'87 韓國自動制御學術會議論文集 1987. 10.16~17

시각센서를 이용한 로보트의 복잡한 곡선추적에 관한 알고리즘 .

권 태 상 김 경 기

한양대학교 전자공학과.

A algorithm on Robot tracking about complex curve with visual sensor.

tae-sang kwon kyung-ki kim

hangyang graduate school

ABSTRACT

In this thesis, we work on the curve recognition with real time processing and the Robot tracking method on recognized curve. Image information of segment curve is supplied to computer to run to a Robot ,so that it is a feedback system. Image coordinate frame to world coordinate transformation represents in this paper and curve matching algoritm subscribs by two method,first transformation matching algorithm ,second image coordinate matching algorithm .Also Robot running time to computer image processing time relationships finally includes.

I 서 론.

로보트를 효과적으로 이용하고 능률을 극대화 시키기 위하여 카메라에서 얻은 정보를 로보트에게 제공하는 로보트비전이 출현하였으며 또한 로보트hand에 카메라를 부착시켜 로보트를 제어하는 방법도 나왔으나 많은 연구가 진행되지 않았다. 카메라를 고정시키는 경우에는 범위의 한계성과 특정성때문에 많은 정보를 제공하지 못하고 있다.

본 논문에서는 로보트hand에 카메라를 부착하여 평면상의 임의 곡선을 추적하는 알고리즘과 수학적방법등을 제안하였다. 곡선을 추적하는 의미는 카메라를 임의의 장소에 고정시켜 한번에 전체곡선 혹은 상당한 부분을 인식하여 추적하는 것이 아니라 곡선의 일부분을 화상처리하고 그 값을 로보트에게 주며 다시 화상처리하여 처리결과를 연속적으로 제공하는 feedback방식을 의미한다. 이는 마치 사람의 손이 복잡한 곡선을 추적할때와 같은 것이나 화상처리시간 및 제어시간의 속도차이가 다르다. 이런 곡선추적 의미를 바탕으로 본 논문에서는 가능한 실시간처리가 되는 화상처리 알고리즘을 제안하며 물체와 곡선이 공존했을 때 곡선추적알고리즘 및 연속된 화상에서의 추적곡선matching방법을 소개한다. 또한 카메라좌표계와 base좌표계와의 거리, 면적, 중심위치의 변환을 유도하였다. 그리고 IBMPC/xt에서의 계산시간과 로보트의 주행시간의 관계를 연구하였다.
또한 본 논문에서는 시뮬레이션에 필요한 실질적 카메라 Calibration 및 Rhino-xr2 의 Kinematics와 대수적 Inverse Kinematics를 풀었다. 위 알고리즘을 시뮬레이션으로 보이겠다.

II. 곡선추출 알고리즘 및 물체와의 구별 방법 .

카메라에서 온 영상신호는 Digitizer 에서 sampling 하여 한 화상당 512X512 의 pixel data을 저장한다. 화상을 메디안 필터 및 enhancement 등의 전처리를 거쳐 처리하는 것은 상당한 시간이 소요되어 실기간처리가 불가능하다. 이런 이유로 본 논문에서는 중심이 (256,256) 이고 폭이 256 인 window내만 처리하고자 하는데 이는 처리시간의 단축 뿐만아니라 곡선을 카메라 중심부근에 오도록하는 효과가 있다.

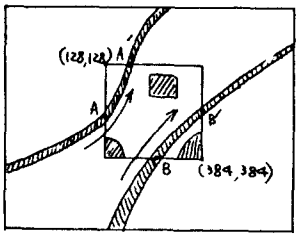


그림 1). 256*256 window 화상도

그림 1).에서 처럼 window내로 지나가는 곡선은 entering point A,B 와 outing point A',B' 가 반드시 존재한다. 이 window의 외곽을 (128,128) point 에서 시작하여 시계방향으로 scanning하는데 Edge detection operator를 사용한다.

시작점으로 부터 k 번째 pixel 에 대해

if flag = 0 and p(k) <= TH,

if g(k) <= TH ,

Edge detection operator ED(k)

g(k-2) >= (g(k),TH) > g(k+1)

g(k-1) >= (g(k),TH) > g(k+2)

g(k-1) >= (g(k),TH) > g(k+1)

if ED(k) = 1 ,

Left edge array ← k.

flag = 1.

if flag = 1 and p(k) >= TH,

if g(k) >= TH,

EDge detection operator ED(k)

g(k-2) < (g(k),TH) <= g(k+1)

g(k-1) < (g(k),TH) <= g(k+2)

g(k-1) < (g(k),TH) <= g(k+1)

if Ed(k) = 1,

Right edge array ← k.

flag = 0.

TH는 threshold 값 (주어짐).

ED(k) 는 조건을 만족하면 ED(k) 는 1 이됨.

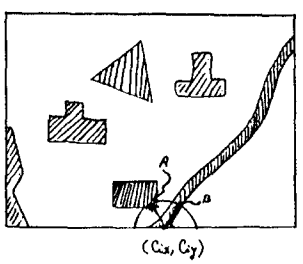
여기서 g(k)는 k 번째 pixel 의 메디안필터를 의미한다. 이렇게하여 window외곽 전체를 scanning하면 물체 및 곡선의 외곽 Edge pair 들을 얻을수있다. 이 Edge pair 들을바탕으로 물체와 곡선을 구별하기 위하여 i 번째 Edgepair를 (Li,Ri) 로 하고 그 크기를 Di, 중심위치를 Cix ,Ciy로 하면

$$Di = \sqrt{(Lix-Rix) + (Liy- Riy)}$$

$$Cix = \frac{Rix + Lix}{2} \quad , \quad Ciy = \frac{Riy + Liy}{2}$$

하고 외곽Edge pair 가 카메라의 Corner에있는 경우에는 외곽을 따른 중심을 Cix,Ciy 로 놓는다.

이 (Cix,Ciy) 를 중심으로 반경 Ri = $\frac{3}{2}$Di는 원을 설정하여 window내의 구간만 scanning하여 Circle Edge pair를 구한다.



(Cix, Ciy)

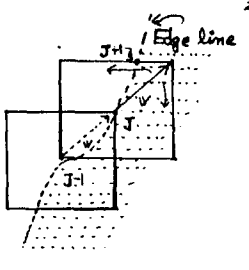


그림 2). 물체와 곡선의 구별도 그림 3). 7*7 Edge window

곡선의 두께를 Cth 로 주어지며 허용범위를 Cthmin과 Cthmax로 한다.

i) Circle Edge pair 갯수가 없을때

외곽 Edge pair 는 물체의 일부분.

ii) Circle Edge pair 갯수가 하나일때

Circle Edge pair 사이의 직선거리 Cth 가 Cthmin <= Cth <= Cthmax 이면 외곽는 곡선이고 아니면 물체.

iii) Circle Edge pair 갯수가 두개 이상일때

그림2). 에서처럼 외각Edge pair 중심에서 각 Circle Edge pair 중심사이의 직선내에 gray level 값이 TH 보다 큰값이 없고 Cthmin <= Cth <=Cthmax 이면 외곽 Edge pair는 곡선, TH 보다 큰 값이 있으면 물체.

만약 복수 곡선일때는 실거리 matching법을 사용한다.

이제 외곽Edge pair 중에서 window내부로의 entering point을 설정하여(설정방법은 뒤에) 초기값 Li,Ri 를 시작으로 곡점(곡선을 대표하는 점) 을 추출하는데 7X7 window를 사용한다. 마찬가지로 window의 외곽 24 개 pixel 만 검사하는데 그림3). 에서 j 번째 window의 중심을 (Jx,Jy) 하고 j-1 번째 window의 중심을, (J-1x, J-1y) 하면 그 위치의 이동벡터 W 는

$$W = (Jx-J-1x)\vec{a}_x + (Jy-J-1y)\vec{a}_y$$

가 된다.

i). 진행방향을 기준으로 오른쪽 Edge 경우.

if g(2Jx-J-1x,2Jy-J-1y) >= TH,

((2Jx-J-1x),(2Jy-J-1y)) 에서 시작하여 시계반대방향으로 scanning한다. m 번째에서 g(m)<=TH 이면 Edge detection operator 를 적용하여 만족하면 m 번째 pixel 좌표값을 (J+1x,J+1y) 로 놓음.

if g(2Jx-J-1x,2Jy-J-1y) <= TH,

((2Jx-J-1x,2Jy-J-1y) 에서 시작하여 시계방향으로 scanning한다. m 번째에서 g(m) >=TH이면 Edge Detection operator를 적용하여 만족하면 m 번째 pixel 좌표값을 (J+1x,J+1y) 로 놓음.

ii).진행방향을 기준으로 왼쪽 Edge일경우.

모든 절차가 오른쪽인 경우와 같으나 부등호가 반대임.

양쪽 window 좌표값이 일정한 거리로 증가하면서 곡점을 추출한다.

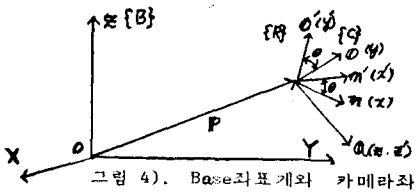


그림 4). Base좌표계와 카메라좌표계의 관계

III). 화상좌표계와 base좌표계의 관계.

카메라좌표계와 base좌표계의 관계를 그림 4). 에 나타내었다. 이를 위해 base좌표계의 X-Y 평면에 평행한 π'을 갖고 o'의 $\vec{a}_z > 0$ 인 reference 좌표계를 설정한다. $\therefore$ $\vec{o}' = o'_x\vec{a}_x + o'_y\vec{a}_y + o'_z\vec{a}_z$ $\pi' = N_x'a_x + N_y'a_y + N_z'a_z$

$\vec{a}' = A_x'a_x + A_y'a_y + A_z'a_z$.

즉 base좌표계의 Z 방향 unit 벡터를 a_z 라 하면

$$n' \cdot a_z = 0$$

$$o' \cdot a_z >= 0$$

따라서 reference 좌표계와 카메라좌표계의 변환은

$$ROT(Z,\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\theta = \cos^{-1}(\vec{o'} \cdot \vec{o})$$

$$= \cos^{-1}(\vec{n'} \cdot \vec{n})$$

$$o' = \cos(\alpha)\cos(\beta)a_x + \cos(\alpha)\sin(\beta)a_y + \sin(\alpha)a_z$$

$$\alpha = \cos^{-1}(A_z) \qquad \beta = \tan^{-1}\left(\frac{A_y}{A_x}\right)$$

되며 이렇게 변환된 값을 reference 좌표계와 base좌표계사이의 변환에 사용된다.

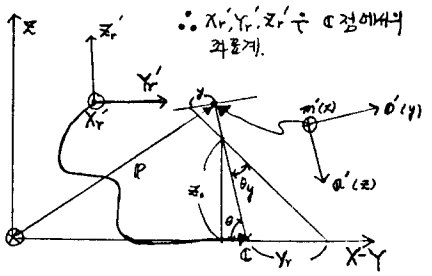
그림 5). 에서 $y'(o')$ 축에 대한 크기변환은

$$Yc = F(x',y') = \left| -\frac{z_o}{\tan(\theta - \theta_y)} + \frac{z_o}{\tan(\theta)} \right|$$

$x'(n')$ 축에 대한 크기변환은 $\therefore \theta_y = \tan^{-1}\left(\frac{y'}{f}\right)$

$$Xc = G(x',y') = \begin{cases} \left| \frac{-z_o \cdot \cos(\theta_y)}{\sin(\theta - \theta_y) \cdot f} \right|, & y \neq 0 \\ \left| \frac{-z_o \cdot x'}{f \cdot \sin(\theta)} \right| & y = 0 \end{cases}$$

여기서 $\theta = \sin(a_x \cdot a)$



$\therefore$ X_r', Y_r', Z_r' 은 C 점에서의 좌표계.

그림 5). 카메라좌표계와 Base좌표계의 변환

$x' = Nx \cdot m$; m 은 x' 축에 대한 pixel 갯수

$y' = Ny \cdot n$; n 은 y' 축에 대한 pixel 갯수

Nx,Ny = pixel 과 실제길이에 대한 비율.

따라서 임의의 position벡터 P 와 approach벡터 a 에 대한 C 점의 base좌표값은

$$Xc = P_x + f\cos\theta\cos\alpha + z_o\cot\theta\cos\alpha \quad \therefore \alpha = \tan^{-1}\left(\frac{A_y}{A_x}\right)$$

$$Yc = P_y + f\cos\theta\sin\alpha + z_o\cot\theta\sin\alpha$$

$$Zc = 0$$

이며 전체 base좌표계로의 좌표변환은

$$Xt = X_c + \sqrt{X_r^2 + Y_r^2}\cos(\theta_f + \alpha)$$

$$\therefore \theta_f = \tan^{-1}\left(\frac{X_r'}{Y_r'}\right)$$

$$Yt = Y_c + \sqrt{X_r^2 + Y_r^2}\sin(\theta_f + \alpha)$$

로 표현된다.

이제 면적 ,중심의 변환을 위해 X_f ,Y_f을 x',y' 에 대해 편미분 하면

$$\frac{\partial F}{\partial x'} = 0$$

$$\frac{\partial F}{\partial y'} = -\csc^2(\theta - \theta_y)\frac{f}{f^2 + y'^2}$$

$$\therefore \theta_y = \tan^{-1}\left(\frac{y'}{f}\right)$$

$$\frac{\partial G}{\partial x'} = \frac{z_o \cdot \cos(\theta_y)}{\sin(\theta - \theta_y) \cdot f}$$

$$\frac{\partial G}{\partial y'} = \frac{z_o \cdot x' \cdot \cos\theta}{f^2 + y'^2 \cdot \sin^2(\theta - \theta_y)}$$

이 되며 면적변환은 1 pixel 의 실거리가 각각 Nx,Ny 이므로

$$\Delta X_f(i,j) = \left.\frac{\partial F}{\partial y'}\right|_a \cdot Ny \qquad \Delta X_f(i,j) = \left.\frac{\partial G}{\partial x'}\right|_a \cdot Nx$$

$$\Delta S(x',y') = \Delta X_f \Delta Y_f = Nx \cdot Ny \cdot \left|\left.\frac{\partial F}{\partial y'}\right|_a\right| \cdot \left|\left.\frac{\partial G}{\partial x'}\right|_a\right|$$

그리고 화상에서의 면적은

$$\Delta Si(i,j) = Nx \cdot Ny$$

$$Si(i,j) = \sum_{S:(i,j)} \sum \Delta Si(i,j)$$

실제면적은

$$S(i,j) = \sum_{S:(i,j)} \sum \Delta X_f(i,j) \cdot \Delta Y_f(i,j)$$

으로 변환되고 X 축의 중심위치는

$$\Delta Mx = X_f'(i,j) \cdot \Delta X_f(i,j)$$

$$Mx = \sum_{S:(i,j)} \sum X_f'(i,j) \cdot \Delta X_f(i,j) \quad sumx = \sum_{S:(i,j)} \sum \Delta X_f(i,j)$$

$$Xm = \frac{\sum\sum X_r'(i,j) \cdot \Delta X_r(i,j)}{\sum\sum \Delta X_r(i,j)}$$

y 축의 중심위치는

$$\Delta My = Y_f'(i,j) \cdot \Delta Y_f(i,j)$$

$$My = \sum_{S:(i,j)} \sum Y_f'(i,j) \cdot \Delta Y_f(i,j) \quad sumy = \sum_{S:(i,j)} \sum \Delta Y_f(i,j)$$

$$Ym = \frac{\sum\sum Y_r'(i,j) \Delta Y_r(i,j)}{\sum\sum \Delta Y_r(i,j)}$$

으로 변환 된다.

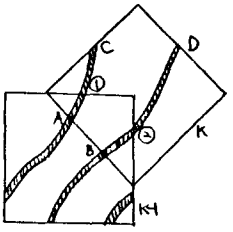


그림 6). 다중곡선 matching도

IV). 곡선 matching 알고리즘 및 카메라 align 방법

다중곡선이나 곡선같은 물체가 있을때 이를 구별하는 것은 조금 까다로운 편이다.
본 논문에서는 그림 6).에서 처럼 k 화상에서 외각 Edge Point A,B,C,D 에서 B Point 가 추적 곡선이라면 B Point 의 Base좌표값과 k-1 화상에서 얻은 Base좌표계 Point 와의 차이가 minimum 이 되는 것을 구한다.
즉 외곽 Edge pair A,B,C,D의 base좌표계점을 A',B',C',D' 라하면 (Cr은 k-1 화상의 추적곡선에 대한 Base좌표계의 곡점집합)

a = min $(\overline{A' Cr})$
b = min $(\overline{B' Cr})$
c = min $(\overline{C' Cr})$
d = min $(\overline{D' Cr})$

이라고 하면 min (a , b , c , d)을 선택하여 추적 곡선을 찾아내고 또 한가지는 k-1 화상의 1,2 점을 Base좌표계의 역 관계식에 의해 k 화상의 좌표계 값을 구하여 그 점으로 부터 곡선 Point 를 찾는다.
물론 카메라와 로보트의 기구학적인 Error 때문에 완전히 일치하지는 않을 것이나 반경 ΔR 내에서 조사하여 추적 곡선을 찾는다.
로보트 곡선을 추적할때 곡선의 다음 부분을 얻기위해 2 방향으로 회전 할 수 있는 장치를 갖춘다면 다음 식으로 카메라를 align 시킬 수 있다.
이것은 로보트와 곡선를 추적하는데 상당한 효과가 있다.
즉 현재 카메라의 position과 orientation P,O,N,A 를 알고 있으므로 P 와 Base좌표계에서 최후로 곡선을 찾은 점을
Xf,Yf 라 하면 align 방향은

$$a_f = \frac{(P_x - X_f)a_x + (P_y - Y_f)a_y - P_z a_z}{\sqrt{(P_x - X_f)^2 + (P_y - Y_f)^2 + P_z^2}}$$

이 되며 위치를 맞추기 위해서 카메라좌표계의 x 축 중심에서의 회전각을 Ox, y 축의 중심에서의 회전각을 Oy 라 하면

$$\gamma_{yz} = \frac{o \cdot a_f}{|a_f||o|} \qquad \gamma_{zz} = \frac{a \cdot a_f}{|a||a_f|}$$

$$O_x = \tan^{-1}\left(\frac{-\gamma_{yz}}{\gamma_{zz}}\right) \qquad \theta_f = \sin^{-1}\left(\frac{n \cdot a_f}{|n||a_f|}\right)$$

이 되며 이를 이용하여 Robot hand와 독립적으로 카메라를 움직일수 있다.

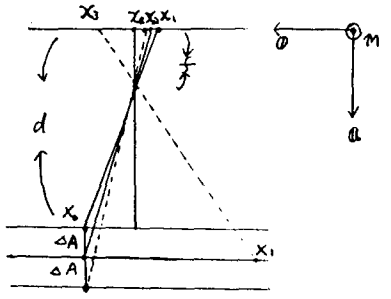


그림 7. 카메라 Celibration

V). Camera Calibration

카메라좌표계에서 Base좌표계로 변환할때 실제적인 실거리 비율 및 촛점거리 f 를 구하는 그림 7). 에서 Nx, Ny 를 구할 수 있는데 방정식 4개에 미지수 3개인 비선형 방정식이므로 일반적으로 computer를 사용한 iteration 방법으로 해를 구하나, 본 논문에서는 4개 방정식으로 close form의 해를 구하여 moniter 의 Ny/Nx 가 0.85 라는 것을 이용하여 최적해를 구하였다.

$$x1 \cdot Nx = -\frac{X_0 \cdot f}{(d - f)} \qquad x2 \cdot Nx = -\frac{X_0 \cdot f}{(d + \Delta A - f)}$$

$$x3 \cdot Nx = -\frac{X_1 \cdot f}{(d + \Delta A - f)} \qquad x4 \cdot Nx = -\frac{X_0 \cdot f}{(d + 2\Delta A - f)}$$

위 방정식에서 Nx, d , f 를 구하였으며 또한 y 값에 대해서도 Nx, d,f 를 구하여 제한조건 f>0 을 이용했다. 또 정확한 값을 위해 d 와 A 를 달리하여 구하였다.

$$d = \frac{(Z_2 Z_4 Z_3 X_0 + Z_1 Z_2 (Z_4 X_1 - Z_3 X_0))\Delta A}{X_0 Z_4 Z_3 (Z_1 - Z_2) - Z_1 Z_2 (Z_4 X_1 - Z_3 X_0)}$$

$$Nx = \frac{d \cdot X_0 (Z_1 - Z_2)}{Z_1 Z_2 \Delta A} - \frac{X_0}{Z_1}$$

$$f = \frac{Z_1 \cdot N_x \cdot d}{Z_1 N_x + X_0}$$

이와 같은 방법을 y 에 대해서도 구하였다. 그 결과 f 와 d 는 일치하고 Ny/Nx=0.85 이었다.

VI). 로보트 Kinematics and Invese Kinematics

본 논문에서는 Rhino-xr2 의 Kinematics and Inverse kine-matics를 풀겠다.

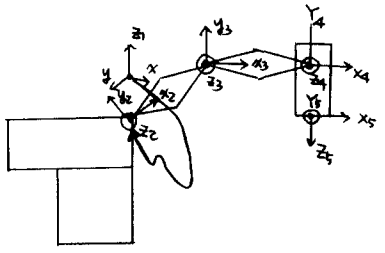


그림 8). Rhino-xr2 의 joint coordinate

Rhino-Xr2 의 Link parameter

Linki	α_{i-1}	d_i	a_{i-1}	θ_i
1	0	d_1	0	θ_1
2	90°	0	0	θ_2
3	0	0	a_2	θ_3
4	0	0	a_3	θ_4
5	90°	d_5	0	θ_5

인접 Link간의 좌표변환 ($i-1$ ~ i)

$$ {}^{i-1}_{i}T = \begin{pmatrix} C\theta_i & -S\theta_i & 0 & a_{i-1} \\ S\theta_i C\alpha_{i-1} & C\theta_i C\alpha_{i-1} & -S\alpha_{i-1} & -S\alpha_{i-1} d_i \\ S\theta_i S\alpha_{i-1} & C\theta_i S\alpha_{i-1} & C\alpha_{i-1} & C\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

전체 변환 (${}^{0}_{5}T$) ∴ Rhino 는 5축임

$$ {}^{0}_{5}T = \begin{pmatrix} C_1C_{234}C_5 + S_1S_5 & -C_1C_{234}S_5 + S_1C_5 & C_1S_{234} & C_1(d_5S_{234} + a_3C_{23} + a_2C_2) \\ S_1C_{234}C_5 - C_1S_5 & -S_1C_{234}S_5 - C_1C_5 & S_1S_{234} & S_1(d_5S_{234} + a_3C_{23} + a_2C_2) \\ C_5S_{234} & -S_5S_{234} & -C_{234} & -d_5C_{234} + a_2S_{23} + a_3S_2 + d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}. $$

Inverse Kinematics는 대수적 Inverse Kinematics와 기하학적으로 풀수있는데 본 논문에서는 대수적 Inverse Kinematics로 푼다.

이에 대한 Inverse Kinematics는

$$ {}^{0}_{5}T = \begin{pmatrix} r11 & r12 & r13 & r14 \\ r21 & r22 & r23 & r24 \\ r31 & r32 & r33 & r34 \\ r41 & r42 & r43 & r44 \end{pmatrix} $$

에 대하여

$$ \theta_1 = \tan^{-1}\left(\frac{r_{24}}{r_{14}}\right) \qquad \theta_5 = \tan^{-1}\left(\frac{f_{21}}{-f_{22}}\right) $$

θ_2 이하 θ_3, θ_4는 2개의 해가 존재한다.

$$ {}^{4}_{1}T\,{}^{0}_{5}T = \begin{pmatrix} f11 & f12 & f13 & f14 \\ f21 & f22 & f23 & f24 \\ f31 & f32 & f33 & f34 \\ f41 & f42 & f43 & f44 \end{pmatrix} $$

$$ \theta_2 = \tan^{-1}\left(\frac{t_1}{t_2}\right) $$

$$ t1 = \frac{2\alpha\gamma \pm \sqrt{4\alpha^2\gamma^2 - 4(\gamma^2+\alpha^2)(\gamma^2-\beta^2)}}{2\cdot(\beta^2+\alpha^2)} $$

$$ t2 = \frac{2\beta\gamma \pm \sqrt{4\beta^2\gamma^2 - 4(\alpha^2+\beta^2)(\gamma^2+(-a^2))}}{2\cdot(\beta^2+\gamma^2)} $$

$$ \alpha = -2a_2(-d_5f_{23} + f_{11}) $$

$$ \beta = -2a_2(-d_5V_{33} + V_{34} - d_1) $$

$$ \gamma = a_3^2 - a_2^2 - (V_{34} - d_5f_{13} + f_{11})^2 - (V_{34} - d_5V_{33} - d_1)^2 $$

에서 4개 해가 나온다. 각각에 대하여

$$ \theta_3 = \tan^{-1}\left(\frac{f_{14} - d_5f_{13} - a_2\cos\theta_2}{f_{34} - d_5f_{33} - a_2\sin\theta_2}\right) $$

$$ \theta_4 = \tan^{-1}\left(\frac{f_{13}}{-r_{33}}\right) - \theta_3 - \theta_2 $$

의 4개의 해 중에

$$ V_{34} = -d_5C_{234} + a_3S_{23} + a_2S_2 + d_1 $$

을 만족하는 해는 2개가 있다.

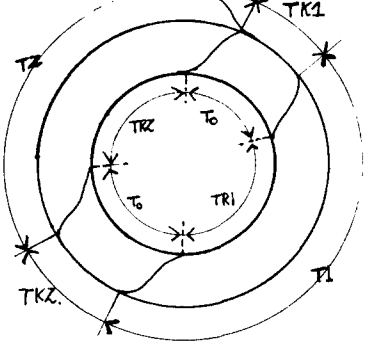


그림 9). 로보트 Running time과 Computer processing time

VII). IBM/PC의 계산시간과 로보트의 Running 시간

IBM/PC에서 화상처리및 Inverse KInematics등 계산시간과 로보트의 Running 시간은 효율문제를 결정한다.

이를 도표로 나타내면

T1 = first 화상 처리시간 TR1 = 로보트 진행시간

T2 = sencond 의 화상 처리시간 TR2 = 2 번째 로보트 진행시간

TK1 = first kinematics and inverse kinematics 계산시간
TK2 = second kinematics and inverse kinematics 계산시간
TO = 정지 시간
T1 + T2 + TK1 + TK2 = TR1 + TR2 + 2TO
이 된다.
IBM/PC의 계산과 로보트의 구간 진행시간이 서로 중첩되어 로보트의 진행 시간이 쉬지 않고 계속 갈 수가 있으나 촬영 시간및 hole check 등 문제로 정지 시간 To 가 있게 된다.

VIII). 실험결과 및 시뮬레이션

본 논문에서 로보트의 곡선추적에 필요한 수학적 정의 및 실시간 곡선추적 알고리즘을 제안하였으며 실제 곡선추출 알고리즘의 계산 시간은 3.5초 정도가 되었고 Inverse Kinematics 계산 시간이 1초 소요되었으며 카메라 Calibration 에서 f 와 Nx, Ny는 f=5.32mm, Nx=0.004911, Ny=0.0041으로 실제 변환 Error 는 4% 가 되었고 면적 변환 Error 는 16% 가 되었다.
시뮬레이션은 Rhion-xr2 로 하였으며 카메라를 수직으로 하여 시행하였다. 화상처리시간과 로보트의 구동시간은 각각 약 4 초와 1.5 초 걸렸다. 로보트의 한 이동구간은 1.5 Cmm 로 하였다. 결과는 만족스럽지만 아직도 화상처리시간이 길어 로보트가 정지하는 시간이 2.5 초 정도가 되어 부드럽지 못하였다.

IX. 결 론.

본 논문에서 제안한 알고리즘은 이동로보트의 궤적 안내 및 공장자동화에 유용하게 사용될수있으며 로보트에게 상당한 독립성을 줄수있다. 특히 로보트와 사각센서간의처리시간을 좀더 효율적으로, 예를들면 두 시스템을 하나의 controller가 담당하게 하면 로보트가 정지되는 과정없이 연속적으로 추적하는 사람의 눈과 같은 feedback system으로 만들수가 있다.

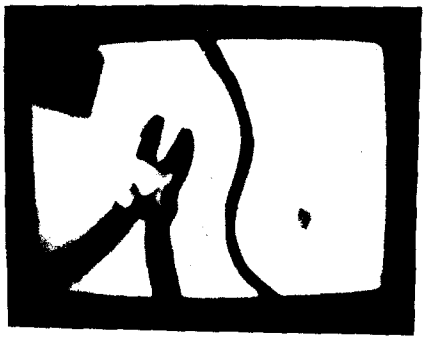


그림10).단일곡선 화상.

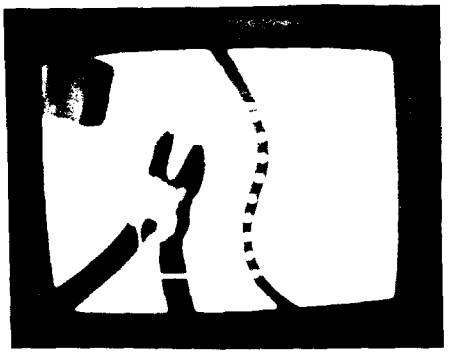


그림11).그림10) 의 처리된 화상.

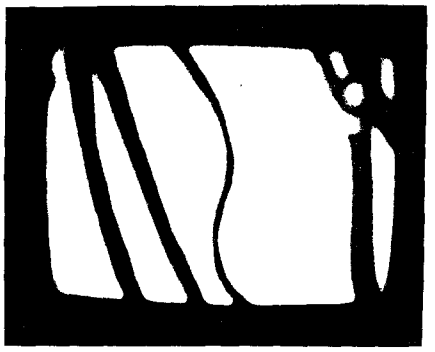


그림12).다중곡선 화상

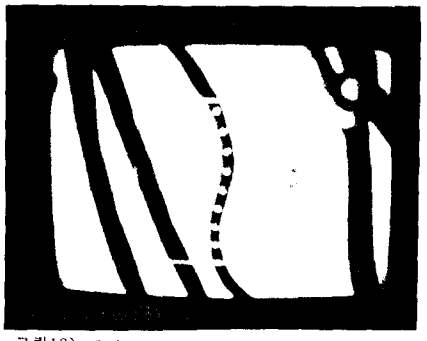


그림13).그림12).의 처리된 화상.

Reference

(1) R.Paul,"Robot Manipulators: Mathematics, Programing, and Control" The MIT Press, 1982.

(2) T. Pavlidis, "Algorithm for Graphics and Image Processing", Computer Science Press,1982.

(3) C.S.G. Lee,R.C. Gonzalez and K.S. Fu (eds.), Tutorial on Robotics, (IEEE Computer Society Press:Silver Spring,1983), PP.5-25.

(4) H. Niemann, Pattern Analysis (Springer- Verlag:Berlin, pp.27-28.