창 립
40주년학술대회
논문87-L-20-4

공정 자동화를 위한 고수준 로보트 프로그래밍 시스템 구성에 관한 연구

○
서 기 성 우 광 방
연세대학교 전기공학과

A Study on High-Level Robot Programming System for Automation of Manufacturing Products

Ki-Sung Seo Kwang-Bang Woo
Dept. of Electrical Eng., Yonsei University

ABSTRACT

This paper describes implementation of a robot programming system for automation of manufacturing products, which is embedded in the C programming language. World representation and motion primitives necessary to describe a manipulator task are provided by a set of procedure calls and user defined data structures.

Off-line programming is implemented with graphic simulation as a debugging tool.

PUMA 560 manipulators are used as a model for one task which inserts a nonstandard power IC into PCB. Communication facilities are provided for collision avoidance of two manipulators.

1. 서론

로보트 매니퓰레이터를 프로그래밍하는 것은 대상 물체를 이곳에서 저곳으로 이동시키거나, 도달해야하는 매니퓰레이터의 공간상의 다른 위치를 묘사하는 것으로 구성된다. 앞으로의 로보트는 task 묘사의 teach 부분을 최소화하고 프로그램 부분을 강조하는 추세이다.

매니퓰레이터 언어를 설계하는데는 두 가지의 방법이 있다. 하나는 로보트 control primitives에 촛점을 둔 전용 언어를 설계하는 것이고, 다른 하나는 일반적인 컴퓨터 프로그래밍 언어를 embedded 형태로 사용하는 것이다.

전용 언어는 그것의 구현이 실제적인 로보트 작업과는 무관한 scanner와 parser와 같은 특정 tool의 설계를 포함한다. 또 수정이 어렵고 모듈성, 매니퓰레이터 독립성이 줄어든다.

복잡한 매니퓰레이터 응용은 더 많은 계산과 데이타 처리 그리고 user interaction이 실제의 motion control보다 더 요구된다. 따라서 고 수준 컴퓨터 프로그래밍 언어와 같은 특성을 가진 융통성 있는 매니퓰레이터 언어를 필요로 한다. C 언어는 구조화된 언어이며 시스템 프로그래밍에 적합하고 pointer의 기능과 주소 연산이 간편하다. C 언어를 사용한 매니퓰레이터 프로그램은 고수준 컴퓨터 언어에서 제공되는 모든 특성을 가지며 I/O 기능과 낮은 level의 제어도 가능하다. 매니퓰레이터의 task를 표현하는데 필요한 world model과 motion 명령은 사용자가 정의한 데이타 구조와 서브루틴 call에의해서 이루어 진다.

본 연구에서는 기존의 C 프로그래밍 언어에 기초한 로보트 매니퓰레이터 언어를 구현하여 효과적으로 PUMA 560 로보트 매니퓰레이터를 제어할수 있는 off-line 로보트 프로그래밍 시스템을 구성하고 Graphic Simulation 를 통해 motion을 분석한다.

여기서 구현한 로보트 프로그래밍 시스템을 MCRC(Modified C language for Robrt Control) 라 한다. 한 작업을 한대의 로보트보다 두대의 로보트가 상호 협조하여 수행하는것이 전체공정의 시간을 단축 시킬수 있고, 특정 작업에 효과적일수 있으므로, 이를 위해 MCRC를 포함한 두대의 컴퓨터를 서로 연결하여 전체 시스템을 구성하였다.

2. Off-line 프로그래밍 시스템 구성

2.1. Off-line 프로그래밍의 필요성

소규모의 생산과 재프로그래밍이 빈번한 자동생산라인은 전체 시스템을 프로그래밍하는데 필요이상으로 시간이 낭비된다. 이를 해결하기 위해 로보트나 오퍼레이터가 분리된 상태에서 시뮬레이터를 통해 동작을 분석할수 있는 off-line 프로그래밍 시스템이 많이 사용되고 있다.

off-line 프로그래밍 방식의 경우, 작업의 정확도는 다소 떨어지지만 로보트를 직접 다루지 않기 때문에 작업을 바꾸기 위해 전체 공정을 멈출 필요가 없고 teach 과정

에서 발생할수 있는 여러가지 위험한 상황으로 부터 안전할수 있다.

또 메인 컨트롤러의 계산 능력이 실시간 처리에 못 미칠 경우에도 off-line 프로그래밍이 요구된다. 따라서 IBM-PC와 같은 낮은 비용의 퍼스널 컴퓨터로서도 효율적인 매니플레이터의 제어가 가능하다.

앞으로는 로보트 프로그래밍 시스템이 CAD/CAM 과 결합되는 것이 필수적이므로 off-line 프로그래밍의 중요성은 매우 높다고 볼수 있다.

본 연구에서는 IBM-PC AT와 IBM-PC XT를 RS-232-C로 연결하여 두대의 PUMA 560 로보트 매니플레이터를 각각 제어할수 있도록 off-line 프로그래밍을 구성하였고 처리 능력이 보다 나은 IBM-PC AT에서 그래픽 시뮬레이션을 행하도록 하였다.

2.2. PUMA 560 로보트 매니플레이터

PUMA 560 로보트 매니플레이터는 Unimation사 에서 제작되었으며 6개의 관절을 가진다. (그림 2.1)

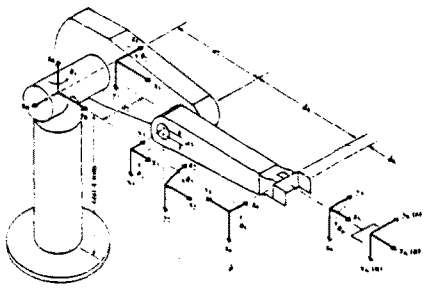

그림 2.1 PUMA 560 매니플레이터의 제원 및 관절 좌표계

PUMA 560 로보트 매니플레이터의 컨트롤러는 LSI-11/02 마이크로 컴퓨터 부분과 조인트 인코더, D/A 컨버터, 전류 증폭기를 가진 6개의 6503 마이크로 프로세서로 이루어진 계층적 제어 구조이다.

본 논문에는 메인 컨트롤러인 LSI-11/02를 IBM-PC로 대체하고 인터페이스 부분과 서보 제어 부분은 연결되어 있다고 가정 하였다. 그리고 VAL이란 전용 언어로 제어되는 PUMA 560 매니플레이터를 C 언어를 이용하여 제어할수 있도록 MCRC 를 구성하였다.

여기서 구성된 MCRC 는 사용자가 궤적 계획이나 운동제어를 다루는 프로그램 부분을 다시 쓸수 있어 연구 개발 목적등에 매우 유용하다.

3. 로보트 운동 제어

3.1 world model

제공된 변환 T6는 매니플레이터의 base에 위치한 기준 좌표축에 대한 end-effector의 위치를 나타낸다. 예를 들어 목표 위치에 대한 매니플레이터의 주어진 운동은 다음과 같이 표시할수 있다.

TBL T6 TOOL = CONV OBJ PG

여기서

TBL : 참고 좌표축에 대한 매니플레이터의 위치이다.

T6 : 매니플레이터의 base 대해 매니플레이터의 end-effecctor 위치를 나타낸다.

TOOL : end-effector에 부착된 tool의 위치를 표현한다.

CONV : 기준 좌표축에 대해 conveyor belt를 나타낸다

OBJ : conveyor belt상에 놓여있는 물체의 위치이다.

PG : OBJ에 대한 end-effector의 위치이다.

위치 방정식이 기준 좌표축에 대해 매니플레이터의 원하는 위치를 얻기위해 T6에 대해 풀어진다.

$T6 = TBL^{-1}\ CONV\ OBJ\ PG\ TOOL^{-1}$

3.2 운동 제어

대부분의 로보트 제어 시스템에서 path control은 두가지 방법의 운동을 가진다: point to point 또는 path control.

point to point 운동 제어는 조인트 보간 방법을 사용하여 얻어진다. 조인트 운동은 그들의 현재값으로부터 특정한 목표값까지 모든 조인트 위치들을 보간함으로써 얻어지고 모든 조인트들은 그들의 운동을 동시에 완료한다. 이런 형태의 방법은 빠른 arm 운동을 제공하지만 arm이 일반적으로 예측할수 없는 복잡한 공간 곡선을 따라서 움직인다.

path control 운동하에서는 arm은 명시된 곡선을 따라서 이동한다. arm path control 운동은 보통 카테이션 공간에서 path segmentation을 필요로 한다. 그리고 knots는 조인트 위치로 변환된다. 조인트 위치의 보간은 조인트 운동을 생성한다.

본 연구에서는 path control 운동 방법을 사용한다.

일련의 카테이션 위치에 의해 표현되는 특정한 곡선이 있다고 가정하면, 처음의 중간 카테이션 위치(knots)가 주어진 곡선을 따라 arm이 움직이도록 하는 카테이션 보간 계획에 의해 만들어진다. 그런다음 이들 카테이션 points에 대응하는 조인트 위치가 계산된다. 이들 모든 계산은 arm 운동에 앞서 실행되고 조인트 위치는 계적 struct에 저장된다. 서보 시스템에 출력되는 마지막 조인트 위치는 arm이 움직일때 queue 형태로 된 계적 struct로부터 조인트 보간 계획에 위해 생성된다. 예측할수 있는 path control 에서는 서보가 일정한 시간 간격 동안에 조인트 위치를 얻고, 두 조인트 위치간의 거리는 움직이는 속도를 결정한다.

4. MCRC의 구성

4.1 특징 및 일반적인 형태

로보트 언어는 사용자와 로보트간의 매개체 역할을 하므로 인간이 사용하기에 쉽고 효율적인 작업을 할 수 있도록 설계되어야 한다. 그러기 위해서는 첫째, 동종변환을 자유롭게 사용할 수 있는 구조적 프로그래밍 이어야 한다. 둘째, 특정 로보트 시스템에 국한되지 않고 매니플레이터의 수학적 구조 부분만을 바꾸면 다른 로보트 시스템에도 적용될 수 있도록 일반성을 가져야 한다. 세째, liblary들을 추가하여 새로운 기능을 가질 수 있도록 확장성과 모듈성이 있어야 한다. 네째, 로보트 시스템에 지능을 부가할 수 있도록 AI언어(Prolog 등)의 인터페이스 문제도 고려되어야 한다.

C 언어에 embeded 된 MCRC는 위의 조건들을 거의 만족시키며 부가적으로 매크로 기능은 프로그램의 reliability를 보다 간단하게 할 수 있다. 사용자 프로그램은 일반 C 언어 프로그램과 동일하고 MCRC 에 존재하는 여러 프로시듀어와 라이브러리들을 call하여 사용된다.

MCRC는 Turbo C compiler를 사용하였고 교정모드, 수행모드, exit모드 등을 포함한다.

4.2 MCRC Library

- gentrans_eul(), gentrans_rot(), gentrans_rpy(), gentrans_trsl() : 각 변환을 생성.
- makepos(lefthand, EQ, righthand) : 일련의 변환으로 된 위치 방정식을 세움.
- move(P), movec(P), movej(P) : trajectory planner를 call하여 보간계획에 의해 정해진 모드에 따라 주어진 위치로 이동.
- stop(T) : 주어진 시간동안 매니플레이터를 정지시킴.
- waitfor(flag), waitas(time) : 플래그가 참, 또는 주어진 시간만큼 사용자 프로그램을 wait 시킴.
- setmode(), setconfig(), setvel() : 모드, 로보트 형태, 속도 등을 세트 시킴.
- mcrc_open() : 작업 시작시에 매니플레이터의 reset 위치를 포함, MCRC 에 필요한 파라미터들을 초기화 시킴.

5. 두대의 PUMA 560 매니플레이터의 응용작업 및 프로그램

다음 예에서 두대의 PUMA 560 매니플레이터를 이용하여 PCB에 부피가 크고 표준화되지 않은 파워 IC를 삽입하는 작업을 보인다. 그림 5.1에 작업환경을 나타내주는 world model이 나와있다. 두대의 매니플레이터는 각각의 메인 컨트롤러인 IBM-PC이 의해 제어된다.

처음에 두대의 매니플레이터는 IC magazine 위의 집으려는 IC 위에 reset 상태로 있다고 가정하고 매니플레이터 1이 매니플레이터 2에 대해 start request 신호를 보내면서 이동을 시작한다. 역시 신호를 받은 매니플레이터 2도 이동을 시작하여 두대의 매니플레이터는 충돌위험성이 있는 컨베이어상의 PCB 경계 위치까지 도달한다.

먼저 도착한 매니플레이터는 상대 매니플레이터에게 정지신호를 보내고 PCB내로 들어가 작업을 마친다음 PCB를 빠져나오면서 상대 매니플레이터에게 restart 신호를 보낸다. 그러면 이 신호를 받은 매니플레이터는 PCB로 들어가 작업을 수행한다.

두대의 매니플레이터간의 통신에 사용되는 데이타는 서보컨트롤러에 보내질 queue의 struct 중의 com field에 넣어지고 queue에 저장되어 있는 계적 조인트 값이 서보컨

트롤러로 보내질 때 com field 를 체크하여 data가 있으면 이를 RS232C를 통해 상대 매니플레이터에 전달하게 한다.

위의 동작은 MOVE명령이 call될 때 trajectory planner 가 내부적으로 수행한다.

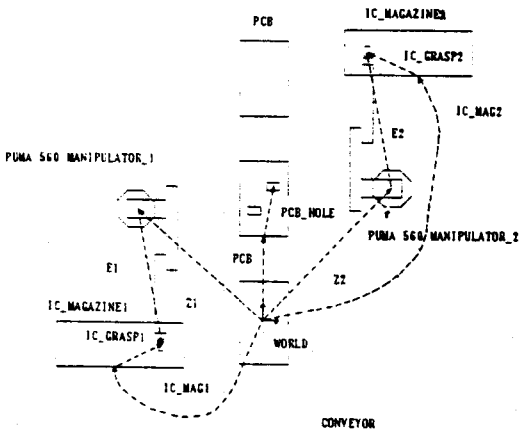

그림 5.1 PCB 삽입을 위한 작업 모델 환경

다음에 사용자 프로그램의 예가 나와있다.

```
/****      PUMA_1 TASK PROGRAM      ****/

#include <mcrc.h>
main()
{
struct transform *Z1, *E1, *IC_MAG1, *IC1,
                 *IC_GRASP1, *PCB, *PCB_HOLE ;
struct position  *P1, *P2 ;

Z1 = gentrans_rot(-700.0, 700.0, 0.0, 'z',
                  -PI/2.0) ;
E1 = gentrans_rot(155.0, -885.0, 660.0, 'z'
                  -PI/2.0, 'y', PI/2.0) ;
IC_MAG1 = gentrans_trsl(-800.0, 150.0, 10) ;
IC_GRASP1 = gentrans_trsl(255.0, 0.0, 20.0) ;
PCB = gentrans_trsl(0.0, 575.0, 10.0) ;
PCB_HOLE = gentrans_trsl(60.0, 80.0, 0.0) ;

mcrc_open() ;

P1 = makepos(Z1, T6, E1, EQ, IC_MAG1, IC1,
             IC_GRASP1) ;
P2 = makepos(Z1, T6, E1, EQ, PCB, PCB_HOLE) ;

startrequest() ;
GRASP ;
setmode('c') ;
move(P2) ;
RELEASE ;
move(P1) ;

    /** end of PUMA_1 task program **/
```

6. 그래픽 시뮬레이션

앞 절에서 설명한 작업을 2차원 그래픽 시뮬레이션을 행하였다. PUMA 560 매니플레이터의 이동운동을 애니메이션을 통해 분석하였고 운동제어는 카테이션 직선 보간 방법을 이용하였다. hand는 시작점에서 목표점까지 직선을 따라 움직이도록 제어하였다.

2차원 평면에서 hand의 이동을 볼수 있도록 매니플레이터의 Euler angles를 고려하지 않고 위치를 나타내는 조인트 1 부터 조인트 3까지만을 보간하였다. hand는 reset 위치로 고정시켰다.

그림 6.1에 PCB 삽입을 수행하는 두대의 매니플레이터의 이동 운동이 그래픽으로 디스플레이 되어있다.

off-line 프로그램에서 계산한 조인트 값을 가지고 IBM PC-AT 에서 Turbo Graphix Toolbox로 그래픽 처리하였다. 나타난 결과는 3차원 물체를 X-Y 좌표상으로 투영한 것이다.

7. 결론

본 논문에서는 두대의 PUMA 560 매니플레이터를 모델로 하여 off-line 프로그래밍 시스템을 구성하고 2차원 그래픽 시뮬레이션으로 운동을 분석하였다.

C 언어를 기초로 한 MCRC는 프로시듀어 call로서 효과적인 매니플레이터 언어가 구성되었고 궤적계획과 운동제어를 다루는 프로그램 부분이 사용자에게 개방되어 있어 연구 개발에 유용하다. 이것은 사용자가 매니플레이터에 대한 특별한 전문지식이 없어도 프로그래밍이 가능하고 구조적 언어의 특징인 모듈성, 확장성 등이 뛰어나며, C 언어의 특징인 낮은 level의 제어가 가능하다는 장점을 가지고 있다. Prolog, Pascal 같은 다른 언어와의 인터페이스도 가능하므로 매우 유용하다.

응용 프로그램에서는 C 언어에 부가된 I/O 기능 및 낮은 level 제어기능을 이용하여 두대의 매니플레이터가 서로 데이터를 주고 받을 수 있는 통신기능을 첨가하여 서로 충돌을 피하면서 한 작업을 수행할 수 있게 하였다.

두대의 IBM-PC를 사용한 off-line 프로그래밍 시스템은 낮은 비용으로서도 그래픽 시뮬레이션을 통해 실제의 매니플레이터를 동작시켜보지 않고도 매니플레이터에 대한 효과적인 연구를 수행할 수 있다.

8. 참고 문헌

1. T. Lozano Perez, "Robot Programming", Proc. IEEE Vol. 71, July 1983 pp 821-841

2. K. Takase , R. P. Paul, "Structured Approach to Robot Programming and Teaching", IEEE Tr. on Systems, Man, and Cybernetics, Vol. SMC-11 No. 4, April 1981, pp 274-289

3. V. Hayward , R. P. Paul "Robot Manipulator Control Using the "C" Language ", IEEE Workshop on Language for Automation, 1983, pp 3-10

4. R. P. Paul, Robot Manipulators : Mathematics, Programming and Control, The MIT Press, 1981

5. K. S. Fu, et. al., Robotics : Control, Sensing, Vision, and Intelligence, McGraw-Hill 1987

6. R. H. Taylor, "Planning and Execution of Straight Line Manipulator Trajectories", IBM Journal of Research and Development, Vol. 23, July 1979, pp 253- 264

7. B. E. Shimano, et. al., "VAL-II : A New Robot Control System for Automatic Manufacturing", Proc. of the Int'l Conf. on Robotics, March 1984, pp 278-292

8. T. C. Hsia, et. al., "A New Robot Control Language", Int'l Conf. on IECON, 1985 pp 833-838

9. T. Sugiyama, "Robot Controller for Off-Line Programming System", Proc. of 15th ISIR, 1985, pp 715-722

10. S. Kawabe et. al., "Interactive Graphic Programming for Industrial Robots", Proc. of 15th ISIR, 1985, pp 699-796

11. H. Matoba, et. al., "Robot Programming System Using Interactive Graphics", Proc. of 15th ISIR, 1985, pp 759-766

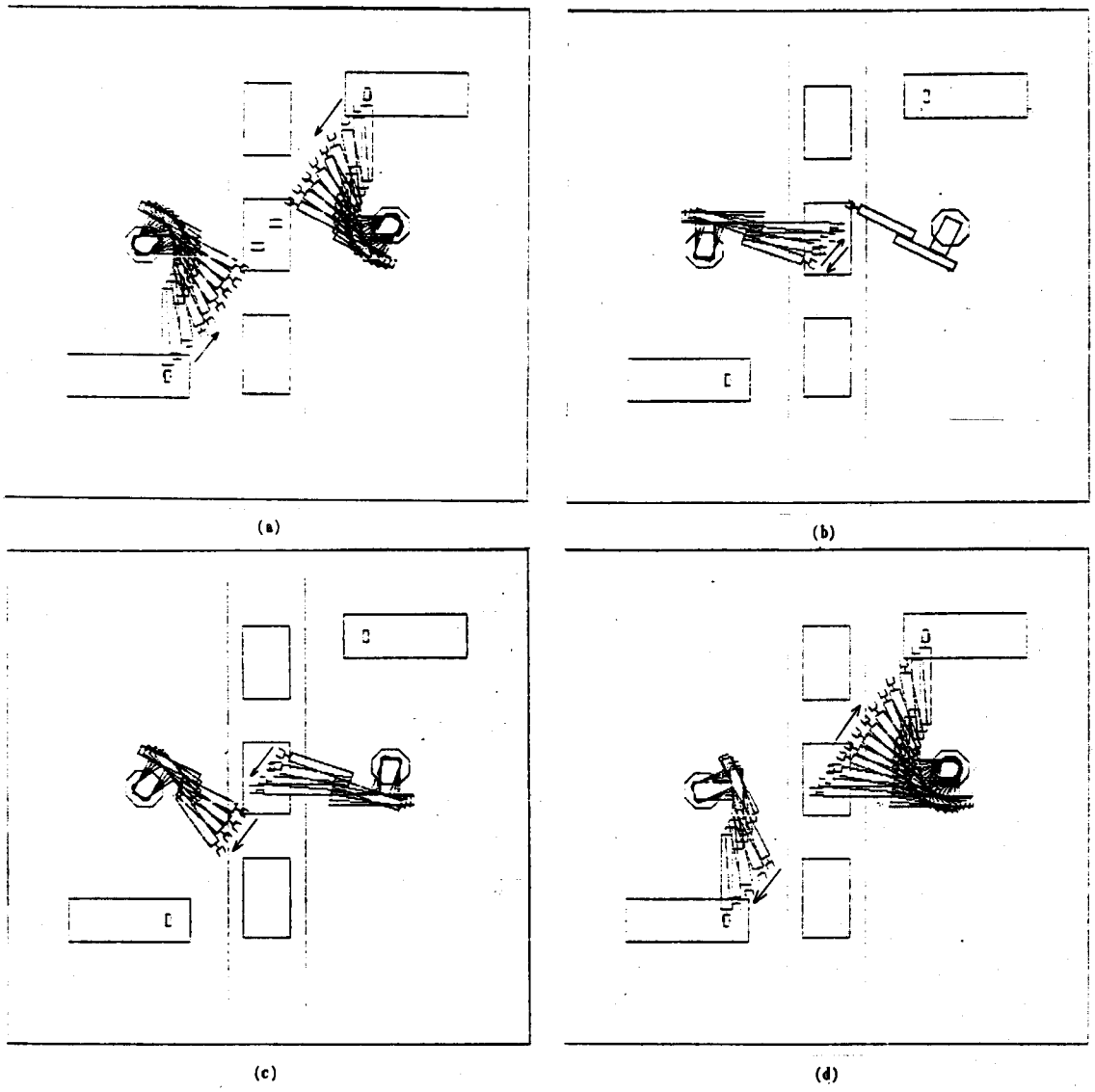

그림 6.1 두대의 PUMA 560 메니퓰레이터의

작업 동작 분석