'87 韓國自動制御學術會議論文集 1987. 10.16~17

로보트와 Vision System Interface

김 선 일, 어 인 택, 박 찬 웅
한 국 기 계 연 구 소 자 동 제 어 실

Robot Vision Interface

S. I. KIM , I. T. YEA , C. W. BAHK
Automatic Control Lab., Korea Institute of Machinery & Metals

Abstract

This paper shows the robot-vision system which consists of robot, vision system, single board computer and IBM-PC. IBM-PC based system has a great flexibility in expansion for a vision system interfacing. Easy human interfacing and great calculation ability are the benefits of this system. It was carried to interface between each component. The calibration between two coordinate systems is studied. The robot language for robot-vision system was written in "C" language. User also can write job program in "C" language in which the robot and vision related functions reside in the library.

1. 서론

공장 자동화의 보급과 더불어 로보트 사용이 증가되고 작은 부품의 조립에는 SCARA형 robot 가 많이 쓰이고 있으며 대부분이 playback type으로 동작되고 있다. 그러나 playback type은 정확한 치구 및 feeder를 요구하고 있고 이로 인해 오히려 주변장치 제작에 더 많은 돈과 시간이 소모되어 자동화 system의 flexibility를 떨어뜨리는 요인이 되고 있다. 이런 문제를 해결하는 한 방안으로 robot vision을 들 수 있으며 이에따라 기존의 SCARA robot에 vision system을 적용시켜 vision guided assembling이 가능하도록 하였다. 이의 구현을 위해 vision system, IBM-PC, Single Board Computer(SBC), robot로 이루어지는 robot-vision system을 구성하였으며 각 component 간 interfacing, calibration 및 language 작성등이 이루어졌다.

2. 전체 System 구성

전체적인 system은 그림1과 같이 vision system, robot 몸체, robot controller 및 system controller 로 구성되어 있다. vision system은 CCD 카메라에서 받아들인 영상신호로 부터 선택된 feature를 뽑아내어 해당 물체의 이름, x-y좌표값, orintation값을 계산해 내며 system controller 는 user interface 및 user program editing, compiling, execution을 행하며 calibration, jog, teaching, 좌표변환, job program

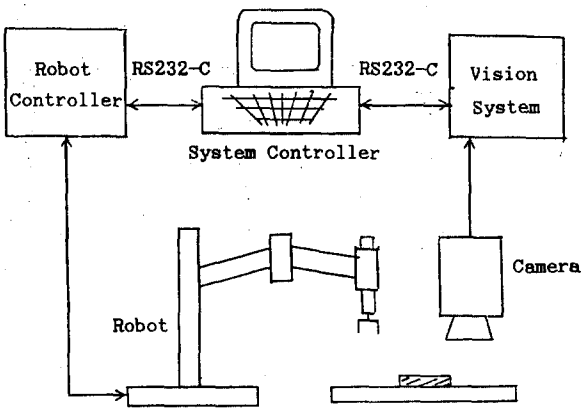

그림.1. Robot-Vision 시스템 구성
Fig. 1. Construction of Robot-Vision System

등을 수행한다. controller 부분은 그림2와 같이 SBC, pulse 분배기, position controller 및 servo pack 으로 이루어지며 SBC는 IBM-PC로 부터 command를 받은후 그에 해당하는 command를 pulse분배기에 주게 된다. system의 구성은 vision system interface가 가능한 방향으로 이루어지도록 하였다. 즉 기존의 controller는 확장이 어려웠기 때문에 위치 제어기 및 servo pack 만 이용하고 pulse 분배기를 제작하여 SBC로 부터 받은 command에 따라 해당 pulse를 분배하게 해주고 각종 명령은 SBC로 부터 나오도록 구현 하였다. SBC는 user interface 및 계산기능이 취약하므로 이를 IBM-PC로 대치 하였으며 IBM-PC의 user interface 기능 및 계산능력을 이용하여 vision system 으로 부터 받은 좌표값을 robot 좌표계로 변환한 다음 SBC에 해당 command를 주게된다. vision system 및 SBC와 IBM-PC와는 RS-232C로 이루어져 있으며 통신속도는 9600 baud rate이다. software 구조는 그림3과 같이 teach program, calibration program, user program 및 library program으로

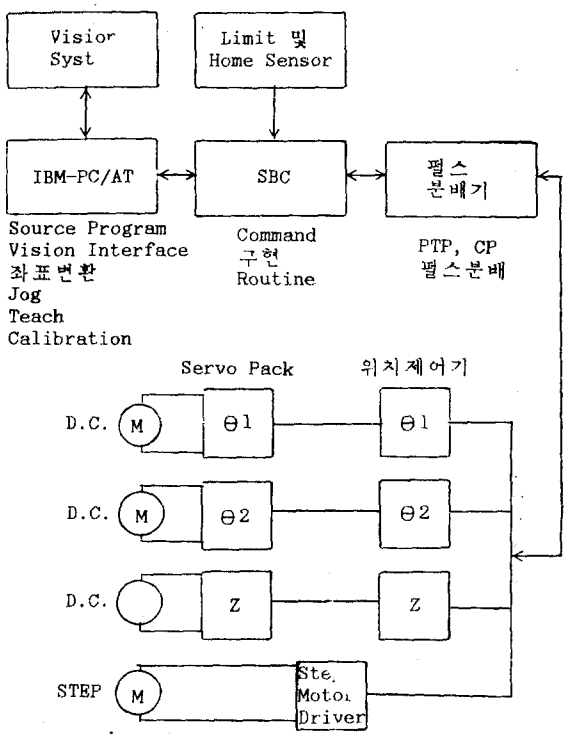

그림 2. Controller 의 구성
Fig. 2. Construction of Controller

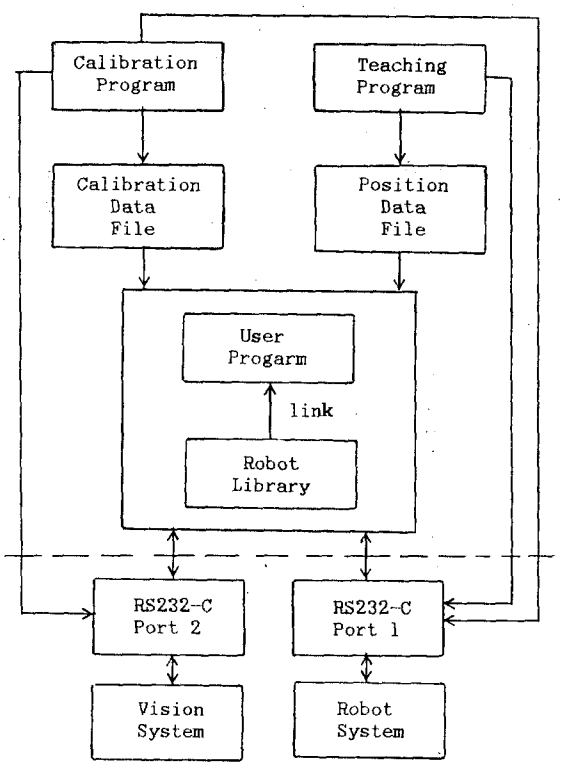

그림 3. 소프트웨어의 구조
Fig. 3. Structure of Software

구성되어 있다. teach program은 RS232-C port1을 통해서 robot를 동작시키며, 위치 데이타를 화면에 display하고 teaching된 각 축의 위치 데이타 값을 teaching된 위치 갯수와 함께 position 데이타 file에 저장한다. calibration program은 RS232-C port1과 port2를 통해서 robot와 vision system을 각각 제어하며 이렇게 얻어진 데이타 값으로 각 좌표간의 관계 변수들을 구하여 calibration 데이타 file에 저장한다. library program은 robot와 vision system을 구동하는데 필요한 각종 command로 이루어진 function으로써 user program과 link된다. user program은 위치 데이타와 calibration 데이타를 읽어서 user가 원하는 임의의 동작을 취하게 된다.

3. Vision System

본 연구에서는 vision system으로 일본 Fuji 전기에서 만든 VR-1000 robot vision system을

사용하였으며 사양 및 protocol은 다음과 같다.

1) 사양

ㄱ) 접속카메라 대수 : 2대

ㄴ) 입출력 및 channel

a) panel key unit interface

b) block 전송 channel(RS-232C)

: 외부 통신용

c) character 전송 channel(RS-232C)

: printer 출력용

d) 외부 trigger 입력

e) 장치 이상 출력

f) user monitor 출력

ㄷ) 처리 분해능 : 256 X 220

ㄹ) 학습등록부품수 : 최대 32개

ㅁ) 1화면중의 처리 부품수 : 최대 16개

ㅂ) 최대 32개의 window 등록

ㅅ) menu

a) mode

b) system set up

c) 학습

d) 인식

e) utility

2) Protocol

데이타 및 command는 block 전송 방식을 이용하여 text는 앞에 STX, 종료시에 ETX 신호를 주며 각 character를 XOR하여 block error를 check한다. text block 전송 protocol은 그림4과 같다.

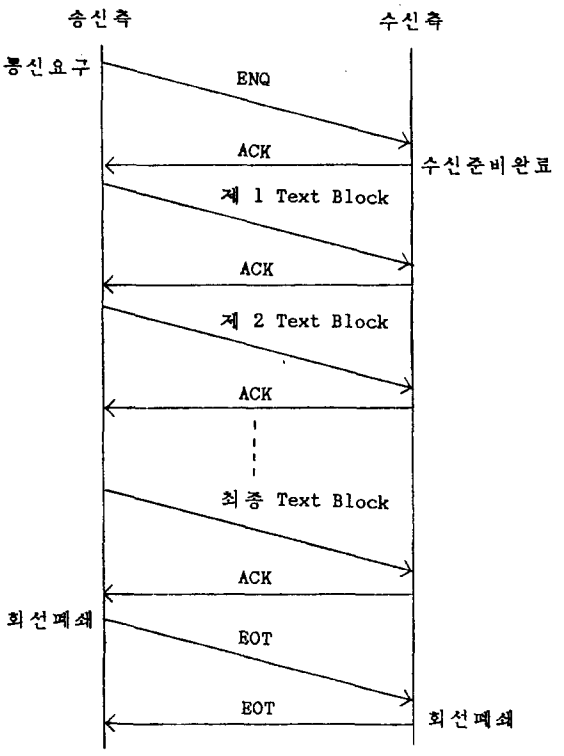

그림 4. Block 전송 Protocol
Fig. 4. Block Transmission Protocol

4. SCARA robot

vision system과 연결한 robot는 삼성항공산업의 WISEMAN SPR-410 SCARA robot으로 1축의 길이가 230mm 2축의 길이가 180mm이고 PTP운동을 하는 playback robot이다.

5. Calibration

robot와 vision system은 서로 좌표계가 일치되어 있지 않으므로 vision system에서 구한 x,y좌표는 robot 좌표계에서의 값으로 변환되어야 한다. 이 좌표 변환에는 그림5에서와 같이 각 좌표계의 원점간의 거리 a,b 및 rotation Θ, scale factor α, β 를 구하여야 한다. 이 값들을 구하기 위해서는 vision 좌표계의 원점에서의 좌표치(x,y)와 robot 좌표계에서의 좌표치(x,y)은 임의의 공간좌표 P라는, 공간상의 같은 위치라는 사실과 이들 좌표계간의 관계식을 알면된다.

즉 $x' = a + \alpha y\cos\Theta - \beta x\sin\Theta$ ---- [1]

$y' = b + \alpha y\sin\Theta + \beta x\cos\Theta$ ---- [2]

실제로 임의의 물체의 중심에 robot의 hand 중심이 오도록 움직인 다음에 그때의 좌표치가 $(x1', y1')$이고 vision으로 이 물체의 중심 좌표치를 구했을때 좌표치가 (x1,y1)이며 이와 같은 과정을 두번 더 되풀이한다음 공간상의 직선의 길이를 이용하여 scale factor α, β 를 [3],[4] 식으로 구할 수 있다.

$$\alpha^2 yld^2 + \beta^2 xld^2 = xld'^2 + yld'^2 \quad \text{---- [3]}$$

$$\alpha^2 y2d^2 + \beta^2 x2d^2 = x2d'^2 + y2d'^2 \quad \text{---- [4]}$$

(xld = xl − x2 , y2d = x2 −x3 etc.)

또한 [1], [2] 식으로부터 $\sin\Theta$, $\cos\Theta$ 는

$$\sin\Theta = -\frac{\beta(xl'-x2')(xl-x2) - \alpha(yl'-y2')(yl-y2)}{\alpha^2(yl-y2)^2 + \beta^2(xl-x2)^2}$$

$$\cos\Theta = \frac{\alpha(xl'-x2')(yl-y2) + \beta(xl-x2)(yl'-y2')}{\alpha^2(yl-y2)^2 + \beta^2(xl-x2)^2}$$

로 구할 수 있다.

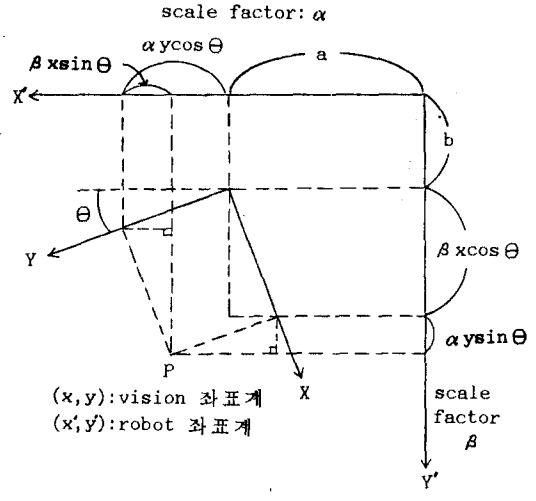

그림 5. 좌표계 사이의 기하학적 관계
Fig. 5. Geometric Relation between Coordinates

위에서 언급한 calibration을 하려면 pulse 갯수로 표현되는 robot의 위치를 kinematic equation을 이용하여 cartesian 좌표계로 나타내어야 하고 vision 좌표계에서 구한 x,y값을 inverse kinematic equation을 이용하여 pulse 갯수로 바꾸어 주어야 하는데 이를 위해서는 정확한 robot 팔길이가 주어져야 한다. SCARA robot의 팔길이 calibration 방법으로는 3점 측정법(three points calibration method) 및 4점 측정법(four points calibration method) (1)이 있으나 digitizer를 사용하여야 하며 실제로 측정시의 error 또한 측정위치에 따라 다르므로 calibration 과정을 거치지 않고 가공치를 그대로 사용했다. 그러나 cartesian 좌표계에서 원점복귀 했을 때 1축 arm과 2축 arm이 일직선상에 있어야 하는데 이는 공간상의 한 좌표 P에 대해 2가지 자세를 가질 수 있고 이때 2축 arm은 1축 arm의 연장선에 대해 서로 같은 크기의 각도를 가지므로 쉽게 구할 수 있다.

6. Language 구현

연구에 사용된 기존의 로보트는 interpreter형 언어로 되어 있으며 vision system을 support할 수 있는 language 구조가 형성되어 있지 않아서 system 자체의 재구성과 아울러 언어구조의 재구성이 필요하였다. (2)는 현재의 로보트 시스템이 요구하는 high level 언어구조를 만족하면서도 어떤 특정한 언어를 새로 구성하는 부담을 덜 수 있는 방법을 제시하고 있다. 이에 따라 앞으로의 확장성, 새로운 언어구성으로 인한 부담감소 sensor integration 등에서 유리한 방법으로 기존의 "C"언어를 사용하여 로보트 동작 및 vision system에 관련된 부분만 function 형태로 작성하여 library화 시켜 사용자가 불러 쓸 수 있도록 구성하였다. 이 방법은 기종의 "C" compiler에 어떤 추가나 변경없이 필요한 기능만 library에 심으면 되므로 언어개발의 부담이 없다. 다음 example 1은 교시를 통해서 "p.dat" file에 save해 둔 위치 데이타를 읽어서 첫번째, 두번째, 세번째 위치로 가고 keyboard에서 'q'가 들어 오지 않는한 계속 반복하게 하는 program이다. example 2는 위치 데이타 file을 읽고 좌표반환에 쓰이는 calibration 데이타 file 을 읽은 다음에 해당위치로 move하고 카메라에서 대상 물체 위치를 파악하여 좌표변환한 후 Z축과 S축은 교시된 위치값을 넣어주어서 해당 물체로 움직이는 program이다. 역시 'q'가 들어오지 않으면 반복한다.

7. 결론

robot-vision system을 구현하는데 있어 제일 큰 문제는 역시 calibration 이다. calibration시에 사용하는 기구로 약 2mm정도의 두께를 갖는 원을

사용하였는데 원의 실제 중심과 vision system에서 잡은 원의 중심이 일치하지 않는 경우가 많았다. 즉 좌표간의 관계 상수를 구하기 위해 원의 중심에 robot를 이동시켜 이 위치를 저장하고 robot를 카메라의 field of view 바깥으로 이동시킨후 vision system으로 원의 중심을 구하여 관계식에 따른 계산을 하게 되는데 조명의 불균일 등으로 그림자가 한쪽으로 몰리게 되고 그만큼 원의 중심이 이동을 하므로 실제로 공간상의 서로 다른점을 같은점으로 보게 되어 위치오차의 원인이 된다. 이를 해결 하기 위해서는 calibration 기구를 알게 제작하고 물체의 이동에 따른 그림자의 변형이 없는 조명조건을 갖추어야 할 것이다.

example 1.

```
#include <SCARA.h>
main()
{
  position  pos ;
  int       con, speed ;

  speed = 6 ;
  pos = getp("p.dat") ;
  do {
    move(pos.p[0],speed) ;
    move(pos.p[1],speed);
    move(pos.p[2],speed);
    con = key() ;
  } while (con !=  'q') ;
}
```

example 2.

```
#include <SCARA.h>
main()
{
  position  pos, vpos ;
  int       con, speed ;

  speed = 6 ;
  pos = getp("p.dat") ;
  getcal("cal.dat") ;
  do {
    move(pos.p[0],speed) ;
    move(pos.p[1],speed);
    vpos = vision() ;
    vpos.p[0]._posz = pos.p[1]._posz ;
    vpos.p[0]._poss = pos.p[1]._poss ;
    move(vpos.p[0],speed);
    con = key() ;
  } while (con !=  'q') ;
}
```

참고문헌

1. Nobuyuki FURUYA and Hirishi MAKINO, "Calibration of SCARA Robot Dimensions by Teaching", 精密機械, 49巻 9号, pp69-74, 1983

2. Vincent Hayward and Richard P. Paul, "Robot Manipulator Control under Unix RCCL: A Robot Control "C" Library", Vol. 8, No. 4, pp94-111, 1986