'91 韓國自動制御學術會議論文集 1991. 10.22~24

CAD-MAP의 개발 및 무인차량의 위치 식별

강 재 관　　서 석 환　　지 우 석

포항공과대학교 산업공학과

Development of CAD-MAP and Position Identification for ALV

Jae-Kwan Kang　　Suk-Hwan Suh　　Woo-Suk Jee

Department of Industrial Engineering

POSTECH, Pohang-Korea, 790-600

요　　약

무인차량(ALV)이 옥외 주행시 자신의 위치를 알아내는 방법으로서 카메라를 통한 입력 영상과 자신이 보유하고 있는 지도 정보를 이용하는 방법에 대하여 연구하였다. 이를 위하여 본 연구에서는 지도정보를 컴퓨터에 효율적으로 표현하는 방법과 이 정보를 이용하여 필요한 투시도를 생성 시켜주는 "CAD-MAP" 시스템을 구현하였고 이를 이용하여 ALV의 위치를 계산하는 방법을 개발하였다. 개발된 방법의 유효성은 컴퓨터 모의실험을 통하여 검증해 보였고, 실제 영상에 적용할때 영상이 가지는 여러가지 노이즈를 감안 위치 계산의 정확도를 높일 수 있는 방법에 대해서도 연구하였다.

1 서　　론

무인차량(Autonomous Land Vehicle, ALV)이란 주로 옥내 주행에 사용되는 AGV (Autonomous Guided Vehicle)와는 달리 옥외(시가지, 산악지형)에서 주행하는 이동 로보트를 말한다. 이러한 ALV는 위험지대에서 운행되는 차량, 군사적 용도의 차량등에 사용될 목적으로 현재 많은 연구가 행해지고 있다.

ALV가 목적지까지 주행하는 임무를 정확하게 수행하기 위해서는 주행 도중에 자신의 위치를 확인하는 과정이 필요하다. 일반적으로 ALV의 자기위치 확인을 위하여 사용하는 방법으로는 크게 표식장치(Land Mark)를 이용하는 방법[1], 추측 항법(Dead Reconing)을 사용하는 방법[2], 인공위성을 이용하는 방법등 여러가지가 있다. 그러나 이러한 방법들은 대부분 ALV 자체 장비외에 부가적인 장치들이 필요하다는 문제점을 안고 있다.

그리하여 본 연구에서는 이러한 문제점을 해결하기 위하여 ALV 의 자기위치 식별을 위한 새로운 한 방법으로 ALV가 주행하고 있는 도중 카메라를 통하여 입력되는 주위 산들의 윤곽선과 자신이 보유하고 있는 지도(MAP)로 부터 얻을 수 있는 지형의 윤곽선을 비교함으로써 자신의 위치를 계산해 내는 방법을 제안 하였다. 이 방법은 기존의 방법에 비해 별도로 외부로부터의 도움이나 부가적인 장치들을 필요로 하지 않는다는 점에서 좀더 자율적이라 할수있고 우리나라와 같이 산악지형이 많은 환경에서는 여러가지 장점을 가진다고 말할수 있다.

본 연구에서는 지도정보를 컴퓨터에 표현하고 이로부터 투시된 지형을 생성하고 이것과 카메라의 영상이미지를 비교하여 자신의 위치를 계산하는 순서로 제시된 방법을 구현하였다. 즉, ALV가 지도를 이용하기 위해서는 우선 지도정보를 컴퓨터에 효율적으로 표현하는 방법과 이정보를 이용하여 주어진 Veiw Information에 해당하는 투시도를 빠르게 생성시켜주는 방법이 필요하다. 이것을 본 연구에서는 "CAD-MAP"이라 부르고, 이를 구현하기 위하여 지도 정보의 입력 방법과 보간 방법 그리고 신속한 투시도 생성방법 등에 관하여 연구하였다.

그리고 ALV의 위치계산은 카메라를 통하여 들어온 지형의 영상과 View Information, 그리고 CAD-MAP에서 제공하는 투시도를 이용하여 계산하는 방법을 개발하였고, 계산된 결과를 검증하는 방법과 입력 영상이 가지는 노이즈와 오차를 고려해 위치의 정확도를 향상시킬수 있는 방법에 대해서도 연구하였다.

본 연구는 무인화 운행의 한 부분인 GPS(Global Planning System)를 위해 개발되었으며, 컴퓨터 모의 실험을 통해 이론의 타당성을 증명해 보였다.

2. CAD-MAP의 개발

CAD-MAP이란 지도정보를 컴퓨터에 표현하고, 표현된 정보를 이용하여 필요한 지형의 투시도를 생성해 주는 시스템을 말한다.

2.1 지도정보의 표현

본 연구에서는 지도상의 등고선 정보를 입력하는 방법으로서 디지타이저를 사용하였다. 즉 등고선을 따라 일정 간격으로 점을 샘플링하여 입력시키면 Cubic Spline Interpolation 방법에 의하여 등고선 상의 점들이 생성되게 된다. 이 때 등고선의 고도값은 따로 키보드를 통하여 입력 한다.

등고선 정보를 입력 시킨 후에는 지도를 일정간격으로 나누어 격자 모양이 되게 한후(그림 1), 임의의 격자를 등고선이 통과할 경우에는 그 등고선의 고도값을 그 격자의 대표값으로 삼고 그렇지 않을 경우에는 다음의 Interpolation식에 의하여 대표값을 계산함으로써 지도정보를 격자 데이타로 표현한다.

$$Cell[i]=\sum_{k=1}^{4} Ratio[k]*Height[k]$$

여기서 k는 현재 Cell의 위치에서 위,아래,좌,우 방향을 나타내는 인덱스이고, Height[k]는 현재 Cell에서 위,아래,좌,우 방향으로 Search하여 처음으로 고도값을 가지는 Cell의 값을 의미한다. 그리고 Ratio[k]는 현재 Cell의 위치에서 위,아래,좌,우 방향으로 처음으로 고도값을 가지는 Cell까지의 거리(distance[i])에 반비례하는 비율로서 다음과 같이 계산한다.

$$Ratio[i]=(1/distance[i])/\sum_{k=1}^{4}(1/distance[k])$$

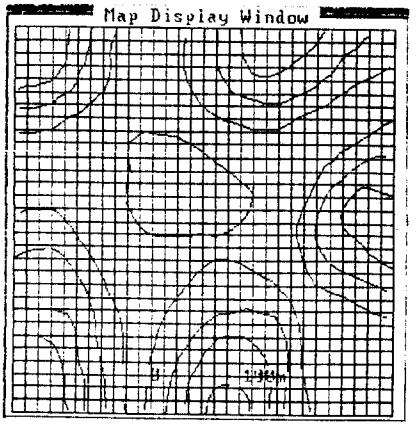


그림1. 격자형태로 나뉘어진 지도

그러나 이렇게 격자 데이타로 지도를 표현하는 것과 더불어 넓은 지역의 지도를 효율적으로 표현하기 위해서는 지도를 일정한 크기로 분할하여 분할된 단위별로 지도를 표현하고 관리하는 것이 필요하다.

본 연구에서는 지도를 단위 구역별로 나누어 각 구역별로 앞서 설명한 방법에 의하여 격자 데이타 화일을 생성시키고, 또 그 구역에 대하여 산 봉우리들의 위치 데이타만을 저장시킨 화일이 따로 생성 되도록 함으로써 넓은 지역의 지도를 효율적으로 관리 할수 있도록 하였다.

산 봉우리 위치 데이타 화일을 격자 데이타 화일과 별도로 따로 생성시키는 이유는 이것이 나중의 투시도 생성과 ALV의 위치 계산시에 매우 중요하게 사용되기 때문이다.

지금까지 설명한 지도정보를 표현하는 과정이 그림 2 에 나타나 있다.

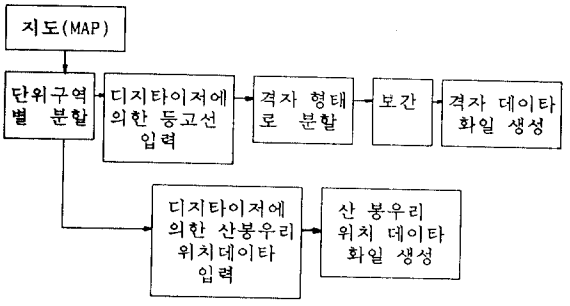


그림2. 지도 정보 표현 과정

2.2 투시도(Perspective View) 생성.

투시도는 앞절에서 설명한 격자 데이타 정보와 ALV의 View Information에 의해 생성된다. ALV의 View Information 에는 ALV의 위치좌표 x,y,z값과 카메라의 팬각과 틸트각 있다.

카메라의 팬각은 카메라의 시축(Optical axis)의 xy 평면에의 정사영이 기준 상태의 x축으로 부터 반시계 방향으로 회전된 각도로 정의 하였고, 틸트각은 카메라 시축과 정사영이 이루는 각도로 정의 하였다

이러한 정보가 주어지면 View transformation식에 의해서 각 격자의 대표값들이 변환되고, 변환된 점들을 이웃한 점들끼리 연결시켜 사각형 Patch를 만들어서 투시도가 생성되도록 하였다.

투시도를 생성 시키는 데에서 중요한 사항은 투시도가 ALV의 위치계산을 위해 사용되므로 실제 지형을 정확히 묘사하는 것과 빨리 투시도를 생성 시켜주는 것이 필요하다고 볼 수있다.

그리하여 본 연구에서는 빠른 투시도 생성을 위하여 지도상의 모든 격자를 다 변환 시키지 않고 View Volume 내에 포함되는 격자 데이터만을 변환 시킴으로써 투시도가 빠르게 생성되도록 하였다. 지도상의 임의의 격자가 View Volume내에 포함 되는지의 여부를 체크하는 방법은 아래와 같다.

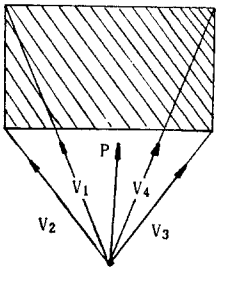


그림3. 격자점의 Visibility 체크

Vi : view volume을 구성하는 네개의 벡터 i=1,2,3,4
ni : view volume을 구성하는 네개의 평면에서의 normal 벡터. i=1,2,3,4
P : 카메라 원점에서 임의의 격자까지의 벡터.

만약 벡터 P와 네개의 normal 벡터 n 의 내적이 모두 음수이면 벡터 P는 view volume내에 포함된다.

3. 위치 계산 (Position Identification, PI)

ALV의 위치는 기본적으로 CAD_MAP의 정보와 ALV의 카메라를 통해서 들어 온 영상정보와 그때의 카메라의 팬각과 틸트각, 그리고 고도계를 통하여 입력된 ALV의 고도정보를 이용하여 계산한다.

예를 들어 ALV가 이미 알고 있는 산의 정상을 향해 그림 3-(a)와 같이 위치해 있다고 하자. 이 경우에 산으로 부터 ALV가 떨어져 있는 거리는 다음 식에 의하여 간단히 유도된다.

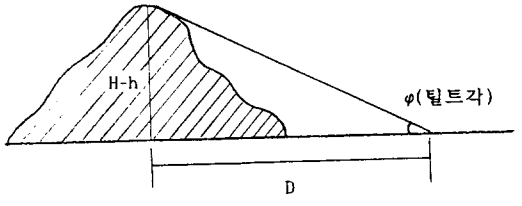
$$D = (H-h)/\tan(\phi) \qquad (1)$$

D : 산에서 ALV 까지의 거리
H : 산의 고도
h : ALV의 고도
ϕ : 카메라의 틸트각

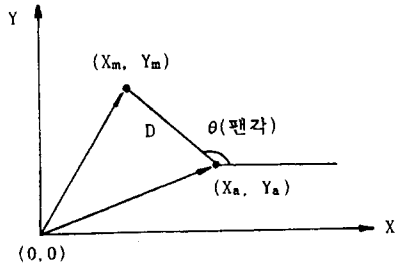
산으로부터 ALV까지 떨어진 거리가 계산되면, 지도상에서 ALV의 위치는 다음과 같이 계산된다 (그림 3-(b)).

$$Xa = Xm - D\cos(\theta) \qquad (2)$$
$$Ya = Ym - D\sin(\theta)$$

(Xa, Ya) : ALV의 위치 좌표
(Xm, Ym) : 산의 위치 좌표
θ : 카메라의 팬각



(a)



(b)

그림4. ALV의 위치 계산

그런데 여기서 카메라의 팬각과 틸트각 그리고 ALV의 고도를 안다고 가정한 이유는 이들 정보가 하드웨어 장비들로부터 얻을 수 있는 것들이기 때문이다. 즉 카메라의 팬각은 나침판으로부터 측정 가능하고, 틸트각 역시 카메라를 회전시키는 모터에 보내진 펄스의 누적 값에서 얻든지 그외 정밀한 측정 전용 장치의 개발이 가능하기 때문이다.

그러나 고도값은 고도를 잴 수 있는 특별한 장치가 없기 때문에 다음과 같은 방법으로 구한다. 즉 ALV는 항상 지상에 있으므로 CAD-MAP의 데이타를 이용하여 산 정상으로부터 지면이 틸트각을 이루도록 직선을 그어 내려 지면과 만나는 점을 적절한 보간을 통하여 구하고 그때의 고도값을 계산해 내면 된다.

그러나 이러한 모든 정보를 완벽하게 얻을수 있다하더라도 이 식은 카메라가 정확히 산의 정상을 향해 있을 경우를 가정한 것이므로 만약 카메라의 시축이 산의 정상과 일치하지 않는 경우에는 적용 할 수가 없게 된다. 그 뿐만아니라 이 식은 영상에 잡힌 산 봉우리가 지도상의 어느 산에 해당하는지를 미리 알고 있음을 전제로 하기 때문에 만약 지도상에 산봉우리가 여러개 있어 해당 산이 어느 것인지를 모르는 경우에는 또 문제가 발생한다.

이러한 문제를 해결하기 위해 본 연구에서는 카메라의 시축이 산의 정상을 향하고 있지 않은 경우에 카메라를 회전하지 않고 ALV의 위치를 계산할수 있는 방법과, 산이 여러개 있을 경우에도 해당되는 산을 찾아내는 방법을 개발하였다.

3.1 카메라의 시축이 산의 정상을 향하지 않은 경우

카메라를 회전시키는 것은 View transformation으로 설명하면 카메라 좌표계에서 좌표축을 회전 시키는 것과 동일하다. 즉 그림 5 와 같이 카메라 좌표계를 설정하면 카메라의 팬각이란 눈 좌표계에서 y 축을 반시계 방향으로 회전 시킨 양에 해당하고, 틸트각은 x 축을 반시계 방향으로 회전 시킨 양에 해당한다.

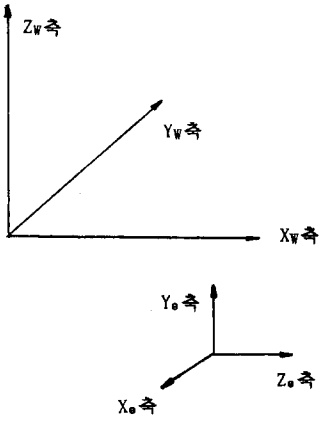


그림5. 카메라 좌표계와 월드 좌표계의 관계

예를 들어 카메라를 기준상태에서 θ^0 , ϕ^0 만큼 회전시켰을때 잡힌 영상이 그림 6 과 같다고 하자. 그림 6은 카메라의 시축이 정확히 산의 정상을 향하고 있지 않는 예이다. 이 때 화면 상에 나타난 산 봉우리의 좌표값을 X_s, Y_s 라 하면 카메라가 이 산의 봉우리를 정확히 향하기 위하여 지금의 상태에서 더 회전 해야 할 팬각($\Delta\theta$)과 틸트각 ($\Delta\phi$)은 다음과 같은 방법으로 계산할수 있다.

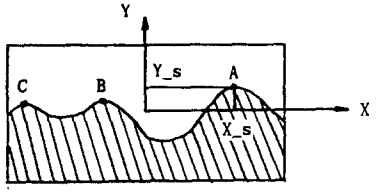


그림6. 카메라에 잡힌 산의 영상

즉 현 상태에서 산봉우리의 좌표를 카메라 좌표계로 표현했을 때 그 값을 (Xe^0, Ye^0, Ze^0) 라 하면 카메라를 $\Delta\theta, \Delta\phi$ 만큼 더 회전 하였을때 산 봉우리의 카메라 좌표계 값(Xe^*, Ye^*, Ze^*)은 다음의 View transformation에 의하여 변환되어 식 (3)과 같이 된다.

$$(Xe^*, Ye^*, Ze^*) = (Xe^0, Ye^0, Ze^0)\cdot T_1\cdot T_2\cdot T_3\cdot T_4 \qquad (3)$$

$$T_1 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_2 = \begin{bmatrix} \cos\Delta\theta & 0 & \sin\Delta\theta \\ 0 & 1 & 0 \\ -\sin\Delta\theta & 0 & \cos\Delta\theta \end{bmatrix}$$

$$T_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\Delta\phi & \sin\Delta\phi \\ 0 & -\sin\Delta\phi & \cos\Delta\phi \end{bmatrix}$$

$$T_4 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

여기서 T_1은 왼손 좌표계를 오른손 좌표계로 변환 시키는 변환 행렬이고, T_2는 카메라를 팬각 $\Delta\theta$ 만큼 회전 시켰을 때의 변환 행렬이다. T_3는 카메라를 틸트각 $\Delta\phi$ 만큼 들어 올렸을때에 해당하는 변환 행렬이고 T_4는 오른손 좌표계를 다시 왼손 좌표계로 변환 시키는 변환 행렬이다.

식 (3)을 풀면

$$\begin{aligned} Xe^* &= Xe^0\cos\Delta\theta + Ze^0\sin\Delta\theta \\ Ye^* &= Xe^0\sin\Delta\theta\sin\Delta\phi + Ye^0\cos\Delta\phi \\ &\quad - Ze^0\cos\Delta\theta\sin\Delta\phi \\ Ze^* &= Xe^0\sin\Delta\theta\cos\Delta\phi + Ye^0\sin\Delta\phi \\ &\quad + Ze^0\cos\Delta\theta\cos\Delta\phi \end{aligned}$$

이 되는데 이것을 다시 화면 좌표계 값으로 변환 하면

$$\begin{aligned} Xs^* &= Xs^0\cos\Delta\theta - d\sin\Delta\theta \\ Ys^* &= -(Xs^0\sin\Delta\theta + d\cos\Delta\theta)\sin\Delta\phi \\ &\quad + Ys^0\cos\Delta\phi \end{aligned} \qquad (4)$$

식 (4) 와 같이 된다. 여기서 d 는 카메라의 촛점거리를 나타낸다. 그런데 카메라의 시축이 정확히 산 봉우리를 향하기 위해서는 Xs, Ys 값이 둘 다 영이 될 때이므로 이로부터 $\Delta\theta, \Delta\phi$를 구하면

$$\Delta\theta = \tan(-Xs^*/d)$$

$$\Delta\phi = \tan(Ys^*/(d\cos\Delta\theta - Xs^*\sin\Delta\theta))$$

이 된다. 이것이 현 상태에서 더 회전 시켜야 할 팬각과 틸트각을 나타낸다.

이렇게 하여 구한 $\Delta\theta, \Delta\phi$ 와 초기 θ , ϕ 로 부터 식 (1),(2)를 이용하기 위해서는 카메라가 기준상태에서 한번에 산정상을 정확히 향하기 위한 팬각(θ^*)와 틸트각 (ϕ^*)를 구해야한다. 그런데 여기서 주의해야할 사항은 θ^*, θ^0, $\Delta\theta$ 와 $\phi^*, \phi^0, \Delta\phi$ 사이에는 일반적으로

$$\theta^* = \theta^0 + \Delta\theta$$
$$\phi^* = \phi^0 + \Delta\phi$$

의 관계가 성립하지 않는다는 점이다. 이는 View transfomation이 변환의 순서에 독립적이지 않기 때문이다.

그러므로 θ^* ϕ^*를 구하기 위해서는 우선 기준 상태에서 $\theta^0, \phi^0, \Delta\theta, \Delta\phi$ 순서로 카메라를 회전 하였을때 회전된 카메라 시축(Ze)상의 단위 벡터(0,0,1)가 월드 좌표계에서는 무슨 좌표값을 갖는지를 계산한 뒤 이 좌표값으로 부터 θ^* ϕ^*를 구하면 된다.

이를 위하여 우선 회전된 카메라 시축상의 단위 벡터에 해당하는 기준 상태의 카메라 좌표계에서의 벡터 Pe(Xe, Ye, Ze)를 먼저 구하면

$$(0,0,1) = Pe\cdot T_1\cdot T_2\cdot T_3\cdot T_4\cdot T_5\cdot T_6$$

$$T_1 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_2 = \begin{bmatrix} \cos\theta^0 & 0 & \sin\theta^0 \\ 0 & 1 & 0 \\ -\sin\theta^0 & 0 & \cos\theta^0 \end{bmatrix}$$

$$T_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi^0 & \sin\phi^0 \\ 0 & -\sin\phi^0 & \cos\phi^0 \end{bmatrix}$$

$$T_4 = \begin{bmatrix} \cos\Delta\theta & 0 & \sin\Delta\theta \\ 0 & 1 & 0 \\ -\sin\Delta\theta & 0 & \cos\Delta\theta \end{bmatrix}$$

$$T_5 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\Delta\phi & \sin\Delta\phi \\ 0 & -\sin\Delta\phi & \cos\Delta\phi \end{bmatrix}$$

$$T_6 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

가 되고, 이로부터 월드 좌표계로 표현된 좌표값 (Xw, Yw, Zw)는 그림 4에 나타난 기준 상태의 카메라 좌표계와 월드 좌표계의 관계로 부터 다음과 같이 구할 수 있다.

$$Xw = Ze$$
$$Yw = -Xe$$
$$Zw = Ye$$

이 좌표값은 회전된 카메라의 시축을 월드 좌표계에서 단위 벡터로 표현 한 것이므로 이것으로 부터

$$\theta^* = \tan(Yw/Xw) \qquad 0 < \theta^* < 2\pi$$
$$\phi^* = \sin(Zw) \qquad -\pi/2 < \phi^* < \pi/2$$

을 얻는다.

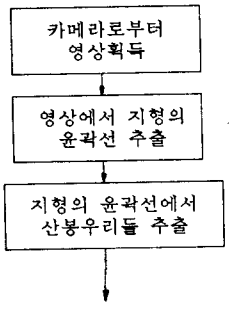
3.2 해당 산을 찾기위한 탐색

앞절에서 제시한 방법에 의하면 카메라의 시축이 산봉우리를 향하지 않더라도 정확히 산 봉우리를 향하게 하는 카메라의 팬각과 틸트각을 계산할수 있다.

그러나 앞서 말한 바와 같이 식 (1) 과 (2)는 영상에 잡힌 산 봉우리가 지도 상의 어떤산에 해당 하는지를 아는 경우에 한하여 ALV의 위치를 계산해 주므로 만약 지도상에 여러개의 산봉우리가 있는 경우에는 문제가 발생한다. 그리하여 본 연구에서는 이 문제를 해결하기 위하여 Hypothesis & Test 방법을 사용하였다.

즉 지도상의 임의의 산을 영상에 잡힌 산봉우리에 해당하는 산이라 가정하고 앞절에서 제시한 방법으로 위치계산을 한다. 그리고 계산된 위치에서 CAD-MAP을 통하여 지형의 윤곽선을 얻어 영상에서 얻은 산의 윤곽선과 비교한다. 이 차이가 허용범위 안에 들면 해당 산으로 판단하고 그 때의 위치값을 ALV의 현위치로 간주하고, 그렇지않은 경우에는 다른 산으로 가정을 옮겨서 테스트를 반복한다.

이 과정을 Flow Chart로 표현하면 그림 7과 같다.



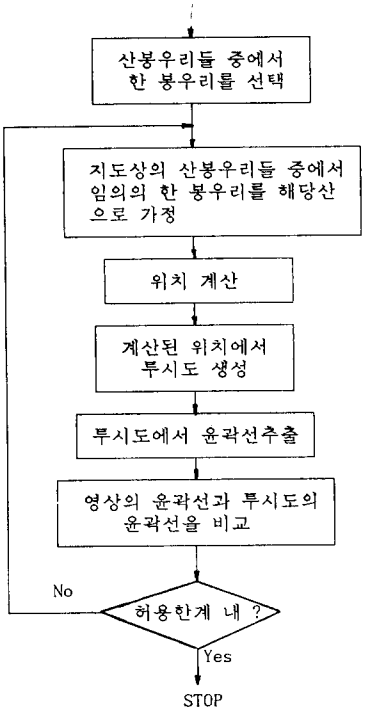


그림7. 해당산을 찾기위한 탐색과정

이 때 영상에서 얻은 지형의 윤곽선에서 산봉우리들을 추출하는 방법은 그림 8 에서와 같은 방법을 사용하였다. 즉 일정 간격으로 윤곽선상의 점을 샘플링한 뒤 기울기를 비교하여 기울기가 증가하다가 감소하는 구간에 대하여 다시 옆으로 Search하여 봉우리를 찾아내는 방법이다. 이 방법은 윤곽선상의 점들을 일정 간격으로 샘플링함으로써 산 윤곽선이 가지는 미세한 기울기의 변화를 무시할 수 있는 장점과 산봉우리를 Search 하는데 걸리는 시간을 단축시키는 효과를 얻을 수 있는 장점을 동시에 가진다고 볼 수 있다.

다음으로 이러한 방법에 의하여 추출된 산 봉우리들 중에서 어떤 산봉우리를 선택하느냐는 카메라의 중심에 가장 가까운 봉우리가 선택되도록 함으로써 카메라 렌즈의 왜곡에 따른 계산의 오차를 줄일수 있도록 하였다.

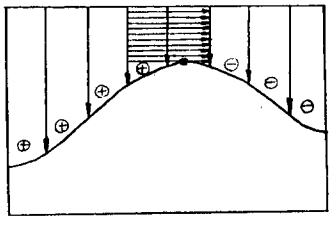


그림8 산봉우리를 찾는 방법

그리고 투시도에서 지형의 윤곽선을 추출 하는 방법으로는 그림 9 에서와 같이 CAD-MAP에서 제공한 투시도를 이용 위에서 부터 Search하는 방법으로 구하였고, 마지막으로 검증을 위해 두 윤곽선을 비교할 때 사용된 Measure는 두 윤곽선간의 차이값을 이용하였다 (그림 10).

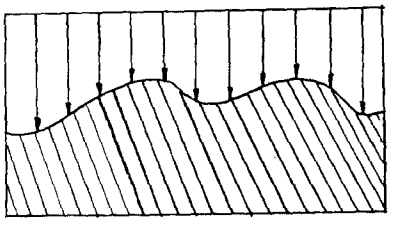


그림9. 투시도에서 지형의 윤곽선을 추출 하는 방법

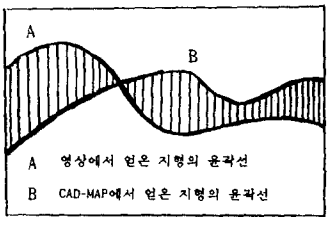


그림10 두 윤곽선의 비교

4. 시뮬레이션

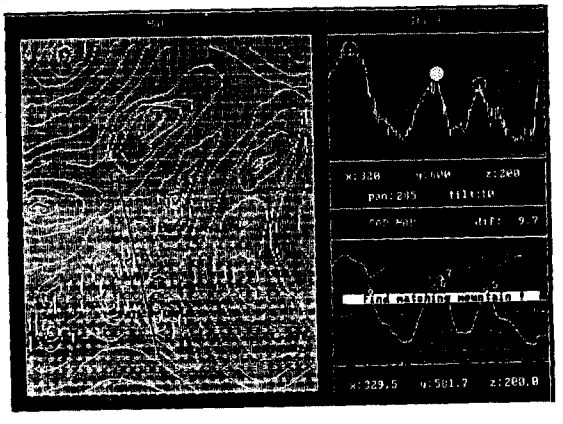
앞 절에서 개발한 알고리즘의 유효성을 검증하기 위하여 컴퓨터 모의실험을 행하였다. 모의 실험된 부분은 입력 영상 부분인데 이를 임의의 View Information에 해당하는 투시도로써 대신하였다. 이렇게 했을 경우 개발된 알고리즘은 Numerical Error를 제외하곤 정확히 자신의 위치를 계산해 내어 알고리즘의 유효성을 입증 하였다.

그러나 이는 어디까지나 모의 실험이므로 실제 지형에 실제 영상으로 실험할 경우에는 모의 실험과는 많은 부분에서 다를 것이다. 특히 카메라를 통하여 입력되는 영상은 많은 노이즈를 포함하여 모의 실험처럼 지형의 투시도와 영상이 똑 같아질 수가 없고, 또 앞서 이미 안다고 가정된 정보들 (카메라의 팬각, 틸트각) 역시 하드웨어 장비로 부터 얻는 것이기 때문에 반드시 오차를 가지게 되어, 이로 부터 계산된 위치값은 실제 위치에 비해 어느정도의 오차를 포함하게 된다.

본 연구에서는 이러한 사항을 고려, 오차를 줄이기 위한 두가지 방법을 제안 하였다. 첫번째 방법은 만약 그림 6과 같이 화면 상에 여러개의 봉우리가 있을 경우 임의의 한 봉우리 (A)는 초기 ALV의 위치 계산을 위해 사용하고, 나머지 봉우리들은(B,C)은 계산된 ALV 위치의 오차를 줄이기 사용하는 방법이다. 즉 임의의 한 봉우리를 선정하여 앞절에서 개발한 알고리즘에 의해 ALV의 위치계산을 하고나면, 카메라 영상과 CAD_MAP에서 얻은 투시도 간의 비교를 통하여 나머지 봉우리들에 해당되는 산들을 Match 시킬 수있으므로 식 (1)과 (2)를 사용하여 쉽게 ALV의 위치를 계산할 수 있다.

이렇게 각 봉우리에 대하여 ALV의 위치를 따로 따로 계산한뒤 평균을 내는 방식으로 오차를 줄일 수 있고, 두번째 방법으로는 현 위치에서 카메라가 향하고 있는 산이외의 다른 산을 향하도록 카메라를 차례로 돌려서 ALV의 위치를 반복 계산하는 방법을 사용할 수도 있다. 이 역시 현재 카메라가 잡고 있는 산이 지도에서 어느 산에 해당하는지를 알고 있으므로 다른 산을 영상으로 잡기위해 회전해야 할 카메라 각도를 쉽게 계산 할 수 있고, 이로 부터 ALV의 위치도 쉽게 계산된다.

본 연구에서는 이 방법들을 영상 이미지의 산 윤곽선에 노이지를 첨가하고, 카메라의 팬각과 틸트각에 조금의 오차를 삽입하여 모의실험하였다.



5. 결 론

본 연구에서는 ALV가 주행 도중 자신의 위치를 계산하는 새로운 방법으로써 카메라를 통한 입력 영상과 자신이 보유하고 있는 지도정보를 이용하는 방법에 대하여 연구하였다.

이를 위하여 디지타이저를 이용하여 지도의 등고선 정보를 쉽게 입력 시키는 방법과 이를 이용하여 격자 형태로 지도 정보를 표현하는 방법, 보간 방법, 그리고 Visibility 체크를 통한 신속한 투시도의 생성등의 방법을 개발 하였고 이것을 기반으로 카메라를 통하여 입력된 지형의 윤곽선과 View Information 을 이용하여 ALV의 위치를 정확히 계산해 내는 알고리즘을 개발 하였다.

개발된 방법의 유효성은 컴퓨터 모의실험을 통하여 검증해 보였고, 실제 영상에 적용할때 영상이 가지는 여러가지 노이즈를 감안 위치 계산의 정확도를 높일 수 있는 방법에 대해서도 아울러 연구하였다.

본 연구의 유효성이 컴퓨터 모의실험을 통해서만 증명되었기 때문에, 앞으로는 실제지형과 실제 영상에 대하여 실험하는 것이 필요하다.

6. 참고문헌

[1] L. Matthies and S. A. Shafer, " Error modelling in streo navigation," IEEE Journal of Robotics and Automation, vol 3, pp.239-248, June 1987.

[2] H.Nasr and B.Bhanu, " Landmark recognition system for autonomous mobile robots," in Proc IEEE Int. Conf. Robotics and Automation, pp. 1105-1111, 1988.

[3] C. S. Park, Interactive Microcomputer Graphics, Addison Wesley Publishing Co., 1985.

[4] Ray Talluri and J.K.Aggarwal, "Position Estimation for a Mobile Robot in an Unstructured Environment," IEEE International Workshop on Intelligent Robots and Systems, pp. 159-166, 1990.