'88 韓國自動制御學術會議論文集 1988. 10. 21~22

로보트 자동 프로그래밍을 위한 원형 시스템의 설계

°조 혜경, 고 명삼, 이 범희
서울대학교 공과대학 제어계측공학과 로보틱스및 지능 시스템 연구실

A Design of a Prototype System for Automatic Robot Programming

°Hye-Kyung Cho, Myoung-Sam Ko, Bum-Hee Lee
Robotics and Intelligent Systems Lab.
Dept. of Control and Instrumentation Engineering
Seoul National University

Abstract

This paper describes an experimental system for automatic robot programming, The SNU-ARPS (Seoul National University Automatic Robot Programming System). The SNU-ARPS generates executable robot programs for pick and place operation and some simple mechanical assembly tasks by menu-driven dialog. It is intended to enable the user to concentrate on the overall operation sequence instead of the knowledge regarding the details of robot languages. To convert task specifications into manipulator motions, the SNU-ARPS uses an internal representation of the world. This representation initially consists of geometric data base from CAD system and is updated at each operation step to reflect the state changes of the world.

1. 서론

날로 그 필요성을 더해가고 있는 자동화의 물결은 좀 더 다양하고 복잡한 로보트의 기능을 요구하고 있다. 로보트 언어는 로보트 구조의 변화 없이 이러한 기능을 부여할 수 있는 도구로서 계속적으로 발전해 왔다. 그러나 교시(teach)를 기본으로 하는 온 라인(on-line) 프로그래밍 시스템은 단순, 반복 작업에는 유용하나, 작업 환경이 위험하거나 작업이 복잡해질 경우 로보트와 작업자 모두에게 비효율을 가져오며, 오프 라인(off-line) 프로그래밍 시스템은 다양한 기능을 소유하며, 그 기능이 완벽해질수록 간단한 작업 조차도 프로그램하기가 매우 복잡해진다는 모순에 빠지게 되었다[1,2].

이러한 문제점을 해결하기 위한 방법으로, 크게, 교시 개념을 확장하는 방법 (extended guiding) 과 작업중심언어(task-level language)의 개발을 들 수 있는데, 본 연구는 후자에 속하는 접근 방법으로 로보트 언어를 모르는 사용자들도 시스템과의 대화를 통해, 로보트 기종및 제어기에 적합한, 실행 가능한 로보트 프로그램을 작성할 수 있도록 하는 지능 로보트 언어 시스템 (이하 SNU-ARPS : Seoul National Univ. Automatic Robot Programming System이라 칭함)을 구성한다. 이 시스템은 기존의 로보트 제어기및 언어 시스템을 그대로 이용하되 그 사용을 간편화할 수 있는 도구를 마련하여 컴퓨터 프로그래머가 아닌 공장의 작업자들도 쉽게 로보트를 이용할 수 있게 함으로써, 지루하고 비효율적인 교시에 의존하고 있는 로보트 사용시의 문제점들을 해결하고자 하는 시도로 제작 되었다.

다양한 종류의 로보트와 로보트 언어를 적용대상으로 하는 SNU-ARPS 는 PUMA 구동을 위한 VAL[3] 언어, SCARA 형 로보트 구동을 위한 SNUL-I[4] 을 생성하여 실험하였으며, 센서 정보 처리를 위해 본 연구실에서 개발중인 SNUL-II에도 적용해 실험중이다. SNU-ARPS 는 IBM AT에서 인공 지능 시스템 개발용 언어로서 융통성있는 문장 처리가 편리한 GCLISP으로 실현된다.

본 논문은 2장에서 SNU-ARPS 구성의 배경에 대해 살펴보고, 3장에 그 구조를 설명하며, 4장에 사용 예를 보인다. 5장에는 SNU-ARPS의 특징과 앞으로의 연구 방향이 제시된다.

2. 문제의 설정

산업 사회의 발전으로 다품종 소량 생산이 요구됨에 따라 로보트가 다양한 생산 작업에 이용되게 되자, 생산 라인이 바뀔 때 마다 장시간의 재교육을 요구하

는 비효율적인 교시 방식은 한계에 부딪히게 되었다. 복잡한 작업을 수행하는 로보트 프로그램 과정을 용이하게 하고자하는 시도는 로보트 언어가 발전되면서 중요한 문제분야로 대두되었는데, 그 접근방법으로는 온라인 교시에 센서 정보를 이용할 수 있도록 교시개념을 확장하는 방법 [1] 과 KARMA, PROGRESS[5] 등과 같이 CAD 및 그래픽 도구를 이용하는 방법, LM, LM-GEO[6], RAPT[7]에서와 같이 위치관계를 기술하는 언어 (geometric language)를 이용하는 방법들을 들 수 있다. 많은 연구가 진행되고 있는 또 하나의 방법으로써 AUTOPASS[8], LAMA[9], HANDEY[10]등과 같은 작업중심 언어 (task-level language) 의 개발을 들 수 있는데, 70년대 중반부터 시작된 이러한 과감한 시도는 대부분 해결한 것 보다 더 많은 문제점을 제기하게 되어 어느것도 완전히 실현되지 못하였으므로, 현재 작업 중심 언어의 실현을 위해 선행되어야할 문제점들에 대한 연구가 활발히 진행중이다.

SNU-ARPS도 궁극적으로는 작업 중심 언어로 발전되어야 하므로, 로보트 프로그램시 작업중심언어 시스템이 자체적으로 해결하여야 할 문제 분야를 살펴 보면 아래와 같다.

- World Modelling

  로보트 자신및 작업 환경, 대상 물체들을 모델링할 수 있어야 한다.

- Task Specification

  로보트의 작업은 보통 작업 순서에 따른 작업 환경의 단계적 상태 변화로 기술되는데, 작업중심언어는 이를 분석하여 작업 환경 내의 각 물체에 대해 로보트가 행해야할 동작 및 이에 필요한 여러 변수들을 계산해 낼 수 있어야 한다.

- Collision-free Path Planning

  작업중심언어는 자신의 작업 환경 내의 물체들을 이동시킴에 있어 불필요한 충돌이 일어나지 않도록 이동 경로를 선택해야 한다.

- Automatic Grasping

  작업중심언어는 작업 지정시 대상 물체에 관한 위치나 방향 정보가 포함되지 않는다. 따라서 작업중심 언어는 환경 모델로 부터 안전한 잡는 위치를 선택할 수 있는 능력이 필요하다.

- Compliant Motion

  작업중심언어는 보통 조립이나 제조 공정등 다소 복잡한 작업을 대상으로 하므로 작업에 따라 적절한 센서를 이용한 컴플라이언트 운동을 실현할 수 있어야한다.

- Error Checking

  작업중심언어는 실제 작업 환경이 모델링된 것과 다르다는 것을 가정하고 있고, 또 실제가 그러하므로 매니퓰레이터로 실행될 프로그램은 센서 정보를 이용하여 이러한 오차를 보상하고, 제대로 실행되지 못한 작업에 대처하는 능력이 요구된다.

위에서 볼 수 있듯이 작업중심언어는 모든 문제를 자동적으로 해결하기 위해 많은 계산이 필요하고, 또 문제 성격상 아직도 해결되지 않은 부분이 남아있다. 이러한 이유로 현존하는 작업중심언어는 실용화되지 못하고 앞에서 언급한 문제들의 일부만을 실현한 연구 단계에 지나지 않는다. 또한 이들 작업중심언어들은 환경 모델링및 잡는 위치 선정, 충돌 회피를 위하여 강력한 비전 시스템을 사용하고, CAD 데이터 베이스를 이용하기도 하며, 직접 로보트를 움직이는 구동부의 언어는 불확실한 상황을 다루기 위해 다양한 센싱기능을 갖추고 있다.

이는 현재 공장에서 가장 많이 이용되고 있는 원시 운동단계(motion level)의 언어[2]들이 작업중심언어의 목적언어 (target language)로서 부적합함을 의미한다. 그러나 앞에서 언급한 제 문제에 대한 우리의 기술 수준과 원시 운동 단계 언어의 기능조차 충분히 발휘시키지 못하는 많은 공장의 로보트 이용 현황을 고려할 때, 작업중심언어의 개발은 다소 성급하게 느껴지며, 일반적인 stand-alone 형의 로보트 언어 시스템에 센싱 기능을 추가하기란 불가능한 일이므로 본 연구는 작업 중심 언어 개발의 첫 단계로 국내에서 널리 쓰이는 기존 언어들을 그 적용 대상으로 하였다. 따라서 SNU-ARPS는 다음과 같은 가정하에 구성되었다.

- 작업대상은 Pick & Place 및 간단한 조립 작업으

로 한다.

- 환경 모델링을 위한 자료의 위치 및 방향 정보 오차는 없다.
- 로보트의 동작에 의한 오차가 없어 실제환경과 모델링 된 환경이 항상 같다.

3. SNU-ARPS 의 구조

2장에 설명한 바와 같은 가정하에 구성된 시스템의 구조는 아래 그림 1과 같다. Discourse module 이 사용자와의 대화를 통해 작업을 입력받으면 domain expert 는 modeller 에 의해 만들어진 world model을 이용, 필요한 계산을 하여 program model을 coder 에 전달하며, coder는 이를 로보트 프로그램으로 바꾸어 준다.

SNU-ARPS 각 모듈에 대해 간단히 살펴본다.

3.1 환경 모델러 (World Modeller)

로보트, 물체, 작업 공간, i/o , 센서등 작업 환경 내의 모든 대상에 대한 정보를 받아들이고 관리하는 부분이다. 모델러가 관리하는 데이터베이스는 충돌회피경로 선택시 장애물의 모델을 제공하고, 각 관절의 각도 제한을 확인할 때, 잡는 위치를 선택할 때등 프로그램시 필요한 의사 결정을 지원할 뿐만 아니라 모든 대상물의 위치 데이타를 제공하므로 로보트 동작의 성패를 좌우하는 중요한 역할을 한다.

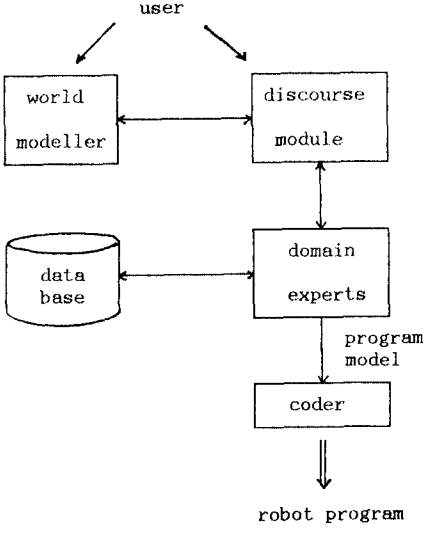

그림 1 SNU-ARPS 의 구조

정확한 위치 데이타를 얻고자 하는 시도는 오프 라인 프로그래밍의 주요 연구 분야로, 크게 CAD등 그래픽도구(graphic tool)를 이용하는 방법[5], 위치관계를 정의하는 언어 (geometric language) 를 이용하는 방법[6,7]과 비젼 (vision) 시스템을 이용하는 방법[10]을 들 수 있는데, 이러한 데이타의 입력 과정은 지루하고 힘든 작업이며 정확한 데이타를 얻기도 쉽지 않다. 일반적으로 조립 작업시의 위치 데이타는 운반시 이용되는 위치 데이타와 세부 조립 작업시 이용되는 위치 데이타로 구분할 수 있는데, 전자는 교시를 통하여 보다 정확히 얻어지나 후자는 오히려 부정확해지는 특징이 있다.

이러한 특징과 현재 국내의 로보트 이용 상황을 고려하여 SNU-ARPS의 모델러는 사용자로부터 입력받은 교시 데이타와 CAD 의 데이타를 분석하여 얻은 데이타, 두가지 모두를 이용한다. 모델은 AUTOCAD의 SNU-ARPS용 사용자 메뉴를 작성하여 각기둥과 원통형을 기본으로 쉽게 입력하게 하였으며, 분석된 데이타는 아래 보인 골조구조(frame)로 저장된다.

```
[BOX
    type : cubic
    size : 10 10 10
    position : p1
    grasp : (0 0 5 0 0 180)
    approach : (0 0 15 0 0 0)
    supported-by : conveyor
    supports : nil ]
```

3.2 Discourse module

작업의 입력 과정을 진행하는 부분으로 다음 두 부분으로 구성된다.

- 프로그램 상담자 (Expert Programmer Consultant)

SNU-ARPS의 명령어를 관리하고, 완전한 프로그램이 생성될 수 있도록 프로그램에 필요한 여러 인자를 질문하며, 사용자가 의문을 갖는 부분에 대해 설명하는 기능을 담당한다. SNU-ARPS의 명령어들은 기존의 작업중심 언어의 명령어들과 비슷한 형태로, 그 성격에 따라 크게 다음의 다섯 부류로 분리하였다.

- System 자체에 관한 명령어
- File 처리에 관한 명령어
- Edit에 관한 명령어
- DataBase 관리에 관한 명령어

- Task 에 관한 명령어

• 윈도우 관리자 (Window Manager)

디스플레이 화면을 관리하는 부분으로, 프로그램의 진행 상황에 따라 몇개의 윈도우가 선택적으로 사용된다.

3.3 Domain Expert

로보트 프로그램시 전문적인 의사 결정을 담당하는 부분으로 다음과 같이 구성된다.

• 경로 계획부 (Path Planner)

작업공간의 상태로부터 접근 위치, 후퇴 위치및 충돌이 일어나지 않기 위한 중간 경위점등을 선택한다. 현재의 충돌 회피 방법은 지극히 간단한 형태로 Z축 방향으로의 경로 변환만이 허용되며, 여기에 사용된 가정은 다음과 같다[11,12].

- 로보트 손 끝의 충돌만을 고려한다.
- 이동시 로보트가 잡고 있는 물체의 방향은 변경하지 않는다.
- 경로 변경은 Z축 방향으로만 허용한다.
- 작업공간 내의 고정 물체에 대한 충돌만을 고려하며 장애물의 모양은 간단한 다각형으로 한다.

위와 같은 가정하에 충돌 회피를 위한 중간 경로를 선택하는 방법은 다음과 같다.

i) 장애물 모델의 근사

그림 2 와 같이 손에 잡고 있는 물체에 따라 물체의 방향 변화에 의한 영향이 무시될만큼 충분히 큰 반지름을 갖는 구로 로보트의 손끝을 근사시키고, 손 끝을 한 점으로 다룰 수 있도록 그림 3 에서와 같이 모든 장애물의 크기를 구의 반지름만큼 증가시켜 조정한다.

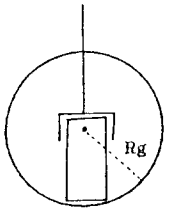

그림 2 로보트 손 끝의 근사 과정

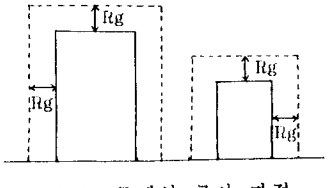

그림 3 물체의 근사 과정

ii) 수평면 확인

이동 경로의 시작점과 끝점에 의해 이루어진 직선과 작업 공간 내의 모든 장애물을 x-y 평면으로 투영 (projection) 시켜 2차원에서 이들의 교점을 구함으로써 충돌 가능성이 있는 물체들을 찾아낸다 (그림 4 참조).

iii) 수직면 확인

ii) 에서 구한 물체들을 이동 경로가 이루는 직선이 x-y 평면으로 투영될 때 이루는 면으로 절단하고, 그 면에서 이동 경로의 시점과 종점이 이루는 직선의 기울기와,시점 또는 종점과 ii)에서 구한 교점이 이루는 직선의 기울기를 비교함으로써 충돌이 일어나지 않도록 적절한 교점을 포함하여 경로를 변경한다.

그림 5에 iii) 의 과정을 보였는데, $\overline{SF}$의 기울기가 $\overline{SO_1}$의 기울기 보다 작으므로 O_1을 경위하도록 경로를 2 로 변경하며, $\overline{O_1F}$의 기울기가 $\overline{O_1O_3}$의 기울기 보다 작으므로 O_3를 지나도록 변경한 경로 3을 얻는다.

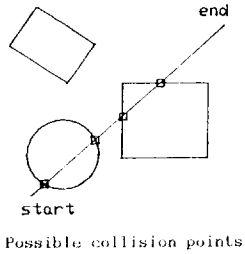

그림 4 수평면 확인 과정

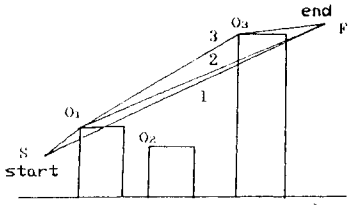

그림 5 수직면 확인 과정

• 작업 공간 관리자 (Work-space Checker)

이 모듈은 역기구학 (Inverse Kinematics) 을 풀고 결과를 로보트의 각도 제한값(joint limit)과 비교하여 도달 불가능한 지점이 프로그램되는 것을 방지하기 위한 부분이다.

• 작업 관리자 (Task Manager)

각 작업에 따라 작업에 필요한 인자를 모아 놓은 QUERY와 작업이 수행되기 위한 선결 조건 (Precondition)과 실행부(Action)를 표현한 규칙이다. 실제 프로그래밍 과정을 진행시키는 부분으로, 아래에 PICK 동작에 관한 예를 보인다.

```
(RULE PICK
      (QUERY target-obj
             time
             sensory-cond)
      (PRECOND (no-supports check)
               (hand-empty check)
               (in-workspace object check)
               (weightbound check)
               (sizebound check)
               (path plan)
               (determine speed))
      (ACTION  (move-to approach)
               (move-to object)
               (grasp)
               (update for pick)))
```

• 오류 확인부 (Logical Error Checker)

로보트 프로그램에 익숙하지 않은 사용자들은 비논리적인 즉, 현재환경에서는 실행할 수 없는 명령어를 입력하기도 한다. 이 모듈은 이러한 오류를 방지하기 위해 작업 상황을 이해하고 있다가 적절한 조언을 하여 사용자의 프로그래밍을 돕는다.

• 속도 관리자 (Speed Manager)

많은 로보트 언어들이 속도를 지정함에 있어 각 관절 모터의 최대 속력에 대한 상대 속도로써만이 가능하다. 사용자는 이러한 상대 속도의 지정에 익숙하지 않으므로 이 모듈은 작업을 수행하는데 필요한 대강의 시간을 추정하여 사용자가 의도하는 작업 속도를 결정하도록 돕는다. SNU-ARPS는 로보트의 이동에 있어 접근 지점까지는 고속으로 이동하고, 원하는 지점 가까이에서는 정확한 저속 직선운동으로 이동함을 원칙으로한다. 이러한 원칙을 반영하여 SNU-ARPS가 속도를 계산하는 방법은 다음과 같다.

$$Tr = T - (h * \text{hand-operation-time})$$

$$Vm = \frac{l}{Tr} \frac{n}{\sum_{i=1}} Di \, Wi$$

$$Vi = Wi * Vm$$

Di:각 이동 구간의 거리
Wi:각 이동 구간 속도의 가중치 ($0<wi\leq 1$)
n:구간 수
h:핸드의 on/off 횟수
T:동작을 마치도록 요구된 시간
Tr:이동시 이용할 수 있는 시간
Vm:최대 속도로 이동하는 구간의 속도
Vi:각 이동 구간의 속도

이렇게 구해진 mm/s 단위의 속도는 상대 속도로 변환되어 실제 프로그램에서 이용된다.

3.4 Coder

위에서 언급한 과정을 통하여 생성되는 것은 실제 로보트 프로그램이 아니라 작업 및 이에 필요한 변수들을 포함한 프로그램 모델이다. 실제 로보트 언어가 생성되는 곳은 바로 이 부분인데 시스템이 이러한 구조를 갖는 것은 여러가지 목적언어를 생성하도록 확장하기위해 목적언어에 무관한 부분과 목적언어와 관계 깊은 부분을 분리하였기 때문이다.

이 모듈은 finite-state automata의 생성규칙(production rule)의 형태로 만들어지며 목적언어에 따라 필요한 데이터 베이스만이 선택적으로 사용된다.

4. SNU-ARPS 시스템 동작의 예

다음은 cup을 잡아 box 위에 놓고 position P1으로 돌아가는 간단한 작업의 예이다. 그림 6은 작업 공간을 설명하는 화면으로 명령어를 선택하는 Command 윈도우, 작업의 수행 상태및 시스템 셋팅상황을 보여주는 Status 윈도우, 사용자와의 대화를 위한 Qurey 윈도우의 동작을 볼 수 있으며, 그림 7은 마지막 MOVE 동작을 입력하는 과정을 보여 준다. 이 작업을 수행하기 위해 생성된 SNUL-I 프로그램 리스트는 그림 8에서 볼 수 있는데 이 프로그램은 접근지점까지는 고속으로 이동하고, 최종 위치까지는 저속으로 이동하는 시스템의 규칙과 필요한 위치 데이타를 구하는 과정을 보여 준다.

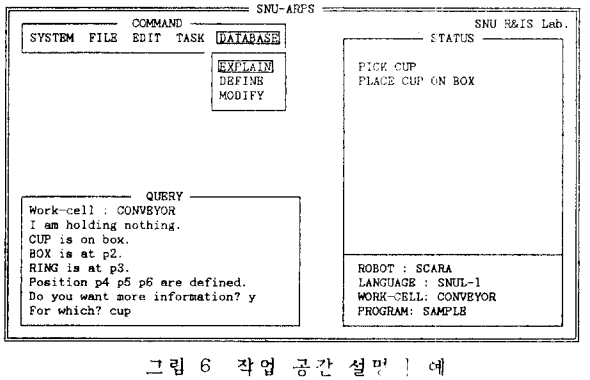

그림 6 작업 공간 설명 I 예

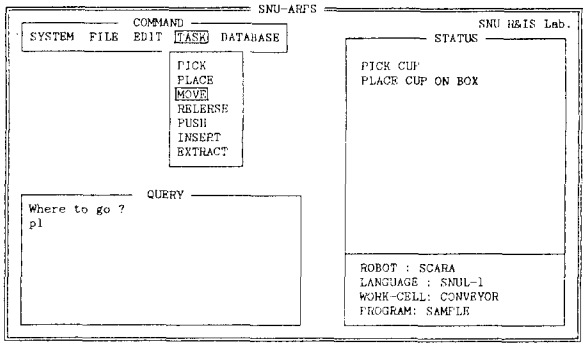

그림 7 작업 프로그램 과정의 예

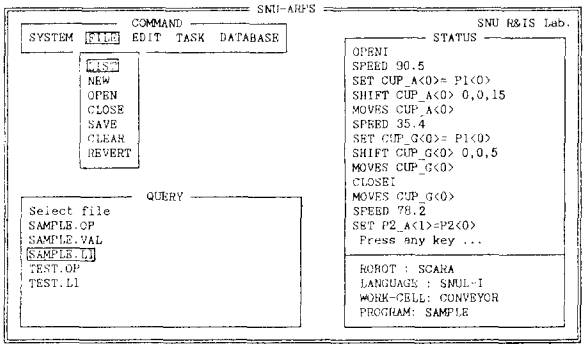

그림 8 작성된 프로그램 리스트

5. 결론 및 향후 연구 방향

앞에서 설명한 바와 같이 메뉴 위주의 대화식으로 구성된 SNU-ARPS는 Pick & Place 수준의 간단한 작업에 대해 실행 가능한 로보트 프로그램을 작성할수 있으며 로보트 프로그램을 용이하게 하고자했던 기존의 시도에 비해 다음과 같은 장점을 지닌다.

- 로보트 프로그램이 일정한 규칙에 의해서만 생성되는 것이 아니라 사용자가 여러 가능성 중에서 선택할 수 있으므로 사용자의 의사가 좀 더 세부적으로 반영된다.
- 다양한 설명기능으로 사용자 교육의 효과가 있다.
- 사용자가 작업중심언어 수준의 문법조차 익힐 필요가 없다.
- 프로그램시 문법상의 에러 이상의 논리적 에러도 확인되므로 프로그래밍에 관한 지식이 없는 사용자도 쉽게 이용할 수 있다.
- 여러 종류의 로보트 및 로보트 언어에 대해 쉽게 확장할 수 있다.

반면, 현재 SNU-ARPS는 프로그램시 필요한 많은 전문적인 의사 결정 부분들 - 충돌 회피 경로 선택, 잡는 위치의 선정등 - 을 매우 단순하게 처리하고 있다. 이러한 의사 결정 부분들을 좀 더 지능적으로 처리하도록 보완하여, 시스템을 작업 중심 언어의 수준으로 향상시키는 것이 앞으로 남은 중요한 과제라 하겠다. 즉, 충돌 회피를 위해 좀 더 효과적인 최단경로 선택 알고리즘을 개발하여 구현하고, 더욱 다양한 물체를 모델링하고 인식하여 적절한 잡는 위치를 선택하도록 하며, 센서 정보를 처리를 위한 지식을 첨가하여 보다 복잡한 조립작업에 응용하고자 한다. 또한 본 연구실에서 개발중인 그래픽 시뮬레이터와의 결합으로 SNU-ARPS에서 작성된 프로그램의 실행을 확인하고 보완하도록 함으로써 보다 효율적인 프로그래밍이 가능해질 것으로 기대된다.

[참고문헌]

[1] T. Lozano-Perez, "Robot Programming," Proceedings of the IEEE, Vol.71, No.7, July, 1983.

[2] S.Bonner & K.G.Shin, "A Comparative Study of Robot Languages," IEEE, Computer, 1982.

[3] Unimation Inc., User's Guide to VAL, Ver.12, June, 1980.

[4] 이기동, "SCARA 형 로보트의 프로그래밍 언어 구성에 관한 연구," 서울대학교 공학석사 학위논문, 1985.

[5] A.Naylor et al., "PROGRESS- A Graphical Robot Programming System," Proc. IEEE Int'l Conf. Robotics & Automation, 1987.

[6] E.Mazer, Geometric Programming of Assembly Robots(LM-GEO), Advanced Software in Robotics, North Holland, 1983.

[7] R.J.Popplestone et al., "RAPT- A Language for Describing Assemblies," The Industrial Robot, Vol.5, No.3, p.131, 1978.

[8] L.I.Lieberman et al., "AUTOPASS: An Automatic Programming System for Computer Controlled Mechanical Assembly," IBM J. of Research & Development, Vol.21, No.4, July, 1977.

[9] T.Lozano-Perez & P.H.Winston , "LAMA : A Language for Automatic Mechanical Assembly," Proc. 5th IJCAI, 1977, pp.710-716.

[10] T.Lozano-Perez et al., "HANDEY:A Robot System that Recognizes, Plans, & Manipulates," Proc. IEEE Int'l Conf. Robotics & Automation, 1987.

[11] J. Khouri, "An Efficient Algorithm for Shortest Path in Three Dimensions with Polyhedral Obstacles," Proc. American Control Conference, 1985.

[12] T.Lozano-Perez et al., "An Algorith for Planning Collision-free Paths Among Polyhedral Obstacles," Com. of the ACM, Vol.22, No.10, 1979.