韓國精密工學會 '95年度 秋季學術大會論文集 pp.537~542

PC를 기반으로한 VISION 기능을 갖는 ROBOT 제어기의 개발

서일홍[+], 김재현[++], 정중기[+], 노병옥[+++]

+ : 한양대학교 전자공학과, ++ : 한국전자부품 연구소(KETI),

+++ : 선문대학교 산업공학과

요 약

본 연구에서는 최근 널리 알려져있는 vision function들을 Robot controller에 맞게 수정하고, 또한 각종 windows에 익숙한 사용자들이 보다 쉽게 로보트를 다룰 수 있도록 하기위해, 이러한 기본 기능들을 X-windows 상에서 vision language로 구현하였다. 기존의 기본 motion language, I/O language와 함께 Vision Language를 구현함으로써 Vision based Intelligent Robot system을 구축하였다.

1. 서 론

지난 10여년간 산업용 로보트를 응용한 공장 자동화에 대한 계속되는 노력은 최근에는 생산성 향상을 통한 국제 경쟁력 강화 및 노동자의 처우개선등을 위하여 더욱 자동화에 대한 요구가 증가하고 있다. 공장자동화에서 산업용 로보트의 응용은 단순한 물건을 집어 옮기기 또는 간단한 작업물 분류를 넘어서 연속된 경로를 오차없이 추적하며 작업을 하는 아크 용접 작업이나 고도의 정밀을 요하는 조립 작업에 까지 확대되고 있으며, 따라서 이러한 응용에 vision sensor의 역할이 더욱 중요시 되게 되었다.

그러나 이를 사용하는 방법이 일정한 command shell에서 Interpreter를 사용하는 방법, 혹은 text menu를 바탕으로한 방법들이 주종을 이루고 있다. 또한 실질적인 Vision Sensor의 응용이 Language로 표현되어 있지 못하고 단순한 command 형식만을 지원하는 것이 대개의 경우였다. 따라서, 본 연구에서는 최근 널리 알려져있는 vision function들을 Robot controller에 맞게 수정하고, 또한 각종 windows에 익숙한 사용자들이 보다 쉽게 로보트를 다룰 수 있도록 하기위해, 이러한 기본 기능들을 X windows 상에서 vision language로 구현하였다. 기존의 기본 motion language, I/O language와 함께 Vision Language를 구현함으로써 Vision based Intelligent Robot system을 구축하고자 하였다.

본 연구에서 개발된 Movic (MOtion & VIsion Controller)은 국내의 산업용 Intelligent Robot System의 초석이 되도록 설계되었고, 앞으로 힘 센서, 근접 센서등이 추가되어 다중센서 Interface가 이루어지고 이의 language화가 이루어 진다면 진정한 Intelligent Robot system이 이루어질 수 있을 것이다.

2. Vision Robot Controller System의 구성

2.1 전체 시스템의 구성

본 연구에서는 Reliability와 Performance가 뛰어난 Intel 80486 DX2-66 CPU와 산업용 PC(Industrial PC), Multi-Tasking Real Time OS 인 LynxOS를 Base로 하여 확장성및 유연성이 좋은 시스템이 되도록 구성하였다. 또한, Robot end-effector에 Vision Camera를 설치하여, calibration, 형상 인식, 각종 기하학적인 특징량등의 2-D Vision Operation 를 할 수 있도록 하였다. 이러한 vision 기능들을 Robot 제어언어에 포함시키므로써 Object의 위치 보정량을 detect하여 실질적인 Object 위치로의 이동을 Program할 수 있게 되었다.

전체 시스템의 구조는 로보트 동작 교시와 이에 필요한 MOTIF에 의한 X-window 상에서의 Menu display, Interpreter Type Robot language를 이용한 program 편집 및 실행등을 담당하는 CPU Board와 Vision Data 처리를 위해 Vision Board에 DSP Chip 이 내장하고 있다.

특히, 전체 시스템을 일반 산업용 PC에 맞도록 구성하여 User들로 하여금 보다 친숙하게 작동할 수 있게 구현하였다. 전체 시스템의 구성은 그림 1과 같다.

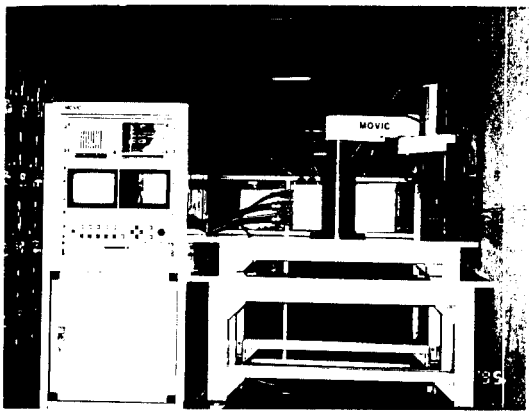

그림 1. MOVIC (MOtion & VIsion Controller)

2.2 제어시스템의 H/W 구성

본 연구에서는 32bit 80486(DX-66)와 ISA BUS방식의 산업용 PC와 Multi-Tasking & Real Time Lynx OS를 기반으로 한 시분할 병렬처리 구조를 갖도록 하여 확장성이 높은 시스템을 구성하였다. 또한 Vision Operation 을 위한 복잡한 계산을 위해 DSP Chip이 내장된 Vision Board를 사용하였다. H/W 구성은 그림 2와 같다.

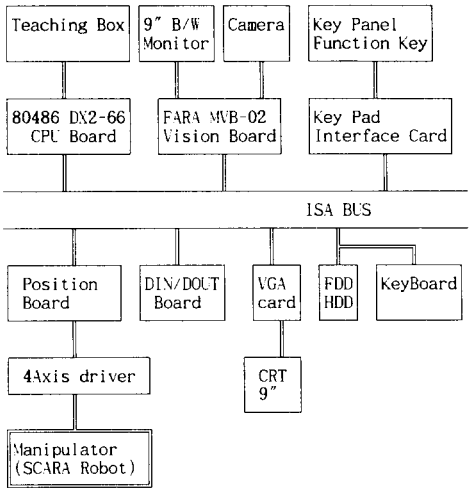

그림 2. H/W Schemetic Diagram

2.3 제어시스템의 S/W 구성

개발된 SoftWare는 Computer가 여러가지 기능을 수행할 수 있도록 표현 방법이 간략하고, 풍부한 연산자를 가지고 있으며, 사용자가 많은 basic언어에 기반을 두고 있다. 입출력 부분 및 Register 조작등에서 사용된 Asembly Language Program을 제외하고는 대부분이 C언어를 사용하여 설계 했기 때문에 쉽게 수정할 수 있으며 될수 있는한 모든 Program을 세분화하여 계층적제어 구조를 이루게 하고 수정및 편집단위를 Program Module별로 할수 있게 하는 새로운 형태의 Manipulator Level Robot 제어 언어를 설계하였다.

이에 대한 시스템의 S/W구조는 그림 3과 같다.

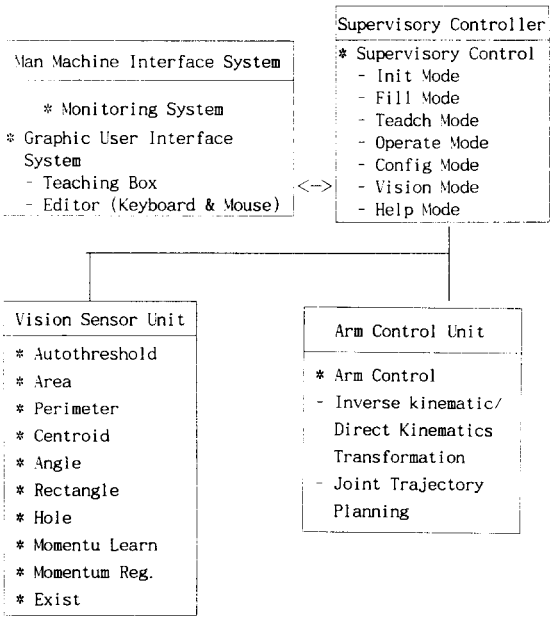

그림 3. S/W Architecture

3. Vision Robot Controller System의 Mode별 기능 및 구성

3.1 Supervisory System

(1) Init Mode

Init Mode는 Robot Controller를 초기화 시켜주는 Mode이다. Servo Power는 Motor Driver가 ON 상태에서 Servo Motor Power를 ON 시켜준다. Home은 Robot Motion의 기준이 되는 원점으로 복귀시킨다.

(2) File Mode

File Mode는 각종 file들을 load, Edit, save하는 등 menu의 전반적인 관리 Mode이다.

- New : edit 화면의 초기화
- Open : job file의 load
- Edit : job program editor
- Save : job file의 저장
- Save as : job file을 다른 이름으로 저장
- Quit : menu program 종료

(3) Teach Mode

Teach Mode에서는 로보트가 움직이기 위해서 필요한 위치 정보를 교시해주는 Mode이다. teach mode는 다음의 4가지 teach 방법을 지원한다.

- Remote Teach : teach box를 이용해서 teach를 할 수 있고 기본적인 프로그램 도 작성할 수 있다. 이때 teach box에서 edit된 프로그램은 RS232C cable을 통해서 PC와 통신을 한다.
- Free Teach : servo power를 끈 상태에서 로보트를 손으로 움직여 원하는 위치에 두고 나서 location data로 저장을 할 수 있다.
- Manual Data Input : 가고자 하는 위치에 대한 data를 직접 입력해서 teach를하는 방법으로 joint angle의 값을 입력해서 add button을 누르면 cartesian 좌표계의 data가 표시된다.
- Teach Box Emulation : teach box에서의 teach mode에 해당하는 mode로서 inching key를 눌러서 로보트를 움직인 뒤 location data로 저장을 할 수 있다.

(4) Operate Mode

Operate Mode는 로보트를 동작시키는 Mode이다.

- Load Teach Data : 실행하기위해 필요한 data file로 저장되어 있는 teach data를 load한다.
- Syntax Check : job file의 Syntax Error를 검사한다.
- Run : 세가지 방식에의해 로보트를 실행시킨다
  - * Step : 한 line씩 실행한다.
  - * Cycle : job의 처음부터 끝까지 한번 실행한다.
  - * Auto : job의 처음부터 끝까지 반복하여 실행한다.

(5) Config Mode

Config Mode는 로보트의 Configuration을 바꾸는 Mode이다. Password를 입력하여야만 access 할 수 있다.

- Setup : Password 입력후 로보트의 각종 Parameter들이 display된다. Mouse로 원하는 부분으로 커서를 옮긴후 수정할 수 있다.
- Change Password : Password를 변경시킨다.

(6) Vision Mode

Vision Mode는 각종 Vision 기능을 실행하기위한 Mode이다.

-Down Load : DSP Chip에 DSP Program을 장착시키는 기능

-Operation : 다음과 같은 10가지의 기능들이 있다.

-Autothresholding : Threshold값을 현재의 화면상태에서 가장 적당한 값으로 설정한다.

-Area : 화면상의 물체의 면적을 구한다.

-Perimeter : 화면상의 물체의 둘레길이를 구한다.

-Centroid : 화면상의 물체의 중심을 구한다.

-Angle : 화면상의 물체 주축의 각도를 구한다.

-Rectangle : 화면상의 물체를 둘러 쌀수 있는 사각형을 구한다.

-Hole : 화면상의 물체의 Hole의 면적과 중심좌표를 구한다.

-Learn : 화면상의 물체를 학습한다.

-Reg : 화면상의 물체를 Moment Learn에서 학습한 값을 가지고 인식 한다.

-Exist : 화면상의 사각형안에 물체가 존재하는지 조사한다.

(7) Help Mode

이 Mode는 munu또는 Controller에 대한 간단한 설명이다.

4. Vision 시스템 구성

본 논문에서 사용된 Robot Vision 시스템은 크게 2가지 응용분야를 목표로하였다. 첫번째 응용분야는 조립작업시의 feeder로 부터 공급되는 부품의 위치오차의 수정과 object의 인식을 통한 object의 분류이다. 이러한 2가지 목표를 이룰 수 있다면 원하는 object의 정확한 위치를 vision operation을 통해 얻고 이를 servo system에게 전달함으로써 보다 효율적인 robot program을 얻을 수 있을 것이다.

4.1 Object Recognition

Object 인식을 위한 Robot Vision System의 구성은 다음 그림 3과 같다.

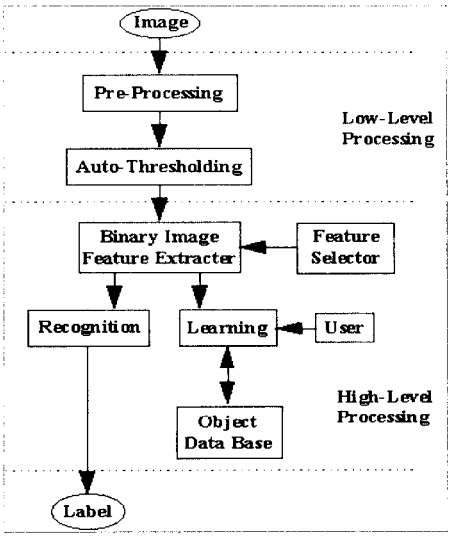

그림 4. Object learning & recognition system

위의 그림에서 보는 바와같이 grey Image가 입력되면 화상의 질을 높이기 위해 pre-processing을 한다. 다음 auto thres-holding을 수행하게 된다. 이러한 과정을 거쳐 binary image를 얻은 후, 이로 부터 아래와 같은 Feature들을 구한다.

Feature 1 : Edge encoding
(using chain coding & check buffer)

Feature 2 : Geometrical values
(area, perimeter, momentum, center_point)

feature 1은 물체를 인식하기 위한 feature 값으로 chain code를[3] 사용함으로써 rotation과 translation에 불변특성을 갖는다. 그러나 Size불변특성은 만족시키지 못하는데 이는 물체를 인식하는 camera의 z-축 position과 인식 대상체의 z-축 위치가 항상 일정하도록 환경을 설정함으로써 극복될 수 있다. 물론 향후에는 거리의 검출를 위해 raser camera를 이용한 active vision를 사용하거나 stereo vision을 이용한 passive vision을 사용하여야 일반적인 알고리즘이 될 수 있을 것이다. feature 2는 위치 error 보정 및 빠른 물체 인식을 위해 사용되었다. 여기서 feature 1을 구하는 방법을 간략하게 소개하면 다음과 같다.

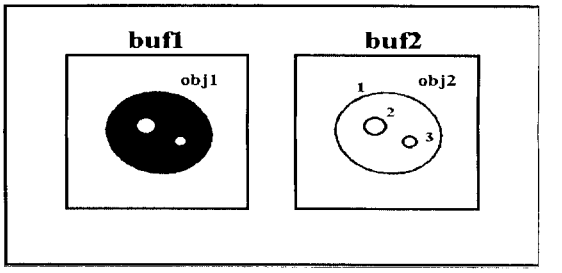

그림 5. chain coding & buffer checking

(1) Edge encoding

그림 1은 2개의 hole이 존재하는 타원 모양의 object를 보여준다. 이러한 object를 encoding하기 위해 먼저 X, Y의 순서로 scanning을 수행한다. 0->1 혹은 1->0으로의 변위가 발생하면 이 지점을 시작점으로하여 8방향 방향벡터를 이용하여 chain code를 만든다. 이와 동시에 buf2에서도 이러한 위치에 1를 기록한다. object내부의 hole를 검사하기 위해 chain coding을 수행한 후 object의 min_X, max_X, min_Y, max_Y를 구한다. 이를 이용하여 다음 scanning의 초기위치를 min_X와 min_Y로 하고, 이 지점 부터 max_X, max_Y까지 scanning을 수행한다. 만일 변위가 발생하면 buf 2의 해당 위치에 1이 기록되어 있는지 확인한 후, 기로되어 있는 경우 이미 encoding한 edge라고 판단하여 skip을 하게된다. 만일 기록되어 있지않다면 encoding을 수행하게 된다. 이와 같은 방법으로 object의 모든 edge를 encoding할 수 있다.

(2) Geometrical values

위의 encoding과 함께 area나 perimeter등과 같은 기하학적인 feature들을 또한 계산할 수 있다.[1,2,3]

위의 2가지 Feature를 사용하여 ODB(Object Data Base)를 다음과 같이 구축할 수 있다.

```
Struct Object{
        char * obj_name = " object1 "   /* 물체 이름 */
        char **chain;                   /* chain code */.
        double perimeter;               /* 최외곽 둘레 길이 */
```

```
double area;            /* 면적 */
double moment;          /* 2차 moment */
        :
} obj[max_obj];         /* object arrey */
```

(여기서 *object1* 은 사용자가 입력함.)

Learning기간 동안은 주어진 object의 feature들을 추출하여 ODB내에 Reference Image로 기록하고 Recognition기간 동안은 Test Image를 받아들여 Reference Image와 다음과 같은 비교를 함으로써 Image를 인식하게된다.

이러한 ODB를 사용하여 물체를 recognition하는 방법은 크게 2가지 mode로 나뉠 수 있다. 우선 n차 feature space에서 separable한 경우(즉, 구분하여야할 object들이 많지않아서 몇개의 feature로 주어진 object들이 구분되는 경우) chain code를 사용하지않고 몇개의 feature를 사용하여 주어진 물체를 인식한다. 그러나, 만일 주어진 object들이 너무 많아서 몇개의 기하학적인 feature들로는 구분하기가 힘든경우, 입력된 test image에 대해 최외곽 chain code를 계산하고 이를 ODB의 각 object와 비교함으로서 물체를 인식할 수 있다. 여기서, chain code의 비교는 각 code별로 비교하고 error sum을 계산한 다음 error sum이 일정한 량을 넘지않는 경우 이를 동일한 object라고 인식한다. 만일 일정한 error sum을 넘는 경우 rotation을 고려하여, 1개 code씩 test chain code를 회전시키고 다시 이러한 과정을 반복한다. ODB내의 모든 object의 모든 회전에 대해 error sum이 일정값을 넘으면 주어진 test image와 일치하는 object가 ODB내에 없는것으로 간주한다.

Calibration은 Camera좌표계(C)의 position값을 Robot좌표계(R)의 position값으로 변환시켜주는 작업으로 일반적으로 (C->R) Transformation Matrix를 구하여 사용한다. 본 controller에서 사용하는 중요한 함수의 정의는 아래 표 1과 같다.

표 1. 각 함수의 기능

flag의 번호	함수의 기능	설 명
0	Area Processing	면적을 구하는 기능.
1	Length Processing	물체의 둘레 길이를 구함
2	Center Processing	물체의 중심을 구한다.
3	Angle Processing	물체의 주축의 각도를 구한다.
4	Orient Processing	물체의 Orient를 구한다.
5	Rectangle Processing	물체를 둘러싸는 사각형을 구한다.
6	In Moment Processing	물체의 인식을 위해 먼저 물체를 학습시킨다.
7	Calibrat Processing	카메라의 좌표계를 로보트 Body의 좌표계로 바꾸어 준다.
8	Re_Moment Processing	먼저 학습된 데이타를 가지고 물체를 인식한다.
9	Hole Processing	물체의 구멍의 갯수를 구한다.

제 5 장 다중로보트 시스템을 위한 로보트 제어 언어

본 장에서는 사용자가 요구하는 동작을 로보트로 하여금 수행 하도록 하기 위해서 필요한 Trajectory Planning과 외부의 환경 변화에 대처하기 위해서 필요한 Sensor신호와 결부된 Trajectory 수정 방법에 관해 기술 한다. 여기서 Trajectory란 각 관절들의 가속도, 속도, 위치의 시간에 따른 변화 상태를 말한다. 이 문제는 사용자가 쉽게 원하는 동작을 기술하는 문제를 포함 한다. 예를 들면 사용자가 End_effector의 목표지점과 자세를 지정하고, 경로를 직선, 원호, 2차 함수등의 언어로 지정하면 이를 이용 Computer가 경로의 정확한 모양, 속도 함수등을 계산한다.

또하나의 문제는 Trajectory를 계산 하는데 실 시간 계산이 가능 해야 한다는 점이다. 일반적인 Robot System에서 Trajectory는 20Hz 내지 200Hz 의 주기로 계산된다. 사용자가 원하는 동작을 어떻게 Robot Control System에 전달할 것인가 하는 문제는 "Robot란 Programmable한 Manipulator다" 라는 Robot의 정의를 들추지 않는다 해도 Industrial Application에서 매우 중요한 문제이다. 이러한 Man-Machine Interface의 한 해결책이 Programming Language 이다.

Motion을 위한 Language는 사용자가 쉽게 기억할 수 있어야 하고,경로의 특징, 목표 지점과 자세, 속도등을 기술할 수 있어야 되고, 외부 환경의 변화를 감시, 적절한 Motion의 변경을 기술할 수 있어야 한다. 이러한 기능을 가지고 있는 Motion Statement의 Syntax Diagram 은 다음과 같다.

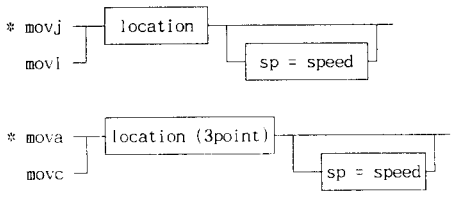

그림 6. motion language grammer

5.1 Robot Cotrol Language

Robot Cotrol Language는 크게 motion, gripper, vision, loop control, algebraic 명령어들로 구성된다.[5]

(1) Motion 제어 명령어

i) movj
- syntax : movj <location> sp=num
- 기능 : 현재의 로보트 위치에서 <location>까지 ptp motion으로 움직인다.

ii) movl
- syntax : movl <location> sp=num
- 기능 : 현재의 로보트 위치에서 <location>까지 linear motion으로 움직인다.

iii) movc
- syntax : movc <location> sp=num
- 기능 : 현재의 로보트 위치와 <location>을 연결하는 직선을 지름으로 하는 원을 따라 시계방향으로 움직인다.

iv) mova
- syntax : mova <location> arg=num sp=num

- 기능 : 현재의 로보트 위치와 <location>을 연결하는 직선을 지름으로 하는 원을 그라 angle값만큼 시계방향으로 움직인다.

(2) gripper 제어 명령어

i) grip_rotate
- syntax : grip_rotate sp=num disp=num
- 기능 : gripper를 displacement값만큼 시계방향으로 돌린다. (4축 구동)

ii) grip_up
- syntax : grip_up sp=num disp=num
- 기능 : gripper를 displacement값만큼 위로 올린다. (3축 구동)

iii) grip_down
- syntax : grip_down sp=num disp=num
- 기능 : gripper를 displacement값만큼 아래로 내린다. (3축 구동)

iv) grasp
- syntax : grasp
- 기능 : gripper를 닫는다. digital I/O board에 gripper에 해당하는 번호에 on을 출력한다.

v) release
- syntax : release
- 기능 : gripper를 연다. digital I/O board에 gripper에 해당하는 번호에 off를 출력한다.

(3) 제어 명령어

i) if 文
- if(expression) then action
  else if(expression) then action
  endif

ii) while 文
- while(expression)
  action

  wend

iii) for 文
- for i_vow=num to num step=num
  action

  next

(4) Vision 명령어

i) vision_init : vision board를 초기화하고 화면을 unthreshold 상태로 만든다.

ii) threshold : autothreshold를 한다.

iii) area : 면적을 구한다.

iv) center : 무게 중심을 구한다.

v) perimeter : 둘레길이를 구한다.

vi) learning : 화면에 잡힌 물체에 대한 moment를 구해서 저장한다.

vii) recognition : learning에서 저장된 data와 비교해서 가장 가까운 물체의 번호를 return한다.

(5) algebraic명령어

<, >, >=, <=, ==, =, AND, OR, +, -, *, /

5.2 Language Interpreter

Language interpreter는 우리가 정해 놓은 language들의 조합으로 이루어진 job file을 해석하여 로보트를 구동하는 실제 함수들을 call할수 있게 해 준다. language interpreter는 token.c와 parse.c로 이루어지는데 token.c에서는 입력으로 들어오는 string에서 우리가 정의해 놓은 language의 module단위가 있을 경우에 token을 생성해 주는 어휘분석기이다. parse.c는 yylex.c에서 넘겨주는 token들의 순서가 우리가 정의한 language의 문법에 맞는지를 해서 맞지 않을 경우에는 error를 출력한다.

language 문법에 위배되지 않을 경우에는 우리가 정의해 놓은 실제 action routine이 실행되게 된다.

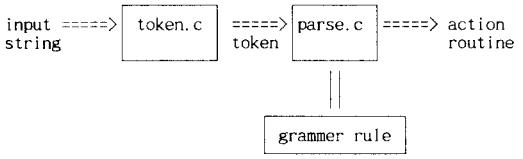

그림 7. language interpreter

제 6 장 실험 및 고찰

앞에서 언급하였던 여러 robot language, control language 및 vision language를 사용하여 그림 8과 같은 위치 보정 실험을 수행하였다. 실험 과정은 다음과 같다.

1) Robot의 end-effetor를 정해진 위치로 이동시킨다.

2) 임의로 놓여진 object를 인식하고 camera 좌표계에서의 중심위치 p를 얻는다.

3) Calibration(C->R)를 수행하여얻은 T-matrix에 p를 곱하여 실제 Robot 좌표계에서의 위치 p'를 얻는다.

3) p'로 이동하여 object를 잡고 Object에 따라 정해진 위치로 이를 이동시킨다.

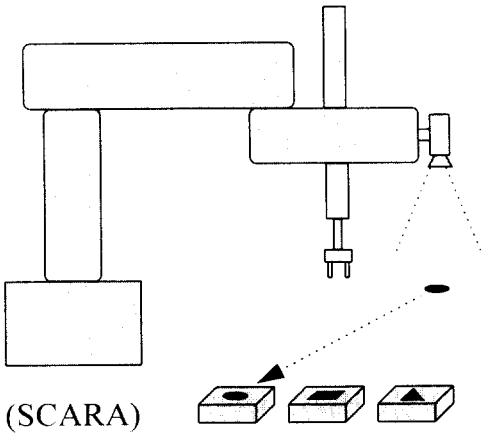

그림 8. vision regconition 실험 환경

여기서, object는 삼각형 사각형 원의 세가지 모양으로 한정하였다. 이를 수행할 수 있는 program은 vision language를사용하여 다음과 같이 만들 수 있다.

```
for i=1 to 50 step=1        /* 50개 물체를 해당 위치에 놓음*/
        movj loc1 sp=40  /*미리 설정된 위치 loc1으로 이동*/
        i1 = recognition  /*인식통해 물체번호를 i1에 저장*/
        loc5 = center    /* 실제물체위치 검사 -> loc5*/
        movj loc5 sp=40       /* loc5로 이동 */
        grip_down sp=40 disp=75  /* gripper를 내림 */
        grasp                    /* 물체를 집는다. */
        grip_up sp=40 disp=75    /* gripper를 올림  */
        if(i1 == 0) then         /* 삼각형의 경우 */
                movj loc2        /* loc2(삼각형)로 이동 */
                grip_down sp=40 disp=75 /*gripper를 내림*/
                release                 /*obj를 놓는다*/
                grip_up sp=40 disp=75   /*gripper를 올림*/
        else if(i1 == 1) then      /* 원의 경우 */
                movj loc3
                grip_down sp=40 disp=75
                release
                grip_up sp=40 disp=75
        else if(i1 == 2) then   /* 사각형의 경우 */
                movj loc4
                grip_down sp=40 disp=75
                release
                grip_up sp=40 disp=75
        endif
next
end
```

제 7 장 결 론

본 연구는 Vision based Intelligent Robot System의 개발에 관한 것으로, 본 연구를 통하여 Vision 로보트를 구성하는 배전부및 전자 제어부의 설계, 제작 및 시험을 수행하고, 또한 상품화에 대한 가능성을 검토하고 핵심 기술의 개발 기법을 취득하는데 주목적을 두었다. 본 연구의 결과를 검토하면 다음과 같다. (1) Vision based Robot Control를 위한 계층적 제어 시스템 개발 (2) 다중 로보트의 효율적인 응용을 위한 기본 Motion Algorithm개발. (3) 다중 센서 처리를 위한 H/W, S/W 개발. (4) 다양한 X-based graphicel Menu 개발. (5) Vision, I/O, 및 기본 Motion 등의 language화 등이다.

본 Robot Controller는 Real-Time based Lynx O.S.을 기반으로하여 각 Motion이 원활한 동작을 수행할 수 있도록 설계하였다. 특히 기본 Motion 수행중 각종 외부 I/O signal 및 sensor signal의 처리를 기본 Motion에 지장을 주지 않도록 Multi-Processing기법을 하였다.

(Reference)

[1] Sing-Tze Bow, "Pattern Recognition and Image Preprocessing," Dekker, 1992.

[2] Rafael C.Gonzalez and Paul Wints, "Digital Image Processing," Addison Wesley Publishing Company, Second Edition, 1987.

[3] Freeman, H., "On the Encoding of Arbitrary Geometric Configurations," IRE Transactions on Electronic Computers, Vol. EC-10, 1961, pp. 260-268.

[4] R.M.Haralick, L.G.Shapiro, "Computer and Robot Vision," Addison Wesley, 1993.

[5] I.H.Suh *et. al.*, "Design and Implementation of a Dual Arm Robot Control System with Multi-Sensor Integrating Capability," *Proc. of IECON'91*, **2**, 898-903 (Oct. 1991).

[6] J.E.Agapakis *et. al.*, "Programming and Control of Multiple Robotic Devices in Coordination Motion," *Proc. IEEE Int. Conf. on Robotics and Automation*,**1**, 362-367 (May. 1990).