'92 韓國自動制御學術會議論文集 1992. 10. 19~21

센서 통합 능력을 갖는 다중 로보트 제어 시스템의 개발

서 일홍*, 현 웅근*, 김 태원*, 여 희주*,
김 재옥**, 윤 승중**
* : 한양대학교 전자공학과
** : 삼성항공

Development of a Multi-Robot Control System with Sensor Integrating Capability

I.H. Suh*, W.K. Hyun*, T.W. Kim*, H.J. Yeo*,
J.O. Kim**, and S.J. Yoon**
* : Dept. of Electronics Eng. Hanyang Univ.
** : Samsung Aerospace Co.

요 약

본 논문에서는 다중 로보트의 협조제어(Coordinated Control)를 위한 로보트 콘트롤러의 설계에 대해서 연구한다.

첫 부분에서는 다중 로보트의 연구배경 및 연구동기에 대해서 논의하고, 이어서 Coordinated Task를 묘사하기 위한 Programming Primiitive Set을 정의하며 구현에 대해서도 논의한다.특히 Motion Primitive는 Synchronous(Coordinated Motion), Asynchronous Motion, Conditional Motion,특수 Motion으로 분류하고, 각각의 궤적계획 및 구현에 대해서도 간단히 논의한다.

특히 본 논문에서는 외부의 변화하는 환경에 효과적으로 적응할 수 있게 하기 위하여 Vision 센서, Encoder 신호와 Limit 센서, Force 센서 등의 다양한 외부 센서를 융합 처리할 수 있는 다중 로보트 제어 시스템을 개발하였다.

1. 서 론

최근 생산성 향상을 통한 국제 경쟁력 강화를 위해 산업용 로보트를 응용한 다중 로보트 제어 시스템 [1] - [3] 의 개발에 점점 더 관심이 집중되고 있다. 이러한 시스템의 잠재적인 응용이 보다 정교한 조립작업의 자동화를 포함하여 전 공장의 무인화 까지를 목표로 하고 있으며 나아가서는 심해작업이나 우주공간에서 작업의 로보트화를 요구하고 있는 실정을 감안할때 이들 요구에 부응하기 위해서 가장 핵심적인 연구인 복수개 로보트 및 다중센서를 제어할 수 있는 다중 로보트 시스템 제어장치의 개발은 반드시 필요하게 될 것이다. 더군다나,움직이는 장애물과 같은 환경의 변화를 고려할 때, 센서 정보에 의해 환경을 묘사하는 방법[4] - [6]이 필요하다. 시각, 거리, 근접, 접촉 및 힘 센서 등과 같은 유용한 센서들에 의해서 다양한 정보를 얻을 수 있다. 따라서, 다중 로보트 제어 시스템을 운영하는 데 있어서 운동 계획, 협조제어 능력과 함께 특히 필요한 기능은 센서 정보의 융합과 통합으로 생각할 수 있다. 이 문제를 해결하기 위해서, 제어 시스템은 Multi-Robot이 협조제어를 할 수 있도록 해야 하고 다중 센서를 융합할 수 있어야 한다. 그러나, 현재 상품화된 거의 모든 로보트 제어 시스템은 기본적으로 사용자가 한 대의 로보트 만을 사용할 수 있게 한 것 같고 적절한 제어 알고리즘과 계산 방법의 미흡으로 인해 효과적으로 다중 센서를 융합할 수 없다.

본 논문에서, 다중 센서 통합 능력을 가지는 다중 프로세서에 근거한 다중 로보트의 제어 시스템을 구현하는 방법을 설명하고자 한다. 특히, 전체 제어 시스템은 동시에 12축을 제어할 수 있으며 시각 센서, 힘 센서 그리고 외부 컨베이어의 위치 엔코더와 같은 환경에 적응할 수 있도록 설계했다. 또한, 이 제어 시스템은 사용자가 티칭 박스, 풀 스크린 에디터를 가지는 고급 언어 그리고 메뉴 구동명령어-특히 다중 로보트 제어를 위해 설계된 모든 것- 를 경유해서 시스템과 통신을 할 수 있도록 해준다. 이 제어 체계는 장애물 회피, 컨베이어 트랙킹 그리고 다중로보트를 제어할 때 필요한 동기 운동과 같은 특수한 기능을 수행할 수 있다.

제안된 설계 개념을 입증하기 위해, 제어 시스템이 그림.1,2,3 에 보여진 것처럼 다중 로보트 제어 시스템, 비젼 시스템과 컨베이어 시스템으로 두 대의 SCARA 로보트가 주사위 맞추는 작업을 성공적으로 수행할 수 있었다.

2. 다중 로보트 제어 시스템의 구성

2.1. 전체 시스템의 구성

본 연구에서는 다음과 같은 여러 개의 프로세서로 나누어 분산처리 구조를 갖도록 하여 확장성 및 유연성이 높은 시스템이 되도록 구성하였다.

먼저 시스템 전체를 관리하며 언어 및 로보트 동작의 교시 그리고 자기 진단등의 기능을 하는 Supervisory Processor, 물체의 위치와 자세 및 형태를 인식하는 Vision Processor, Man-machine Interface를 위한 Display Processor, 그리고 로보트의 제어를 담당하는 로보트 제어 Processor등으로 구성되어 있고, 이들 각각의 통신은 Bus Arbitration에 의한 VME Global Bus를 통해 공유 메모리(Common Memory)를 이용하여 수행한다.

특히 Supervisory System은 Real Time O.S.인 XINU를 사용하여 구현하였고 Servo 시스템은 Interrupt 방식을 사용하여 구현하였는 바, 전체 구성은 그림 1과 같다.

2.2. 제어 시스템의 H/W 구성

본 연구에서는 여러개의 프로세서로 나누어 분산처리 구조를 갖도록하여 확장성 및 유연성이 높은 시스템을 구성하였다. 이에 대한 Hardware 구성은 그림 4와 같다.

2.3. 제어 시스템의 S/W 구성

개발된 Software는 C언어에 기반을 두고 모든 Program을 세분화하여 계층적 제어구조를 이루게하고 수정 및 편집을 단위 Program Module별로 할 수 있게 하는 새로운 형태의 매니퓰레이터 레벨 로보트 제어언어를 설계하였다. 이에 대한 제어 시스템의 S/W 구조는 그림 5와 같다.

3. 다중 로보트 제어 시스템의 기능별 구성

3.1. Supervisory System

Supervisory System을 설계하는 데 있어서, Supervisory는 각 Slave에 작업을 효과적으로 할당하여 수행시켜야 하고, 또한 조작자에게는 로보트 언어와 같은 사용자에게 친숙한 Man-machine Interfacing을 준비하여 시스템 Set-up뿐만 아니라 운영상의 실수를 줄일 수 있도록 고려되어야 한다. 또한, 전체 시스템의 원활한 유지를 위해 자기 진단 기능이 필요하다. 이들 요건을 충족 시키기 위해서, 본 시스템에서는 다음과 같이 6가지 모드를 설정하였다.

(1) Diagnosis Mode (2) Set-Up Mode (3) Teach Mode (4) Edit Mode (5) Look Mode (6) Run Mode

(1) Diagnosis Mode

진단 모드에서는 우선 Supervisory System을 메인 하드웨어 시스템 (CPU 보드 , Common Memory 보드, Motor Servo Modules), 주변 하드웨어 시스템(Floppy Disk Driver, Printer, Teaching Box)등의 상태를 확인하고, 그것들의 현 상태를 표시한다.

(2) Set-up Mode

Set-up 모드에서는 응용시에 제어 시스템의 융통성을 부여하기 위해서 여러가지 파라미터를 Set-up하거나 수정할 수 있다. 본 시스템에서는 각 로보트 파라미터(로보트 타입 , 기구학 변수, Motor 정격), 좌표 원점 파라미터(홈 위치, 속도, 오프셋)와 시스템 파라미터(Base 좌표계, Tool 좌표계, 위치 레벨, 전용 I/O , 외부 장치)를 설정한다. 그래서 이들 입력 파라미터를 조정하여 3가지 형태의 로보트(X-Y-Z 직교형,수평다관절형, 수직다관절형)를 제어할 수 있다.

(3) Teach Mode

로보트의 End-effector를 정의된 여러 좌표계를 기준으로 이동시켜, 로보트의 "위치"와 "방향"을 Memory에 기억시키고 각 점의 이름을 할당하여 운동 계획에 사용한다.

본 연구에서는 여러 좌표계(Robot Base, User_Defined, Tool_coord)를 준비해서 Teaching을 용이하게 하는 한편, CAD Data를 직접 입력하는 MDI(Manual Data Input)와 Motor의 전원을 끈채로 로보트를 직접 이동시켜 Teaching하는 Free Teach Mode가 있다.

Teaching Box는 두대의 로보트 및 외부축을 조작하고, 작업에 대한 궤적을 지시, 수정하며 실제로 로보트의 작업을 편집할 때 로보트 언어중 MOVE 관련 명령어 (MOVJ, MOVL, MOVA, MOVC)들을 편집, 등록 , 수정할 수 있다. 아울러, 로보트의 상태 표시 기능으로 I/O Device의 상태를 표시, 제어할 수 있다.

(4) Edit Mode

본 시스템에서 사용하는 로보트 언어가 High Level Language에 가까우므로 본 에디터는 PC의 풀키(Full Key)를 사용하여 프로그램을 작성할 수 있고, 또한 많이 쓰이는 명령어의 편리를 위하여 로보트 언어 MACRO 기능을 갖추고 있다. 본 에디터는 Full Screen Editor로 Doubly Linked List Structure로 편집중인 Line을 가리키는 Pointer는 Text를 위한 Buffer와 시작과 끝을 가리키는 Pointer를 가지고 있다. 그리고, 본 에디터가 갖고 있는 주요기능으로 Search & Replace, Block Copy, Move, Delete, Block Print, File Print, File의 Read/Write등이 있다.

(5) Run Mode

Supervisory System은 프로그램된대로 자기 자신의 작업을 수행할 뿐만 아니라, 상태를 Monitor하면서 각 Servo에 필요한 작업을 적절히 할당한다. 프로그램의 정확성을 확인하기위해서, 세가지 형태로 프로그램을 수행할 수 있도록 하였다. 즉 단계별 동작 확인을 위한 단계별 수행(step run), 처음 단계에서 마지막 단계까지 동작 확인을 위한 사이클 수행(cycle run)과 동작 확인 후 반복 동작을 위한 자동 수행(auto run)이다. 충돌 회피와 두 대 팔의 협조 작업을 효과적으로 조작하기 위해, Supervisory는 궤적을 계획하고 계산하여, 이를 Common Memory를 통해 각 Servo System과 통신한다. 또, 연속궤적제어의 부드러운 동작을 보장하기 위해 각 관절 동작이 계획되어질 때 자동적인 가감속이 고려되어야 한다. 이를 위해, 2차 이산 Shaping Filter가 직각 공간에서 궤적을 보장하기 위해 적용된 후 관절 궤적으로 변환된다.

(6) Look Mode

Look Mode에서는 크게 선처리와 후처리로 나눌 수 있다.선처리는 카메라 선택 (4 대중 한대를 선택), Image grab, Snap, Threshold를 이용한 Binary Processing등의 기능이 있고, 후처리에서는 윈도우 처리와 물체 학습, 저장 기능이 있다. 물체 학습은 각 물체의 특성에 따라 Chain Code, Projection, Matching등 여러가지 방법을 선택할 수 있다. 학습된 물체의 특징량은 그 이름과 함께 Common Memory에 저장된다. 이 데이타는 Supervisory CPU가 물체를 인식하여 동작할 때 쓰인다.

이들 기능 모두를 효과적으로 수행하기 위해, Supervisory 컨트롤러는 Real Time O.S.인 XINU를 사용하여 구현하였다.

3.2. Vision 시스템

비젼 시스템은 VME 사양에 맞게 설계된 SVS900-DT Board를 사용하였는데, 이는 해상도가 512 × 512 이고, 명암도가 256단계이다. 본 비젼시스템은 물체인식, 위치검출의 역할을 담당하고 있으며, 또한 최대 4대의 카메라를 사용할 수 있도록 함으로써 필요부분에 설치하여 효율을 높일 수 있도록 하였다. 이를 위한 제어명령어가 Table 1에 정의되어 있다.

3.3. 로보트 제어 시스템 (Arm Control System)

이 시스템은 Supervisory 시스템에서 받은 로보트 명령에 대해 Inverse Kinematics을 풀고, 이를 관절 운동으로 변환하여 Actuator 시스템을 구동시켜 실제로 로보트 Arm을 제어한다. 제어 장치는 Position 보드와 Power Amp로 구성되어 있으며 최대 12축까지 제어할 수 있다. 이 장치는 매 Solution 시간마다 다음의 4가지 일을 수행한다.

1) 원하는 궤적 정보을 얻고 로보트의 현 상태를 알기 위해 Supervisory와 통신
2) 관절 궤적을 구하기 위해서 Inverse Kinematics 변환과 현재 위치 표시 위해 Direct Kinematics 변환
3) 관절 궤적 계획(Pulse Generation and Exponential Filtering)
4) 매 5msec당 펄스 명령을 분배.

4. 주변기기 및 센서 인터페이스 시스템의 기능별 구성

4.1. Man_machine Interface System (VFIOC)

VFIOC (Video/Floppy/IO Card)는 VME 사양에 맞게 디자인된 Man_machine Interface Board이다. 이는 PC Bus Converter Interface, Floppy Interface, Parallel Interface, Key_board Interface, Key_pad Interface의 Module을 내장하고 있으며, 이에 대한 Block Diagram은 그림 6과 같다.

(1) PC Bus Converter Interface

PC Bus Converter Interface는 Motorola IO Channel Bus의 Signal을 IBM PC/XT Bus의 Signal로 전환시키는데. 이때

Address Expansion Register를 사용하여 Address Window를 확장시켜 준다.

(2) Floppy Interface

Floppy Interface는 3.5″ Floppy Drive와 IBM PC Data Format을 사용한다. 이의 구현은 XINU Kernel의 Read, Write 및 Format기능을 이용하였다.

(3) Key_board Interface

Key_board Interface는 IBM PC/AT Key_board로 부터 Data를 받아 Host로 보내는데, 이때 XINU Kernel을 이용하여 Read, Write 및 Status Report 기능을 제공한다.

(4) Key_pad Interface

Key_pad Interface는 16개의 Push_button을 인식하여 간이 Key_board기능을 제공하는데, 이때 여러개의 Key가 눌려지는 경우, 최초의 Key_data를 Latch한다.

4.2. 다중 로보트용 Multi-Functional Teach Box

본 연구에서 개발한 Teaching Box는 두 대의 로보트 및 외부축을 조작하고, 작업에 대한 궤적을 지시, 수정하며 실제로 로보트의 작업을 편집할때 로보트 언어 중 70% ~ 80%를 사용하게 되는 MOVE 관련 명령어(MOVJ, MOVL, MOVA, MOVC)들을 편집, 등록, 수정할 수 있다. 아울러, 로보트의 상태 표시기능으로 I/O device의 상태를 표시, 제어할 수 있으며 또, 로보트의 위치 및 자세를 사용자가 선택한 좌표계를 기준으로 나타낼 수 있는 등 기존의 단순한 Teaching Box 기능보다 크게 향상시켰으며, Teaching Box의 Display인 LCD Display를 이용하여 Man_machine Interface 부분을 향상시켜 사용자로 하여금 보다 편리하게 명령어의 편집, 등록 및 수정 등을 할 수 있게 하였다.

본 연구에서는 개발한 Teaching Box의 기능을 다음과 같이 크게 5가지로 대별된다.

1) 교시 기능 및 축 조작 기능
2) 궤적의 확인, 실행 기능
3) 명령의 등록 및 편집, 수정 기능
4) 로보트의 상태 표시기능
5) 긴급사태시 조작 및 안전기능

4.3. 다중 센서 인터페이스 시스템

로보트 작업 내용이 복잡해지고, Motion 중심이 아닌 Task 중심의 프로그램을 하게 되면 로보트는 항상 외부 변화를 감시하고 이에 대응하는 동작을 수행해야 한다. 따라서 환경 변화를 감시하기 위해 센서가 필요하고 센서 신호에 의존하여 Trajectory Planning을 하는 Motion이나, Motion을 수정하는 기능이 필요하다.

본 시스템에서는 센서 처리 보드를 준비하여 16개의 Digital Input Signal을 Check하는 Din(Digital input), Digital Signal을 출력하는 Dout(Digital output), Conveyor Tracking을 위해 Conveyor Encoder 신호, Vision 신호등을 인식한다. Digital Signal 입력 장치로는 MACRO6744 Board, 출력 장치로는 MACRO6745 Board를 이용하였다.

4.4. Conveyor Tracking

Conveyor 동기 운전기능이란 Conveyor 이동량을 고려한 궤적계획에 의해서 Conveyor가 정지한 상태로 교시한 궤적을 동작중인 Conveyor위로 재현하는 기능이며, 궤적 추종을 위한 Conveyor의 이동량의 검출은 Encoder의 Feedback Pulse 누적치로 계산된다. 보정기능의 하나로 속도 변동에 따른 추종오차를 줄이기 위하여 평균 속도를 일정 주기마다 (Ts=40msec) Sampling하여 Extrapolation을 통해 다음 주기에 가야할 양을 미리 계산하여 추종한다. 또한, 동기동작의 개시순간 추종지연에 따른 오차를 줄이기 위하여, 동기 개시시 추종지연시간 후의 오차량을 계산해 Conveyor 방향으로 추종오차량을 보정한다. 전체 Conveyor System구성은 그림 7과 같다. 사용된 Sensor는 위와같이 평균속도 측정용, 동기개시용 Limit Sensor가 있어 Conveyor상의 Object가 평균속도 측정 Limit Sensor에 닿았을경우 기준이 되는 Conveyor의 속도를 구하고, 동기개시 Limit Sensor에 닿았을때 동기 동작이 시작된다. 그리고, Conveyor동기 동작을 위해 구현된 Language는 다음과 같다.

```
CV_SPCHCK  CV#                              ①
CV_SYNCH   ROBOT# CV#  T=time              ②
   MOVS    ROBOT# TO LOC WITH SP=100        ③
      ..
      ..
CVEND                                       ④
```

① : Number(#)에 해당하는 Conveyor의 속도를 구한다.
② : Conveyor의 동기 동작을 개시한다. 만약, Time초 이내에 Object가 Sensing되지 않는다면 ④으로 벗어나고, 그렇지 않으면 다음 Command를 수행 한다.
③ : Robot와 Conveyor의 합성속도를 고려한 궤적 추종 동작을 한다.
④ : 동기 동작을 마친다.

5. 다중 로보트 시스템을 위한 로보트 제어 언어

본 연구에서 개발한 로보트 제어 언어는 한대의 로보트를 위한 상용화된 로보트 언어을 포함하여 전체 67가지의 명령어을 제공하고 있다(Table 1). 우선,관절운동(PTP)을 위한 MOVJ, 연속 궤도 운동(CP)을 위한 MOVL, 주어진 세점을 통과하는 원호 운동을 위한 MOVC, 현재 위치에서 주어진 오프셋 거리만큼 증분치 운동을 위한 MOVI와 같은 기본 운동 명령어를 제공한다. 그러나, 이들 기본 명령어만을 이용하여 두대의 로보트가 함께 일하는 것이 실제로는 어려우므로, 본 시스템에서는 Master & Slave 운동 명령어를 준비하였다. 한대 로보트를 Master로 다른 한 대를 Slave로 정의하여, Slave 운동이 Master의 운동을 추종하도록 한다. 특히, MOVJ을 제외한 모든 기본 운동 명령어는 Table1.에서 보는 바와 같이 Master/Slave 운동으로 사용할 수 있다. Master/Slave 운동의 동기와 손목 힘 센서를 사용하여 유연성 있는 운동을 제어하는 것이 필요하다. 그러나, 본 연구에서는 아직 힘 제어 알고리즘이 구현되지 않았지만, 다음 연구에서는 개발될 것이다. 따라서, 이 시점에서는 Master/Slave 운동은 단지 위치 Servo 알고리즘만을 이용하여 제어하였다.

또한 두대 로보트의 동시동작(Concurrent Motion)을 위하여COBEGIN and COEND 명령어[11]를 준비하였다. COBEGIN과 COEND 선언 사이의 모든 동작명령어는 두대로보트가 동시에 자기자신의 컨트롤러에 의해 제어되는것처럼 동시에 수행된다.

본 시스템은 Positioner 혹은 Conveyor 시스템와 같은 외부 장치을 포함하여 동시에 12축까지 제어할 수 있기에, 두대 로보트 사이의 충돌 회피와 외부 장치와의 동기를 고려하여야 한다. 이를 위해, 조건운동 명령어를 ON condition DO action의 문법으로 제공하였다. 그러한 조건적 명령어(Conditional Motion)는 위에 기술한 모든 운동 명령어에 사용할 수 있다. 예를 들어, 두대의 로보트 사이의 가능한 충돌회피를 하기 위해, 거리 조건과 함께 COLL_AVOID 명령어를 다음과 같이 사용할 수 있다.

```
cobegin
    movl robot1 to loc1 with sp = 60
    movl robot2 to loc2  ON dist < 10.0  DO coll_avoid
coend
```

특히, 일반적인 3-차원 충돌회피[12],[13]는 실시간에 구현하기가 어렵기 때문에 본 시스템에서는 단지 두 대의 SCARA Type 로보트에만 적용되는 충돌 회피 기능을 구현하였다. 알고리즘은 다음과 같다.

만약 두대 로보트 사이의 거리가 주어진 조건보다 작으면 (충돌 가능성이 있으면) i) 두 대 로보트가 다른 방향으로 움직이는 경우에는, Master 로보트는 목표를 향해 움직이고 Slave 로보트는 충돌이 회피 될 때까지 안전한 지점까지 회피하고, ii) 두 대의 로보트가 같은 방향으로 움직이는 경우에는 우선 Master 로보트가 움직이고, 뒤 따라오는 로보트는 잠시 동안 기다린다(Wait).

이러한 동작 명령어는 16점 디지탈 입력 신호을 확인하는 DIN 조건, 동작 시간을 확인하는 TIME 조건, 입력 센서신호에 근거로 시스템이 궤적을 다시 계획하는 MOVS 동작, 시스템을 멈추는 STOP 동작, 16점 디지탈 신호를 출력하는 DOUT과 같이 이용할 수 있다.

전술한 동시 동작 명령어와 조건문장을 구현하기 위해서, Real Time O.S. XINU을 채택했다. 특히, 모든 동작 명령어는 프로세스로써 정의한다. COBEGIN 과 COEND문장인 경우, 모든 프로세스는 COBEGIN과 COEND 사이에서 Create 되고 Resume 된다. Conditional Motion인 ON condition DO action 문장의 경우에서는, Motion을 수행하다가 Condition을 만족하면 Motion을 정지 시키고 Action을 수행한다.

전술한 동작 명령어 외에 Teach Point 근처를 연속적으로 경유하는 Position Level과 좌표계를 변환하여 작업하게하는 Trans_Coord와 기준 좌표계를 새로 설정하는 Ref_Coord등을 이용한 Job Shift 기능, Pallet를 위한 Palletizing 기능등이 있다. 이외에도 Tool 제어, 논리 연산, Loop 제어, 산술 연산 명령어 등이 구현되어 있다. (Table 1 참조)

6. 실험 및 결과

제안된 두 대의 로보트 컨트롤을 위한 제어 시스템 및 로보트 배치(Layout)는 그림 2와 같고, Vision System의 Camera와 조명의 설치는 그림 3과 같다.

로보트 제어 시스템의 효용성을 보이기 위하여 작업대 위의 공동작업영역(Common Workspace)내에 있는 주사위를 각 주사위 눈에 맞는 Stock 위치로 옮겨 놓은 다음, 다시 그 주사위를 Conveyor상의 Plate위에 올려 놓고, Limit Sensor가 Sensing될때까지 기다린 후, 이동된 Conveyor상의 주사위를 들어서 작업대 위로 옮겨 놓는 작업을 반복 수행하였다. 비젼 시스템에서 각 주사위의 위치, 각도, Hole수를 계산하고, 또한 각 주사위 사이의 상호거리 및 각도를 계산해서 집을 수 있는 주사위를 선정한 다음 재 선정된 주사위의 위치 및 각도, Hole수를 로보트 좌표계로 환산해서 공유 메모리(CM)에 쓰면, 그 다음 Supervisory 프로세서에서는 물체가 놓여져야 할 위치 및 이동거리를 결정하여 로보트를 움직이는 명령을 Servo 시스템에 내린다. 이 경우 충돌회피 및 Concurrent한 Motion을 하면서 주사위를 맞추는 작업 및 Conveyor Tracking을 할 수 있음을 보였다. Demo Scenario를 위한 전체 시스템의 구성은 그림 8과 같고, 이를 위한 로보트 Programming은 그림 9와 같다.

7. 결 론

본 논문에서는 다중 센서 융합(Multi Sensor Fusion)능력을 갖는 여러 프로세서를 기본으로한 두대 로보트의 제어를 Multi-tasking O.S.인 XINU를 사용하여 구현하였다. 본 제어 시스템은 두대 로보트의 제어에 필요한 장애물 회피, 조건동작(Conditional Motion)혹은 동시동작(Concurrent Motion)과 Device와의 동기 Motion(Conveyor Tracking)을 수행할 수 있게 구현하였고, 주사위 맞추는 작업을 통해 우수성을 입증하였다.

앞으로의 연구과제는 1) 자유도가 6 관절형인 수직다관절 Manipulator를 위한 충돌회피 알고리즘의 개발 2) Two-arm Robot의 상대 위치를 위한 Auto-Calibration System의 개발 3) Force Feedback에 의한 협조제어의 개발등이 있다.

본 연구는 상공부 공업기반 기술개발사업의 연구비 지원으로 수행 되었음.

< 참 고 문 헌 >

[1] J. W. Roach and M. N. Boaz, "Coordinating the Motions of Robot arms in a Common Workspace ," IEEE Journel of Robotics and Automation, Volume RA-3 No.5, October, pp 437-444, 1987

[2] R. Guptill and S. Paul ,"Multiple Robotic Devices : Position Specification and Coordination," IEEE International Conf. on Robotics and Automation, pp 1655-1659, 1987

[3] A.A. Mangaser , Y. Wang, and E.S. Butner,"Concurrent Programming Support for a MultiManipulator Experiment on RIPS," IEEE International Conf. on Robotics and Automation, pp 853-859, 1989

[4] M.A. Turk, "A Vision System for Autonomous Land Vehicle Navigation," IEEE Trans. Patt. Anal. Mach. Intell., Vol.10, No.3, pp 342-361, May 1988.

[5] S. Venkatesan and C. Archibald, "Real time tracking in five degrees of freedom using two wrist-mounted laser range finders," Proc. of IEEE Int. Conf. on Robotics and Automation, pp 2004-2010, May 1990.

[6] A.J.koivo, and N. Houshang, "Real-Time Vision Feedback for Serving Robotics Manipulator with Self-Tunning Controller," IEEE Trans. on Sys. Man. Cybern., Vol.21, No.1, pp 134-141, January/February 1991.

[7] M. A. Abid ,R. O. Eason,and R.C.Gonzalez, "Autonomous Robotics Inspection and Manipulation Using Multisensor Feedback," IEEE Computer, Vol.24, No.4, pp 17-31, April 1991.

[8] D.M.LYONS, and M.A.ARBIB, "A Formal Model of Computation for Sensor-Based Robotics," IEEE Trans. on Robotics and Automation, Vol.2, No.3, pp 280-293, June 1989.

[9] Z.Bein, S.R.Oh, I.H. Suh, J.O. Kim, and Y.S. Oh, " Automatic Assembly for Microelectronic components," IEEE Control Systems Magazine, Vol. 9, No. 4, June 1989

[10] Motorola series in solidstate Electronics, VMEbus Specification Manual, 2nd printing Revision C.1 October 1985

[11] S.Mujtaba and R.Goldman: AL users' Manual, Stanford Artificial Intelligent Laboratory Memo, AIM-323, 1979

[12] R. A. Basta, R. Mehrotra, and M. R. Varanasi, "Detecting and Avoiding collisions between two robot arms in a common workspace," Robot Control Theory and Application, pp 185-192, 1988

[13] R.A.Basta, R.Mehrotra, and M.R.Varanasi "Collision Detection for Planning Collision Free Motion of two Robot Arms," Proceeding of the IEEE International Conf. on Robotics and Automation, 1988

Fig. 1. 다중 로보트를 위한 제어 시스템

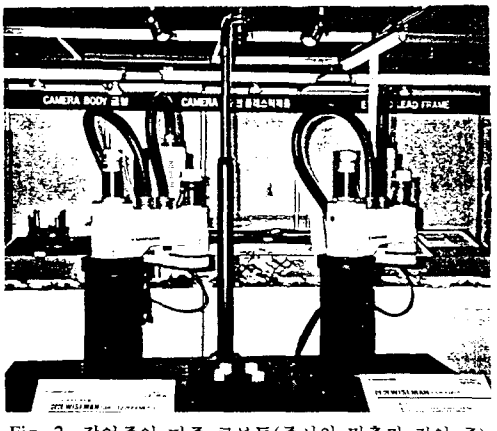

Fig. 2. 작업중인 다중 로보트(주사위 맞추기 작업 중)

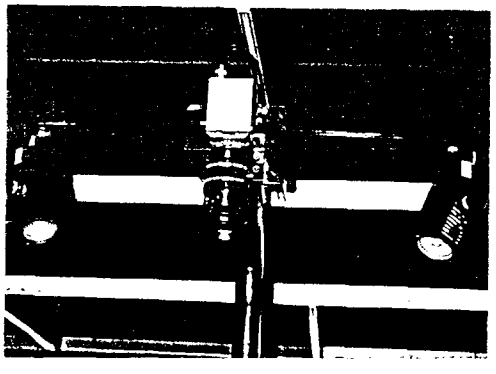

Fig. 3. 비젼 센서부

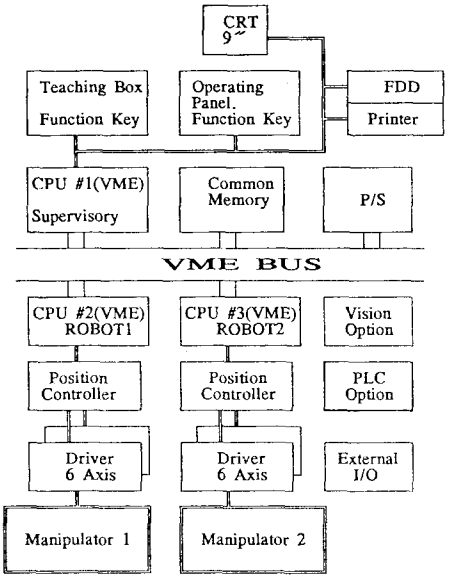

Fig. 4. 제어 시스템의 H/W 구조

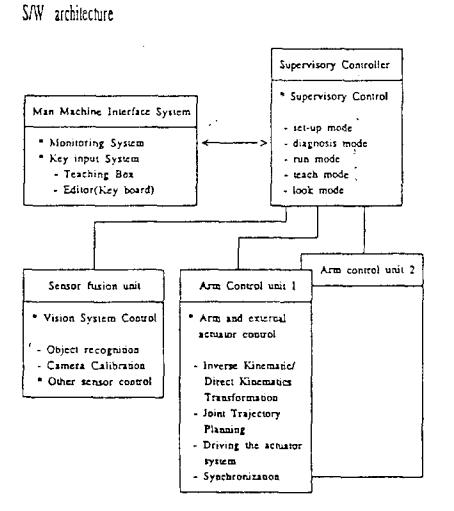

Fig. 5. 제어 시스템의 S/W 구조

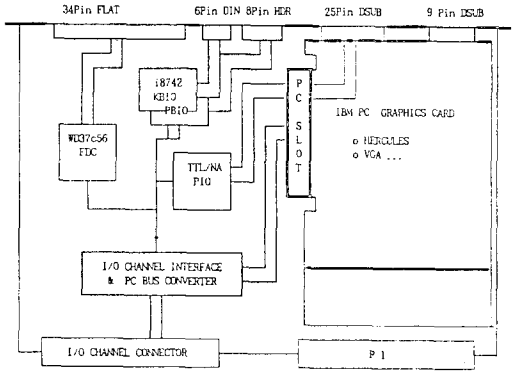

Fig. 6. VFIOC Board의 전체 구성

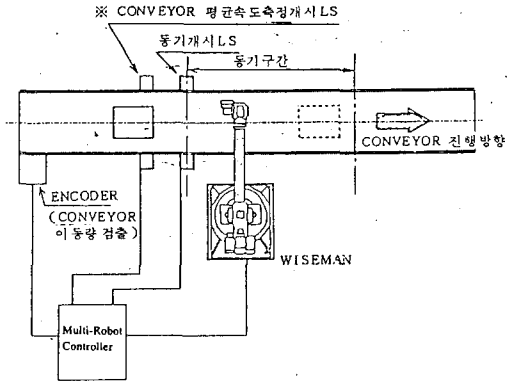

Fig. 7. Conveyor 동기 운전의 시스템 구성

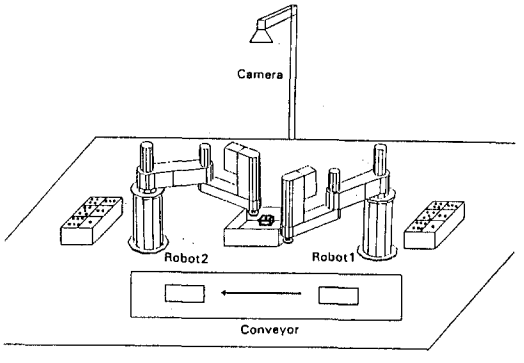

Fig. 8. Demo Scenario를 위한 전체 시스템의 구성

```
A part of program for dice matching work
by using conveyor tracking and vision system

define   master  robot1
define   slave   robot2

speed = 50

for i0 = 0 to 4

    sv_recog

    cobegin
       movj master to vi_0 ON dist < 10  Do coll_avoid
       movj slave to vi_1
    coend

    cobegin
       grip_down robot1 with disp = 55  sp = 10
       grip_down robot2 with disp = 55  sp = 10
    coend

    grasp robot1
    grasp robot2

    cobegin
       movj robot1 to base2_1
       movj robor2 to base2_2
    coend

    affix robot1 for vi_0
    unfix robot1
            .
            .
next i0     .
            .
cvspchk cv#1
cv_synch robot cv#1 t=10
   movs robot1 to vi_0 with sp=10
            .
            .
cvend       .
            .
```

Fig. 9. 로보트 언어의 실례(주사위 맞추기 작업)

Table 1. 로보트 제어 언어 일람표

Instruction Group	Instruction	Structure
Basic Motion Instructions	MOVJ MOVL MOVC MOVA MOVI MOVS	movj / movl <robot> to<loc1> with sp=<speed> pl = <positioning level>. movc / mova <robot> via <loc1> <loc2> with sp = <speed>. movi <robot > by<offset> with sp = <speed>. movs <robot> st=<time>
Master & Slave Motion Instructions	MS_MOVL MS_MOVI MS_MOVC MS_MOVA	ms_movl master to <loc1> with sp = <speed> ms_movc master via <loc1> <loc2> with sp=<speed>
Collision avoidance motion Instruction	COLL_AVOID	do coll_avoid
external device control Instruction	MOV_EXT	mov_ext <device#> to <loc> with <speed>
Conditional Instructions	On Condition Do Action	ON <Statement> DO <Statement>
Conveyor Tracking Instruction	CV_SPCHK CV_SYNCH MOVS CVEND	cv_spchk cv#<number> cv_synch robot#<number> cv#<number> t=time movs robot#<number> to <loc> with sp=<speed> cvend
Concurrent Instructions	COBEGIN / COEND	cobegin simple statement 1 simple statement 2 coend
Coordinate Translation Instructions	TRANS COORD, REF_COORD, MAKE- POSITION, AFFIX, UNFIX	trans_coord <loc1> for <coord> ref_coord <coord> makeposition(TRANS, TOOL1...LOC) affix <object> with <robot> unfix <robot>
Gripper Control Instructions	GRASP, RELEASE, GRIP_UP, GRIP_DOWN GRIP ROTATE TOOL	grasp / release <robot> grasp <robot> with width=# grasp <robot> until < sensor related Instruction>=<#> grip_rotate <robot> tool <tool#>
Boolean Logical Operators	AND, OR, NOT,XOR	<variable> AND <variable>
Relational Operators	<,<=,>, >=,==,!=	<variable> <,<= <variable>
Algebraic Operators	+,-,*,/	<variable> +, -, *, / <variable> <coordinate> +, - <coordinate>
Control Instructions	FOR,WHILE, UNTIL, IF, IF-ELSE, CALL RET,STOP, NOP PAUSE	 CALL function_name WITH arg1,arg2,arg3; PAUSE IF IN#10 = 1
Vision Related Instructions	SV_INIT SV_GRAB, SV_CLS, SV_THSET, USECAMERA, SETSIZE, SV_RECOG, PRINT, SV_SNAP	 SV_THSET <number> USECAMERA <number> SETSIZE <number> SV_RECOG <object name> PRINT <string> ON <x position>,<y position>
Other sensor Related Instructions	TIME , DIST FORCE, LIMIT	LIMIT #<number>