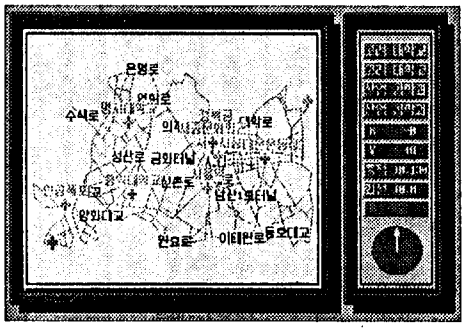
지도를 저장하기 위해서는 크게 두 가지 방법으로 접근이 가능하다. 그 하나는 실제로 도로 교통 지도를 스캐너(Scanner)로 입력 받아 이를 이미지 화일 (Image File)로 저장, 사용하는 방법과 도로의 교차로와 다른 위치 정보를 기초로 하여 화면에 도로 및 기타 정보를 복원하여 나타내는 방법이 있는데, 첫번째 방법을 Raster 방식<그림 2>, 두번째 방법을 Vectorizing방법 <그림 3>이라고 부른다.

Raster 방식은 화면에 이미지(Image)를 띄우게 되므로 산뜻한 모양새를 가지고 있으며, 목적에 필요하지 않은 정보도 위치 파악이나 정확한 정보에

도움을 주도록 별도의 자료(Data)저장 없이 가능하다. 그러나, 이미지 화일때문에 많은 이미지 정보를 저장하여야 하며, 화면에 이미지를 나타내기 위하여 많은 시간이 소요 된다. 그 외에도 화면이 유연하게 움직이면서 정보를 계속 보여줄 수가 없으며 크기를 자유자재로 조절하기가 힘든 단점을 가지고 있다. 또 정보의 변화에도 많은 비용과 시간이 들 수 밖에 없다.



<그림 2> Raster 방식 지도



<그림 3> Vectorizing 방식 지도

Vectorizing방법은 화면에 보일 모든 정보 즉 위치 및 지명 등을 모두 복원해 그려내야 한다. Raster방식에 비하여 모양이 보기 좋지 않지만 이미지 화일을 사용하지 않으므로 정보량이 대폭 감소되고 방법에 따라서 화면에 정보를 나타내는 데에 시간이 덜 걸리게 할 수 있다. 게다가 크기나 회전, 위치 변화 에 제약을 받지 않으며 정보의 변화에도 상당히 민감할 수 있다. 그러나, 이 방법도 많은 정보를 효율적으로 다루어야 만 그 효과를 누릴 수 있다.

두 가지 방법을 사용 용도로 비교해 보면 Raster 방식은 움직임이 적고 상세한 정보가 필요한 부분에 적합하다. 한정된 지도를 화면에 상세히 보여주기 위해 많은 비용을 들여서 지도를 새롭게 그릴 필요가 없고, 기존에 만들어져 있는 지도를 쉽게 비용없이 읽어 들일 수 있기 때문이다. 즉 한 지역내의 순찰차 위치, 상하수도 관리, 교통 신호 상태, 전화선, 화재의 위치 등에 적합할 것이다. 반면에 Vectorizing방법은 넓은 지역과 움직임이 극심하며, 정보의 변화가 빠르고, 동일한 정보의 사용자가 많은 경우에 적합하다. 이에는 자동차의 주행안내, 국토관리, 군사관계등에 사용이 가능하다.

연구자는 Raster방식으로 이 시스템을 구현해 본 결과 자료(Data)량이 너무 많아져 비효율적이라 판단하였다. 그래서 이를 Vetorizing방법으로 새롭게 구현했다.

2.1.3. 계층화된 화면 정보 관리

주행 안내 시스템은 소형화, 휴대용을 목적으로 하므로 무한정한 메모리(Memory)를 사용하는 것은 비용면이나 소형화 의도에도 적당하지 않다. 한정된 메모리로 많은 정보를 유지 관리하기 위하여 화면 정보를 모두 계층적으로 나누어 관리했다.

계층화된 정보 관리란 정보의 중요도에 따라서 loading과 displaying을 결정하는 방법이다. loading은 데이타베이스 화일에서 Memory Buffer로 정보를 가져 오는 것을 의미하고 displaying은 정보를 Memory Buffer에서 화면으로 보내는 것을 의미한다. 중요도가 높다는 의미는 축척이 작아져도 지도에 나타나야 한다는 뜻이고 낮다는 의미는 대축척 지도에서도 나타날 필요가 적다는 뜻이다. 그러므로 loading과 displaying이 필요하지 않게 된다.

계층화된 정보 관리를 구현하기 위하여 이 시스템에서 Memory Buffer를 사용한다. 이는 계속되는 메모리 loading 시간을 감소시켜 주고, 정보가 겹쳐 보이는 것을 방지할 수가 있으며, 보는 사람의 혼잡도 막을 수 있다.

Memory Buffer를 효율적으로 사용하려면 매우 큰 데이타베이스에서 작은 Memory Buffer에 일정량의

정보를 가지고 오는 방법이 필요하다. 그 방법의 하나로 축척과 정보의 Priority를 사용하였다.

각 정보는 우선 나타내고자 하는 현 축척에 따라 loading해야 할 정보를 결정해야 한다. 식에서 보듯이 정보 priority가 현 축척에 따른 최소 priority a보다 크고 최대 priority b보다 작으면 loading을 한다. 지도가 변화함에 따라서 축척과 화면 좌표가 변하게 된다. 변경된 좌표 u1, v1, u2, v2라 하고, 변경된 축척을 k'이라 하자. 그때 k'이 최소 priority a와 최대 priority b 사이에 있고, 현재 화면이 memory buffer 화면 안에 존재하면 memory buffer는 현 상태를 유지하고, 그렇지 않으면 새로운 정보를 메모리에 loading한다.

f(축척) => k 현 축척에 따른 Priority
min a (최소 Priority)
max b (최대 Priority)
좌측 상단 (x_1, y_1)
우측 하단 (x_2, y_2)

$a \le data \le b$ 라면 *Loading* 한다.

지도의 움직임에 따라 변하는 요소 (f'(축척))
현 *Screen*(u_1, v_1, u_2, v_2), 현 Priority k'

$a \le k' \le b$ 이고,
$x_1 \le u_1$, $u_2 \le x_2$ 이고,
$y_1 \le v_1$, $v_2 \le y_2$ 이면 *Loading* 하지 않는다.

f(축척) : 현재 *Loading*이 이루어 지면서 변하게 되는 *Block*의 좌표와 Priority 의 값을 의미함

f'(축척) : 현재 지도가 움직이면서 변하게 되는 *Block*의 좌표와 Priority 의 값을 의미함

< 축척과 정보의 Priority를 이용한 화면 정보 관리

displaying은 실제 화면에 너무 많은 정보가 보여서 겹칠 수 있다. 이런 현상을 방지하기 위해서 그 정보의 중요도에 따라 priority를 가지고 있으면서 현재 지도의 축척에 따라 displaying이 될 정보를 결정하게 된다.

2.1.4. 파일 정보 관리 방법

파일 정보 관리는 Memory Buffer에 정보를 탐색하는 부분과 화일에서 원하는 목적지 또는 정보를 찾는 부분이 존재한다.

Memory Buffer에 존재하는 정보는 화면에 displaying을 주목적으로 한다. Displaying 여부를 판단하기 위하여 모든 정보를 거의 한 번씩은 검색하여야 한다. 그러므로 순차적으로 정보에 접근하는 것이 가장 유리하다. 반면에 화일에 존재하는 정보에서 원하는 목적지 또는 정보를 찾는 것은 key값을 탐색하는 것이 목적이다. 모든 정보를 메모리로 가져와서 검색하기에는 정보가 너무 많고 화일에서 직접 찾으면 시간이 오래 걸린다. 그러므로, 정보를 메모리에 loading하지 않고 index file을 구성하여 index file에서 key를 탐색, 정보의 화일 위치를 알아서 직접 화일에 접근하게 하는 방식을 취하였다. index file의 구조는 B tree로 구현하였다.[3)]

2.2. 최적 경로 알고리즘

최적 경로를 구하기 위해서 교차로를 node로 하고 교차로 사이의 길을 arc로 하여 풀게 된다. 각 arc의 비용은 본 연구에서는 절대 거리로 구하였으나 교통정보를 수신하여 시간거리로 계산됨이 바람직하다. 도로상에서 차량이 움직인 절대거리보다 시간거리로 움직이는 것이 경제적인 면이나 환경적인 면에서 보다 이익이기 때문이다.

또한 도로에서 반대로 돌아가는 길은 존재하기가 희박하고 목적지의 거리에서 멀어져 가는 일은 최적 경로가 되기 어렵다. 그래서 목적지와 반대로 가게 되는 방향을 고려하지 않는다.

그리고 매우 먼 거리의 길을 탐색하려면 정보량의 과다로 인하여 메모리가 모자라게 되므로 이를 나누어 두 지점간의 주요 도로로 경로를 찾고 양단의 끝까지 가는 경로를 탐색하여 길을 찾도록 한다. 예를 들면, 서울 시청에서 대전 시청까지 가기 위하여 서울과 대전사이의 모든 정보를 탐색하여 볼 필요는 없다. 서울과 대전 사이의 고속도로와 국도를 먼저 찾아서 결정한 후에 서울의 시청에서 고속도로나 국도로 나가는 길까지를 따로 찾고, 대전에 들어가는 길에서 대전시청까지 가는 길을 따로 찾아 가는 것이다.

이 네트워크(Network)는 arc의 비용이 음이 아닌 값으로만 이루어져 있고 목적지까지의 경로만 찾으

면 된다. 이 특성을 이용하여 Dijkstra 알고리즘[4] 을 변형, 탐색속도를 향상시켰다.

그 변형된 식은 다음과 같다.

(a) *INIT*

$\overline{S}=\{2, ..., N\}$, $\Pi(1)=0$, $\Pi(i)=\begin{cases} l_{1i} & \text{if } i\in\Gamma_1, \\ \infty & \text{otherwise.} \end{cases}$

(b) *SEARCH* $j\in\overline{S}$ *such that* $\Pi(j)=\min_{i\in\overline{S}}\Pi(i)$.

Set $\overline{S} \leftarrow \overline{S} - \{j\}$

If $|\overline{S}| = 0$, *END*; *otherwise goto* (c)

(c) *For all* $i \in \Gamma_j$ and $i \in \overline{S}$, *set*

$\Pi(i) \leftarrow \min(\Pi(i), \Pi(j) + l_{ji})$

goto (b)

< 정의 >

Set $X = \{1, ..., N\}$.

Let l_{ij} *be the length of the arc* (i, j) if $(i, j)\in U$

$\Pi(i)$ = (1) *If* $i\in S$, *then* $\Pi(i) = \Pi^*(i)$

(2) *If* $i \in S$, *then*

$\Pi(i) = \min_{k\in S,\, k\in\Gamma_i} (\Pi(k) + l_{ij})$

< Dijkstra의 알고리즘 >

(a) *INIT*

Let Destination k

$\overline{S}=\{2, ..., N\}$, $\Pi(1)=0$, $\Pi(i)=\begin{cases} l_{1i} & \text{if } i\in\Gamma_1, \\ \infty & \text{otherwise.} \end{cases}$

(b) *SEARCH* $j \in \overline{S}$ *such that* $\Pi(j) = \min_{i\in\overline{S}} \Pi(i)$.

Set $\overline{S} \leftarrow \overline{S} - \{j\}$

If $\{ |\overline{S}| = 0$ *OR* $\{ k \in \overline{S}$ *AND* $\Pi(k)\le\Pi(j) \}$

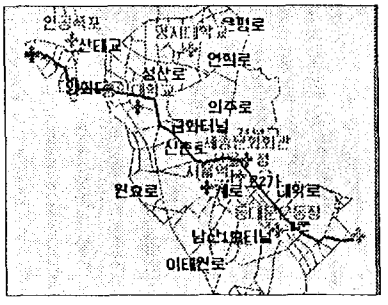
END; *otherwise goto* (c)

< 정의 >

k is destination node.

< 알고리즘을 개선한 a, b >

<그림 4>은 목동에서 고려 대학교까지의 최적 경로를 나타내고 있다.



<그림 4> 목동에서 고려대학교까지의 최단 경로

2.3. 다양한 정보 안내

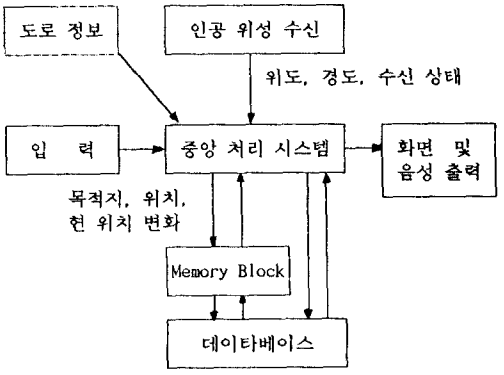
목적지를 찾아 가는 길을 안내하는 일 외에도 가까운 주유소나 숙박 시설을 안내해 준다. 이는 원하는 정보를 지도 상에 입력하여 주면 언제나 사용이 가능하므로 병원이나 소방서와 같은 다른 다양한 정보도 이용할 수 있게 할 수 있다.

사용자의 위치를 지도 상에 십자 모양으로 표시하여 주는 것 외에도 주행 중에 운전자의 편의를 위하여 움직이는 방향이 언제나 위를 향하도록 지도가 회전하게 되면 방향 감각을 잃지 않도록 북쪽 방향을 언제나 가리키게 된다. 또한 현재 위치를 위도와 경도로도 볼 수가 있으며, 하단에는 지도 축척에 따라 축척 자가 적당한 크기로 변하며 보여 준다.

2.4. 목적지와 정보 탐색에 필요한 입력 방법

본 연구에서는 키보드를 사용하여 입력방식을 사용하고 있다. 그러나, 운전 중에 흔들리는 차 안에서 올바르게 입력을 하기는 좀처럼 쉽지는 않을 것이다. 이를 음성을 사용하여 입력하는 것이 이상적으로 보인다. 그러나 현 단계로 모든 사람의 소리를 거의 높은 신뢰도로 인식이 어려운 상황이므로 이에 대한 대안으로 Touch Screen을 사용하여 불편한 키보드사용을 피할 수도 있겠다.

3. 주행 안내 시스템 구성



주행 안내 시스템은 GPS(Global Positioning System)를 이용하여 차량의 현재 위치를 인공 위성으로부터 수신 받아 화면의 지도상에 이를 표시하고 입력된 목적지까지의 주행 경로를 계획하여 이를 화면에 표시함과 동시에 음성으로 안내하도록 되어 있다.

이 시스템은 세부분의 입력을 가진다. 인공 위성으로부터는 현재의 절대 위치를 수신하고, 중앙 관제 센터에서는 도로의 상황을 수신하며, 사용자로부터는 원하는 목적지 등 필요한 정보를 입력받는다.

GPS로 인공 위성에서 수신한 현재의 위도, 경도, 수신 상태는 현재의 위치를 판단하고 Memory Block의 변화를 통제 하게 된다.

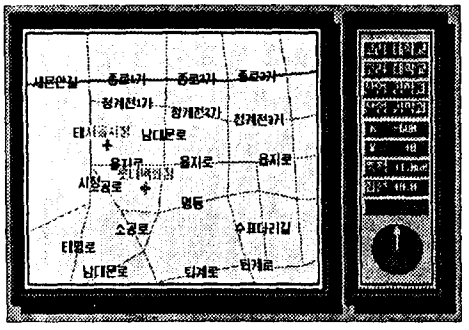
사용자 입력은 원하는 목적지를 데이타베이스에서 찾아 경로를 계획하고 이를 음성과 화면으로 알려 주기 위해 필요하다.

도로 정보 즉 각 도로의 교통 체증 현상과 도로의 날씨 등을 교통 관제 센터와 같은 곳에서 종합 분석하여 보내 주게 된다. 이를 수신하면 경로 설정에 이용할 수 있다.

실제로 구현한 주행 안내 시스템이 <그림 5>에서 서울시청 주위의 지도를 보여 주고 있다.

4. 결론

주행 안내 시스템이 실용, 보편화 되면 자동차 운전자의 편의를 도모할 뿐만 아니라, 교통 분산화



<그림 5> 주행 안내 시스템의 실제

를 촉진하여 교통체증을 덜어 주는 효과를 기대할 수 있다.

특히 효과를 극대화하기 위하여 교통정보를 중앙에서 집중하여 각각의 안내 시스템에 알리는 방법을 연구하고 현실화하여야 한다. 여기서는 통신 방법과 통신 수단이 중요한 문제로 대두하게 되며, 정보 수집 방법 또한 중요하다. 통신은 기존의 방송 주파수 사용이 가능하며, 정보 수집은 카메라, 속도 Sensor 등 여러 방법이 이용될 수 있다.

이 시스템에서 해결해야 할 또 하나의 문제는 GPS의 오차이다. 애초에 군사용으로 만들어진 것을 민간용으로 사용할 수 있도록 오차를 고의로 발생시켰다. 이는 다른 용도로 악용을 방지하고자 하는 목적때문이다. 그러므로, 이 오차를 Differential 시스템을 사용하여 오차를 줄이거나 주행 안내 시스템 용도에 알맞도록 통제하는 방안이 나와야 할 것이다.

< 참 고 문 헌 >

1 _____, *SIXGUN™ MODEL 610/620 Series DGPS RECEIVERS*, Motorola

2 _____, *KODEN Operation Manual*, KODEN ELETRONICS CO., LTD.

3 Michael J. Folk and Bill Zoellick, File Structures, Addison_Wesley, 1992

4. M. Gordon and M. Minoux, *Graphs and Algorithms*, wiley-interscience, 1984.