Journal of the Korean Institute
of Industrial Engineers
Vol. 18, No.2, Dec., 1992

시각센서를 이용한 고기능 산업용 로보트의 제어 시스템 연구+

박 혜숙*, 김 남정**, 장 동식*, 박 귀태**

Development of control system for High performance Industrial Robots using Visual Sensor+

Hea sook Park,* Nam jeong Kim,** Dong sik Jang,* Gwi tae Park**

Abstract

The purpose of this study is to improve the performance of robot by providing visual function with robot and developing high-functioned controller and control program. The developed high-functioned controller and software have the better accuracy and flexibility in the movement of robot by complementing the existing robot controller and software problems.

To provide visual function to robot, camera calibration, thresholding and contouring tools are also developed and applied in this study.

These tools help robot recognize the central points and orientations of objects on work-bench.

1. 서 론

산업용 로보트는 1960년대에 처음으로 산업계에 등장하여 제품생산에 큰 몫을 담당하고 있다. 더우기 생산현장에서의 자동화 시스템이 복잡해지고, 그에 따른 인간의 기술수준이 향상됨에 따라, 산업현장에서 흔히 존재하는 단순 반복적인 작업들을 로보트에게 대행시킴으로써 작업의 효율을 증대시키고자 하는 노력이 점점 증가하고 있다. 그러나 대부분의 산업용 로보트는 극히 제한된 범위내에서의 지적기능만을 가지고 있기 때문에 작업의 구조가 잘 설계되어 있지 않으면 아주 단순한 작업도 수행하기가 힘들고 작업 내용이나 작업환경이 변경된 경우 로보트는 필요한 작업을 제대로 수행하지 못한다. 따라서 시각, 청각 등 정보처리기능이나 현장에 대한 계획 및 학습기능 등 인간와 우수한 동작제어능력의 일부를 로보트에게 부여함으로써 고기능화된 유연성있는 로보트를 만들려는 시도가 행해지고 있다[6][7]. 실제로 산업현장에서 로보트의 작업은 미리 교시된 고정위치에서의 pick & place 작업이 대부분이다. 작업내용이 복잡해짐에 따라 로보트 제어시간이 많이 필요하고 작업

+이 논문은 1990년도 문교부지원 한국학술진흥재단의 대학부설연구소지원 학술연구 조성비에 의하여 연구되었음.

* 고려대 산업공학과

** 고려대 전기공학과

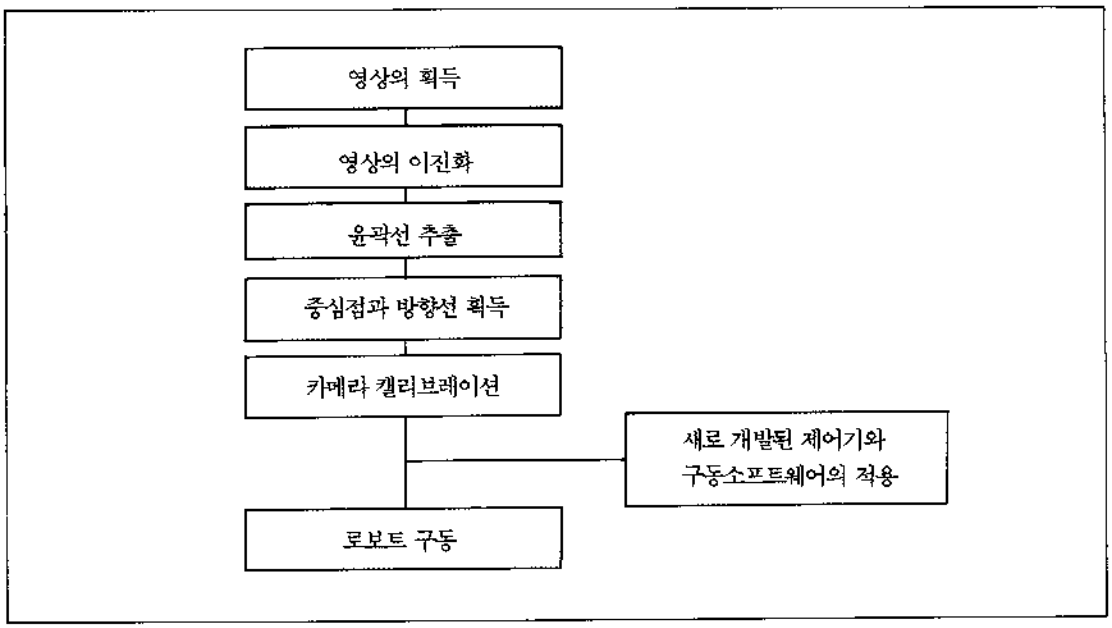
교시 시간도 급증하게 되어 조작 에러의 발생 증가는 물론 생산성측면에서도 기대에 못미치고 있다.

본 연구에서는 로보트에 시각기능을 부여하여 작업자가 미리 교시하지 않아도 로보트가 대상물을 인지하고 대상물에 대한 정보를 획득하여 필요한 작업을 수행할 수 있는 소프트웨어와 또한 기존 제어기의 문제점을 보완한 제어기를 개발하여 고기능화된 로보트 시스템을 구축하였다. 연구의 전체적인 흐름도는 [그림 1-1]과 같다.

2. 시각 센서의 응용

시각센서의 응용부분에서는 실제의 작업 현장에서 teach pendant나 컴퓨터 키보드에 의해 로보트의 작업을 교시하지 않고 카메라 등을 이용하여 대상물을 인지하고 그것에 대한 정보를 획득하여 작업자의 개입없이 로보트가 작업을 할 수 있도록 로보트에 시각 기능을 부여하였다. 대상물에 대한 인지 기능이 없는 로보트는 작업에 내용에 대한 유연성 및 작업효율이 낮다. 그러므로 시각센서의 응용은 로보트의 고기능화의 핵심적인 역할을 한다. 또한 이러한 시각센서의 응용에 있어서 카메라 캘리브레이션(camera calibration)은 필수적이다.

카메라와 로보트가 분리된 작업환경에서 카메라와 로보트가 서로 다른 좌표계(coordinates)를 갖게 되었을 때, 카메라좌표계(camera coordinates)상에서 획득한 대상물의 위치정보를 로보트를 제어하는 데 그대로 이용할 수가 없다. 따라서 카메라좌표계에서 획득한 위치정보를 로보트좌표계의 위치정보로 변환시켜 주어야 할 필요가 있다. 본 연구에서는 카메라좌표계상의 위치정보로 변환시키는 캘리브레이션을 하여, 그 정보를 로보트 제어기로 입력시킴으로써 로보트가 작업자의 개입없이 필요한 작업을 수행할 수 있도록 하였다.



[그림 1-1] 로보트 시스템

2.1 카메라 영상으로부터의 정보획득과정

카메라를 통해 받아들인 대상물의 영상은 계수화(digitization)를 거쳐 컴퓨터의 메모리에 저장된다. 저장된 영상으로부터 대상물에 대한 정보를 추출하기 위하여 최적의 이미지를 만들고 이로부터 필요한 정보를

획득하게 된다. 연구에 사용된 영상은 512×512의 해상도와 256그레이 레벨을 갖는다.

이 과정은 크게 3가지 단계로 나누어 지는데 첫째, 영상의 이치와 둘째, 외곽선 추출(contour extraction) 세째, 대상물의 중심점(central point)과 방향(orientation)을 구하는 것이다.

2.1.1 이치화(thresholding)

각 픽셀의 그레이 레벨을 읽어들여 임계치보다 큰 그레이 레벨을 갖는 픽셀의 그레이 레벨을 1(흰색)값으로 대체시키고 임계치보다 작은 그레이 레벨을 갖는 픽셀의 그레이 레벨을 0(검은색)으로 대체시킨다. 이 단계를 식으로 나타내면 다음과 같다[3].

$$\text{IF } f(x, y) >= T \text{ then } f'(x, y) = 1 \quad (1.1)$$

$$\text{IF } f(x, y) < T \text{ then } f'(x, y) = 0 \quad (1.2)$$

여기서 f(x, y)는 영상의 (x, y) 좌표에서의 그레이 레벨을 나타내고 T는 영상을 이치화 시키는 임계치(threshole value)이다. f'(x, y)는 이치화가 된 후의 (x, y)좌표의 그레이 레벨이다. 이 과정이 끝나면 영상의 왜곡과 잡음은 제거되고 대상물은 배경과 구별되어 강조되게 된다.

2.1.2 외곽산 추출(contour extraction)

이 과정은 이치화 과정이 끝난 후, 대상물에 관한 구체적 정보를 구하는 첫단계로 대상물의 외곽선을 추출하여 대상물을 배경으로부터 분리시키고 그것의 크기나 모양 등에 관한 정보를 얻는 것이다. 여기서 외곽선이란 대상물과 배경(background)의 불연속성이 존재하는 경계면에서의 픽셀들의 집합을 뜻한다[1][10]. 본 논문에서 사용한 외곽선 추출방법은 contour following 알고리즘이다. 이것은 대상물 둘레에 있는 임의의 한점을 출발점으로 하여 본래 그레이 레벨 영상에 단계적으로 대상물의 외곽선을 찾아가는 방법이다. 일단 대상물 주위의 임의의 한점을 시작점으로 외곽선 경로를 따라 해당 픽셀의 뒤에 오는 픽셀을 평가하여 시계 방향으로 대상물을 추적하게 된다. 이 알고리즘의 내용은 다음과 같다.

1. 물체위의 임의의 한 점을 찾고 왼쪽으로 돌고나서 새로운 추적 방향을 탐색하고, 외곡선 추적자가 배경 픽셀에 있으면 오른쪽으로 돌고나서 새로운 추적방향을 탐색한다.

2. 외곽선 추적자가 서치과정에서 찾은 대상물 픽셀에 도달하게 되면 멈춘다.

이 알고리즘에 의해 추출된 외곽선은 해당 픽셀의 좌표값으로서 기억되어 숫자 영상은 매우 작은 양의 데이타로서표현되어지고 대개 단일경로(single path)를 형성하며 대부분의 경우에 폐곡선을 형성한다.

이 알고리즘은 첫째, 8-connected 대상물의 일부가 손상될 수 있고 둘째, 조명조건에 매우 민감하며 세째, 외곽선 점의 중복이 발생할 수도 있고, 마지막으로 대상물내에 위치하는 hole 등의 부분 영역을 찾지 못한다는 단점을 가지고 있으나 알고리즘이 간단하고 빠르며 노이즈에 영향을 덜 받는다는 장점때문에 이용하였다.

2.1.3 대상물의 중심점(central point)과 방향(oprientation)구하기

대상물의 위치는 중심점과 방향을 구함으로써 알 수 있다. 카메라 영상으로부터 획득한 중심점과 방향은 카메라 캘리브레이션을 거쳐 로보트 제어에 필요한 정보로 변환되고 로보트제어기로 전해지게 된다. 로보트는 이 정보를 이용하여 대상물의 중심 위치로 이동, 정확하게 대상물을 잡을 수 있게 된다. 본 연구에서 사용한 방법은 물체의 중심점을 지나는 직선 중에서 원점부터 중심까지의 거리를 일정하게 유지하고, 이 직선과 물체 내의 모든 점들사이의 거리의 합이 최소가 되는 직선을 구하는 것이다. 이 직선은 X축으로부터 θ값이 물체의 방향이 된다[2].

이 과정은 크게 두 과정으로 나누어 진다. 첫째는 대상물의 중심점을 구하는 과정으로 중심점의 좌표(x*, y*)는 식(1.3)에 의해 구해진다. [그림 2-1]는 9개의 픽셀로 이루어진 물체의 좌표를 나타내고 있다.

$$x^* = \frac{\sum_x \sum_y x f(x, y)}{A}, \quad y^* = \frac{\sum_x \sum_y y f(x, y)}{A} \quad (1.3)$$

f(x, y) : 이치화된 그레이 레벨값.
(x, y) : 픽셀의 좌표값.
(x*, y*) : 대상물의 중심점의 좌표값.
A : 대상물의 전체 면적.

(x−1, y+1)	(x, y+1)	(x+1, y+1)
(x−1, y)	(x, y)	(x+1, y)
(x−1, y−1)	(x, y−1)	(x+1, y−1)

[그림 2-1] 픽셀단위로 나타낸 물체의 그림

둘째는 대상물의 방향을 구하는 과정으로 수행절차와 사용된 주변변수는 다음과 같다.

〈주요 변수〉

f(x, y), (x, y), (x*, y*) : 중심점 구하는 과정에서와 동일
θ : x축과 직선이 이루는 각
δ : 원점과 직선 사이의 거리
r : 대상물 영역 내의 임의의 한점과 직선 사이의 거리
E : 영역 내의 모든 점들에 대한 r^2의 합

〈절 차〉

1. 중심점을 지나면서 x축과 각 θ를 이루는 직선의 방정식을 구한다.

$$(x-x^*)\sin\theta-(y-y^*)\cos\theta=0 \quad (1.4)$$

2. 대상물 영역 내의 임의의 한점과 이 직선 거리의 제곱을 구한다.

$$R^2=((x-x^*)\sin\theta-(y-y^*)\cos\theta)^2 f(x, y) \quad (1.5)$$

3. 영역 내의 모든 점들에 대한 R^2의 합 E를 구한다.

$$E=\sum x\sum y((x-x^*)\sin\theta-(y-y^*)\cos\theta)^2 f(x, y) \quad (1.6)$$

4. 식(1.6)에서 $x'=x-x^*$, $y'-y^*$라 하고 식(1.7)에서 식(1.9)으로부터 식(1.6)은 식(1.10)으로 표현된다.

$$a=\sum x\sum y(x')^2 f(x, y) \quad (1.7)$$

$$b=\sum x\sum y(y')^2 f(x, y) \quad (1.8)$$

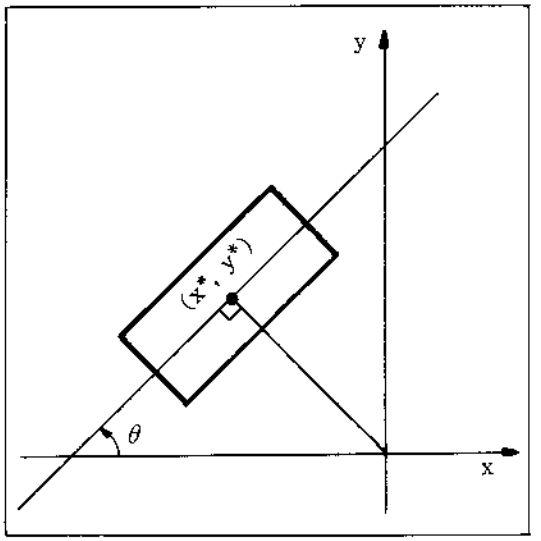
$$c=\sum x\sum y(y'x')^2 f(x, y) \quad (1.8)$$

$$E=a\sin^2\theta+b\cos^2\theta-2b\sin\theta\cos\theta \quad (1.10)$$

5. 식(1.10)의 E 값을 최소화하는 θ^* 값을 구한다.

$$\theta^*=\tan^{-1}(b/a-c)/2 \quad (1.11)$$

수행 결과는 식(1.11)에서 구한 θ^* 값으로 대상물의 방향이 된다. 이때 기울기 θ^*의 직선을 물체의 중심을 지나면서 물체 내의 모든 점에서 이 직선사이의 거리를 최소화한다. [그림 2-2]는 평면상에서 원점으로부터 중심까지의 거리가 δ이고, X축으로부터 각 θ를 이루는 직선을 나타내고 있다.

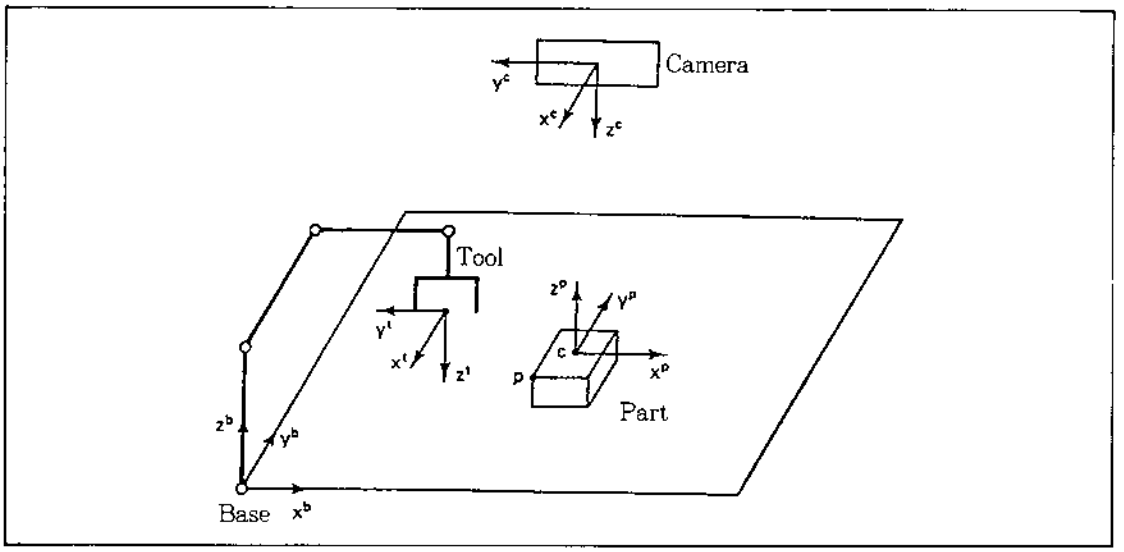


[그림 2-2] 평면상에서의 기울기 θ인 직선

2.2 카메라 캘리브레이션

일반적으로 카메라와 로보트는 분리된 작업 환경에 있다. 그러므로 두 장치는 서로 다른 좌표계(coordinate system)를 가지게 된다. 이에 대한 그림은 [그림 2-3]과 같다. 로보트와 카메라가 분리되어 있으므로 카메라 좌표계(camera coordinate system : CCS)상에서 획득한 대상물에 관한 위치정보(x, y, z, pitch, roll)들은 로보트좌표계(robot coordinate system :

RCS)에서의 위치 정보와 일치하지 않는다. 따라서 CCS와 RCS사이의 대응관계를 구하는 것을 카메라 캘리브레이션이라 한다[9][13]. 본 연구에서는 CCS와 RCS 사이의 관계를 4×4변환행렬로 나타내고 이 행렬을 구하여 CCS상에서 얻은 물체의 위치정보로부터 RCS상의 위치정보를 구하였다.



[그림 2-3] 서로 다른 좌표계상에 있는 카메라와 로보트

이 알고리즘에서 두 좌표계 사이의 관계는 다음 식으로 표현된다.

$C = AR$

$$\begin{bmatrix} C_x \\ C_y \\ C_z \\ 1 \end{bmatrix} \quad \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \quad \begin{bmatrix} R_x \\ R_y \\ R_z \\ 1 \end{bmatrix}$$

$R^t=(R_x, R_y, R_z, 1)$, $C^t=(C_x, C_y, C_z, 1)$

여기서 R은 RCS상의 임의의 점을 나타내는 벡타이고, C는 CCS에서의 R에 대응하는 벡타이다. 그리고 A행렬은 4×4 변환행렬(transformation matrix)이다.

이 알고리즘을 이용하여 카메라 캘리브레이션을 할 때 카메라 한 대를 사용하였기 때문에 3차원 공간에서의 임의의 한점은 카메라의 2차원 평면상의 점으로 대응되므로 물체의 높이에 대한 정보는 무시된다. 이를 고려할 때, 식(2.1)은 식(2.2)로 축소된다. 여기서 축소된 A (3×3) 행렬을 A'로 표시한다.

$$\begin{bmatrix} C_x \\ C_y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{14} \\ a_{21} & a_{22} & a_{24} \\ a_{41} & a_{42} & a_{44} \end{bmatrix} \begin{bmatrix} R_x \\ R_y \\ 1 \end{bmatrix} \tag{2.2}$$

카메라 캘리브레이션 절차는 다음과 같다.

1. A'행렬의 원소가 9개이므로 이 해를 얻기 위해서는 RCS에서의 기지(known)의 3점과, 이에 대응하는 CCS의 3점을 알아야 한다. 이들의 관계는 식(2.3)과 같다.

$$\begin{bmatrix} C_{x1} & C_{x2} & C_{x3} \\ C_{x1} & C_{x2} & C_{x3} \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{14} \\ a_{21} & a_{22} & a_{34} \\ a_{31} & a_{32} & a_{34} \end{bmatrix} \times \begin{bmatrix} R_{x1} & R_{x2} & R_{x3} \\ R_{y1} & R_{y2} & R_{y3} \\ 1 & 1 & 1 \end{bmatrix} \tag{2.3}$$

2. 식(2.3)로부터 A'행렬을 구한다.

$$\begin{bmatrix} a_{11} & a_{12} & a_{14} \\ a_{21} & a_{22} & a_{24} \\ a_{41} & a_{42} & a_{44} \end{bmatrix} = \begin{bmatrix} C_{x1} & C_{x2} & C_{x3} \\ C_{y1} & C_{y2} & C_{y3} \\ 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} R_{x1} & R_{x2} & R_{x3} \\ R_{y1} & R_{y2} & R_{y3} \\ 1 & 1 & 1 \end{bmatrix}^{-1} \quad (2.4)$$

3. 과정 2에서 구한 A'행렬의 역행렬과 CCS에서의 대상물의 위치정보로부터 이에 대응하는 RCS의 위치정보를 식(2.5)에 의해서 구한다.

$$\begin{bmatrix} R_X \\ R_Y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{14} \\ a_{21} & a_{22} & a_{24} \\ a_{41} & a_{42} & a_{44} \end{bmatrix}^{-1} \begin{bmatrix} C_x \\ C_y \\ 1 \end{bmatrix} \quad (2.5)$$

4. 과정 3에서 구한 RCS의 위치정보는 로보트 제어기로 전달된다.

3. 로보트 제어기와 구동 소프트웨어의 개발

본 연구에서 사용된 로보트는 5축 로보트인 SCORBOT ER Ⅲ이며 이 로보트는 로보트 본체와 이를 제어하는 로보트제어기(controller)로 나누어져 있다. 로보트는 컴퓨트로 제어기를 통하여 제어를 할 수 있다. 컴퓨터로 제어하기 위해서는 제작사에서 제공하는 램상주 구동소프트웨어를 호출하여 사용한다. 컴퓨터와 제어기 사이의 통신방식은 RS-232C를 사용하고 있다[11].

3.1 SCORBOT ER-Ⅲ 제어의 특징

3.1.1 모터 구동 시스템(drive system)

로보트의 모든 조인트는 DC 서버모터에 의해 구동된다. 각 모터는 cloosed-loop로 제어되므로 움직인 각도와 방향이 feedback된다. 이를 위하여 각 모터마다 rotary optical encoder가 장착되어 있다. 그리고 각 모터들은 펄스단위로 제어된다. 모터들의 transmission ratio에 의해 한 펄스당 각 축의 회전율을 구할 수 있다. 로보트 각 축의 회전율은 〈표 3-1〉과 같다.

〈표 3-1〉 로보트 각 축의 회전율

축이름	base	shoulder	elbow	pitch	roll
회전율	0.094	0.1175	0.1175	0.458	0.458

또한 호스트 컴퓨터에서 보내는 모든 명령은 아스키코드로 전환되어 로보트 제어기로 순차적으로 전달된다. 이 명령들은 제어기로 보낸 명령어에 대해 리턴 값들의 유무에 따라, 그리고 어떤 값이 리턴되는 지에 따라 세가지 타입으로 나누어진다[4].

1. Uni-directional commands : 구동명령
2. Regular bi-directional commands : 각 모터의 상태 체크명령
3. Special bi-directional commands : 모터의 회전각 측정명령

3.1.2 스코보트 구동소프트웨어

스코보트 구동소프트웨어로 Utilities가 있다. 이 소프트웨어는 램상주 프로그램으로서 assembley언어, basic, c언어와 인터페이스가 가능하다. 따라서 사용자는 이 소프트웨어에서 제공되는 함수(function)를 이용하여 프로그래밍할 수 있다. 그런데 이 소프트웨어를 이용하여 로보트 제어를 하고자할 때는 제어하고자 하는 축의 관절값으로 펄스값을 입력하여야 한다[12]. 그런데 본 소프트웨어로는 작업물의 위치정보(x. y. z. pitch roll)로부터 각 축의 관절값을 구할 수 없고 사용자가 임의로 값을 입력해 주어야 한다. 따라서 작업물 위치로 정확하게 이동하지 못하고 위와 같은 작업을 반복적으로 수행해야하는 비효율성이 문제로 제기된다. 제어기의 순차적인 명령수행 방법과 소프트웨어상의 문제점 때문에 이 로보트는 저수준의 작업 능력을 갖게 되고 작업내용에 대한 유연성도 떨어져 다양한 형태의 작업을 수행하지 못하는 경우가 많다. 본 로보트의 제어기와 소프트웨어의 문제점은 다음과 같다.

1. 각 축의 독립적인 제어는 가능하지만 동시제어가

불가능하다.

2. 속도가 느리다.

3. 축 제어시 미리 입력값을 알고 있어야 한다.

4. 로보트의 현재위치를 파악할 수 없어서 원하는 위치로 이동했는지, 필요한 양만큼 움직였는지 파악할 수 없다.

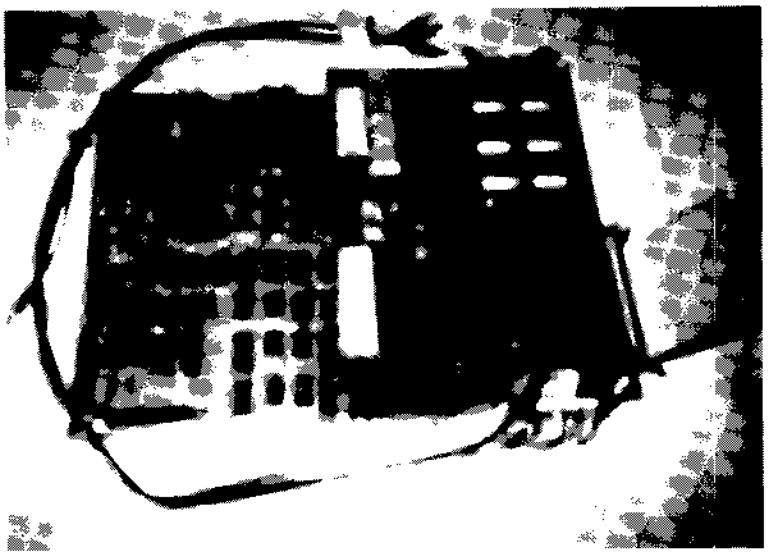
3.2 새로운 제어기와 구동소프트웨어의 개발

위에서 지적한 문제점들을 해결하고 제어기 자체의 성능을 향상시켜 제어속도와 제어의 유연성을 높이고자 새로운 제어기와 구동 소프트웨어를 개발하였다. 개발된 제어기는 기존의 제어기가 +15, -15V의 전원을 필요로 하는데 비해서 모터 구동 방식에 있어서도 전압 기존 방식과는 달리 PWM(pulse width modulation) 방식을 채택하여 제어의 유연성을 높이고 있다. 또한 기존의 제어기는 위치제어시 counter를 사용하지 않고 encoder의 상태를 직접 읽음으로써 제어한 반면, 개선된 제어기는 모터에서 나오는 encoder pulse의 수를 4체배하고 이를 카운팅하여 위치에러를 줄이고 있다. 또한 완전 디지틀화된 방식을 채용하여 사용자의 목적에 따라 소프트웨어의 개발이 가능하므로 이 부분이 기존의 제어기에 비해서 유연성면에서 뛰어나다고 할 수 있다. 그리고 고속의 CUP(SDP)를 채용하여 샘플링 타임을 상당히 줄임으로써 각 축이 CUP를 공유할 수 있어 각 축을 원하는 데로 제어할 수 있다. 모터 구동 방식은 DC 전압을 chopping하여 모터 공급용 DC 전압을 만들어서 모터를 구동하고 있다. 따라서 모터의 특성에 따라 광범위하게 속도를 가변시킬 수 있도록 설계되었다.

3.2.1 개발된 제어기의 구성

DC 모터 위치제어를 수행하기 위해 설계 및 제작된 하드웨어는 엔코더와 연결된 4체배 위치검출회로, PWM 파형 발생회로, PWM 구동회로 및 PWM 증폭회로 등으로 구성되어 있다. 이들의 동작과 상호관계를 간단히 설명하면 다음과 같다.

카메라 캘리브레이션 과정에서 로보트 제어기로 입력된 위치정보를 마이크로 프로세서에 의해 받아들인 후 펄스폭을 계산한다. 이 데이터가 PWM 발생회로로 보내져 PWM 파형이 만들어지고 이 파형은 PWM 구동회로를 거쳐 PWM 증폭기에 인가되어 모터를 구동하게 된다. 개발된 제어기의 하드웨어 구성도는 [그림 3-1]과 같다. 제어기의 주요 하드웨어에 대한 설명은 다음과 같다.



[그림 3-1] 개발된 제어기의 하드웨어

3.2.1.1 DSP(digital signal processor)

DSP의 구조는 신호 처리 시스템을 위해 최적화 되어 있다. 주요 구조는 다중 버스, 16비트 구조, 32비트 레이스터 그리고 많은 기능의 하드웨어적 구현 등이다. DSP는 신호 처리에 있어서 계산상의 문제점을 최소화하고, 고도의 세련된 기술을 이용하여 고성능 시스템에서 요구하는 대역폭을 충족시킨다. TMS320의 구조적 특징과 그 장점을 〈표 3-2〉에 요약하였다.

〈표 3-2〉 TMS320의 구조적 특징

특 징	장 점
1사이클 명령어	실시간으로 고급제어 알고리즘을 수행한다.
파이프라인구조	높은 대역폭의 시스템을 제어한다.
하바드 구조	데이터와 명령어를 동시에 가져온다.
하드웨어 곱셈기	계산시간 지연을 최소화한다.
하드웨어 쉬프터	넓은 다이나믹 레인지를 갖는다.
16비트 워드길이	양자화 오차를 최소화한다.
32비트 레지스터	잘림에 의한 오차를 최소화한다.
하드웨어 스택	빠른 인터럽트 처리를 제공한다.
포화모드	오버플로우에 의한 누산기의 부호 바뀜을 막는다.

3.2.1.2 PWM 발생회로

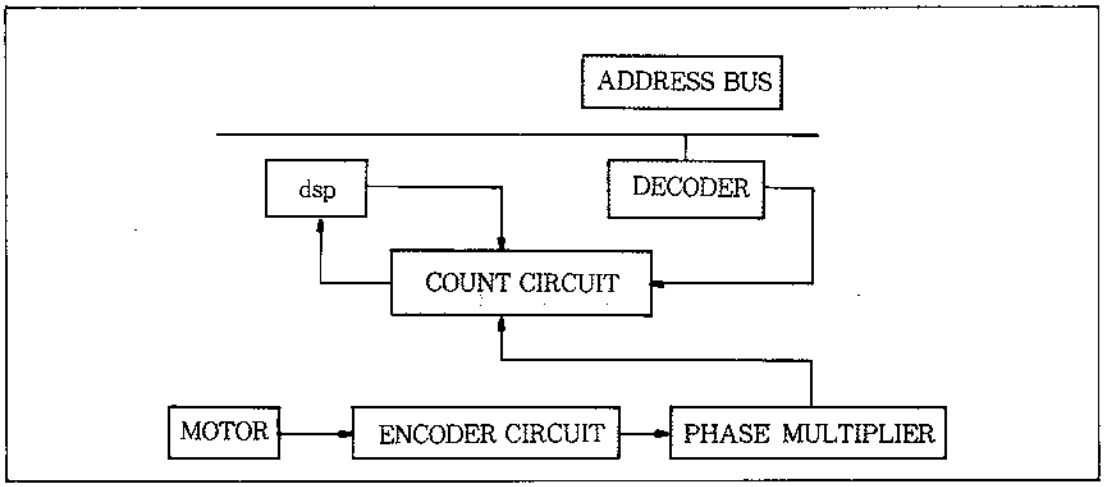
마이크로 프로세서가 입력 정보를 받아들이거나 연산을 하는 동안에도 자동적으로 적당한 PWM 신호를 발생시켜 줄 수 있는 회로가 필요하다. 이와 같은 기능을 하는 것이 PWM 발생회로이다. Analog PWA 파형 발생회로는 복잡한 회로로 구성되어 있어 조정이 어렵고 전기적 잡음에 약한 반면에 digital PWM 파형 발생회로는 간단한 digital회로로 되어있고, switching 주파수를 쉽게 바꿀 수 있다는 장점이 있으나 가격면에서 불리하다.

3.2.1.3 펄스 4체배 회로

기존의 제어기의 펄스 코더는 회전당 6개의 펄스를 회전 방향에 따라 90도의 위상차를 갖는 A상과 B상의 2가지 펄스를 발생시켰다. 그러나 보다 정밀한 정보를 얻고 특히 저속에서의 펄스 손실을 막기위해 4체배 회로를 설계 제작하여 펄스의 수를 4배로 늘려 주었다.

3.2.1.4 위치 검출 회로

펄스 코더는 모터가 회전하는 각도에 비례하여 펄스를 발생한 장치이다. 따라서 이 펄스를 카운트하면 모터가 얼마만큼 회전하였는가를 알 수 있으며 모터의 현재 위치를 알 수 있다. 펄스 코더에서 발생하는 펄스를 카운터하기 위해서는 펄스 카운트 회로가 필요하다. 이 회로는 모터의 회전양을 카운트하여 입력 포트를 거쳐 마이크로프로세서로 읽어들이도록 구성했다. 위치검출 회로의 블록선도는 [그림 3-2]와 같다.

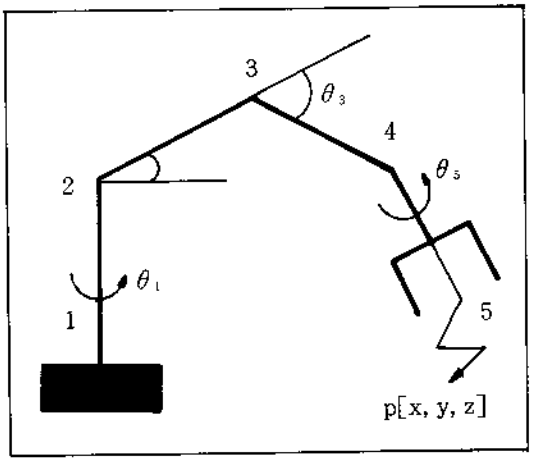


[그림 3-2] 위치검출 회로의 블록선도

3.2.2 로보트 구동소프트웨어의 개발

로보트를 컴퓨터로 제어하기 위해서는 로보트의 상호 인접한 연결축 사이에 운동경로방정식이 필요하게 된다. 즉 목표점이 입력값으로 주어지면 그리퍼로부터 좌표계에 대한 각 조인트 운동방정식의 해 θ의 값을 구하여 로보트 각 축을 제어한다. 이 해를 구하는 방법으로는 대수적인 방법에 의해 해를 구하는 denavit와 hartenberg의 A행렬방법이 있고 기하학적인 해석에 의해 해를 구하는 geometric inverse kinematics가 있다[5]. 본 연구에서는 기하학적인 해석에 의한 방법을 사용하였다.

[그림 3-3]는 로보트 각 축의 관절값을 구하기 위해 사용되는 각 관절의 위치를 나타낸 것이다. 각 관절의 위치를 나타내고 각도를 구하는 절차와 사용된 주요변수는 다음과 같다.



[그림 3-3] 로보트의 각 관절값의 할당

〈주요 변수〉

P(px, py, pz) 공간상에서 로보트의 그리퍼가 놓여야할 목표점의 좌표값

len12 : base의 길이

len23 : shoulder의 길이

len34 : elbow의 길이

len45 : pitch의 길이

$\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$: base축부터 roll 축의 회전각

로보트 각 관절값의 해를 구하는 주요 절차는 다음과 같다.

1. θ_1

   허리관절의 회전각 θ_1을 구한다.

   $\theta_1 = \tan^{-1}(py/pz)$

2. θ_2

   수평면과의 어깨관절(shoulder) 사이에 이루는 각 θ_2를 구한다. 그림에서처럼 θ_2는 θ_{21}과 θ_{22}에 의해 구할 수 있다.

   $\theta_{21} = \cos^{-1}(len23^2 + len24^2 - len34^2/(2\ len23\ len24))$

   $\theta_{22} = \cos^{-1}(len24^2 + len12^2 - len14^2/(2\ len24\ len12))$

   $\theta_2 = \theta_{21} + \theta_{22} - 90$

3. θ_3

   [그림 3-3]에서처럼 어깨관절(shoulder)과 손목관절(elbow)이 이루는 각 θ_3를 구한다.

   $\theta_3 = \cos^{-1}(len24^2 + len23^2 - len34^2/(2\ len23\ len24))$

4. θ_4

   피치(pitch)와 수평면이 이루는 각 θ_4를 구한다.

   $\theta_4 = \cos^{-1}(len45^2 + len56^2 - len46^2/(2\ len45\ len56))$

5. θ_5

   θ_5는 2장 1절에서 구한 대상물의 방향과 직각이 되는 값이다.

새로운 로보트 구동소프트웨어를 개발함으로써 실제 작업 현장에서 작업자가 teach pendant나 컴퓨터 키보드에 의해 로보트의 작업을 지시하지 않아도 된다. 즉, 카메라를 이용하여 작업물의 중심점으로 로보트를 이동, 정확하게 물체를 잡아, 다른 위치로 이동시킬 수 있도록 하였으며, 또한 작업물의 위치가 변경되어도 작업자가 다시 위치정보를 입력시킬 필요가 없게 하였다.

4. 결 론

본 연구에서는 로보트에 시각기능을 부여하여 작업자

가 미리 교시하지 않아도 로보트가 대상물을 스스로 인지하고 대상물에 대한 정보를 획득하여 일련의 작업을 수행할 수 있는 소프트웨어를 개발하였다. 또한 기존 제어기의 제어방식의 문제점을 보완하여 로보트를 효과적으로 제어할 수 있는 제어기를 개발하여 고기능화된 로보트 시스템을 구축하였다.

본 연구에서 개발한 로보트 시스템은 기존의 시스템보다 작업수행의 정확도가 4배 향상되었고, 각 축의 동시제어와 구동속도의 향상으로 작업수행 속도가 크게 빨라졌으며 각 축의 동시제어와 독립제어가 가능하고 속도조절이 가능하도록 개발되었기 때문에 작업내용에 따라 유연하게 대처할 수 있다. 또한 좌표계상에서 작업물의 위치정보(x, y, z, orient)만으로 로보트를 자동으로 작업물의 중심위치까지 구동시켜 작업을 수행할 수가 있었다.

시각센서 응용시 작업물의 정보를 얻는 과정에서 발생한 문제점으로는 기역자 모양이나 디귿자 모양과 같은 특이한 형태의 물체인 경우에는 정확한 방향선을 구할 수가 없다. 이러한 문제점을 해결하기 위해서는 패턴의식(patten recognition)이 가능하도록 하거나, 물체에 대한 다양한 정보의 획득 기법을 개발해야 할 것이다. 또한 현재 개발된 시스템은 카메라 한 대를 이용하였기 때문에 작업물의 높이에 대한 정보를 시각센서의 응용부분에서는 구할 수가 없었고, 높이정보는 작업자가 직접 입력하거나 값을 고정시켜야 하는 문제점이 있었다. 이 부분은 스테레오 비젼(stereo vision) 등을 이용한 3차원 정보를 추출하는 연구를 통해 개선시켜야 할 것이다.

참 고 문 헌

[1] Ballard, D. H. and Brown, C. M., *Computer Vision*, Prentice Hall Inc. New Jersey, 1982.

[2] Berthold Klaus Pour Horn, *ROBOT VISION*, THE MIT PRESS, pp 46-53.

[3] Fairhurst, M. G., *Computer Vision for Robotics System* : An introduction, Prentice Hall, New Jersey, 1988.

[4] HAIM SCHLEIFER B. Sc., Eng *SCORBOT EDUCATIONAL ROBOT TEXTBOOK* 2, 1984.

[5] J. J. Craig, *Introduction to ROBOTICS Mechanics & Control*, ADISSON WESLEY PIBLISHING COMPANY, 1955.

[6] Mansour R. and Waldemar K., "A Reserch Paradigm in Human-Robot Interaction", International Journal of Industrial Ergonomics, Elsevier, pp. 59-71, 1990.

[7] Nof, S. Y., *Robot Ergonomics : optimizing robot work*, Handbook of industrial robot, pp. 549-604, 1985.

[8] *PCVISONplus FRAME GRABBER USER'S MANUAL*, ITI, 1987.

[9] R. C. Gonzalez and P. Wintz, *Digital Image Processing*, 2nd Eds Adisson Wesley, 1987.

[10] Schalkoff, R. J., *Digital Image Processing and Computer Vision*, John Wiley & Sons, 1989.

[11] *SCORBOT ER-Ⅲ USERS' MAMUAL*, ESHED ROBOTEC LIMITED, 1988.

[12] *SCORBOT ER-Ⅲ UTILIT+ MAMUAL*, ESHED ROBOTEC LIMITED, 1988.

[13] ROBERT J. SCHILLING, *FUNDAMENTALS OF ROBOTICS Analysis and Control*, Prentice-Hall International Editions, 1991.