'95 대한전기학회 하계학술대회 논문집(1995. 7. 20~22)

객체 지향 방식의 로봇 그래픽 시뮬레이터 개발에 관한 연구

허원
공주대학교 공과대학 전기공학과 조교수

A Study on the development of an Object Oriented Robot Simulator

Ho, Won
Department of Electrical Engineering
Kongju National University

ABSTRACT

It is very important to understand the 3-dimensional movement of a robot manipulator in developing a robot manipulator system. The robot design software package is required to test the specification. Usually these robot simulators are turn-key based and not possible to be used on the other robot system. The aim of this paper is to develop a general purpose robot simulator. AutoCad is selected for the developing environment to avoid the difficulties of building a cad system from the scratch. Because Autocad provides a semi-open structure to a Lisp programmer, it is quite successful to achieve the goal of building the simulator. At the moment the kinematic analysis is possible on the package. Further study will be advanced to the application and analysis of dynamic area, which would not be that difficult to be implemented, considering the many third party tools available for Autocad.

1. 서론

로봇시뮬레이터는 제작된 로봇을 대상으로 작업상황에서의 이동경로를 산출, 확인하려는 목적, 로봇을 개발할 때 개발 로봇의 운동학적, 동역학적인 특성을 연구 하기위한 목적, 또는 교육적인 용도로써 3차원공간에서의 로봇의 동작 특성이해를 위한 목적등으로 이용된다. 대부분의 경우 특정 로봇 매니퓰레이터에 대해서는 제작회사가 관련 로봇 시뮬레이터를 제공하는 경우가 많고 또한 특정 로봇언어를 사용함으로써 자체 로봇의 기능을 지원하는 경우가 많다. 이러한 경우 전체의 사용모듈이 Turnk-key 방식으로 되어 특정 로봇에 대한 의존성이 높게 되고, 필요한 부분만을 이용할 수 없게 된다.

이러한 문제로 범용성, 확장성, 경제성이 있는 시뮬레이터의 개발이 요구되는데, 이런한 목적을 만족할 수 있는 개발 환경으로 Autocad 를 선택하였다. 이는 PC 기반 CAD 도구로서는 가장 넓은 시장을 점유하고 있으며, 일반 교육 연구 용으로 경제성있게 구입할 수 있고, 준 개방 시스템으로서 Autocad 의 개발언어인 Autolisp 을 사용하면 하위 레벨에서의 CAD환경을 이용할 수 있기 때문이다. 또한 많은 Third Party 프로그램을 이용할 수 있으므로, 필요한 기능들을 개발 패키지와 연계하여 사용할 수 있다.

일차적인 개발로서 3차원에서의 로봇의 움직임에 초점을 두어 운동학적인 면을 고찰할 수 있는 로봇 시뮬레이터를 제작하였다. 이는 필요에의한 관절과 그리퍼를 설계하고 이러한 객체들이 각 관절의 상태에 따라서 작업환경에서의 운동상태를 확인할 수 있다. 동역학적인 해석에 대하여는 다루지 않았으며, 각 로봇 매니퓰레이터의 요소들은 객체 지향 방식의 설계에 의하여 작성되도록 개발환경을 구축하였다. 이러한 방식으로 여러가지 형태의 관절을 만들 수 있고 이들의 속성을 객체 지향 방식의 장점인 유전성을 이용하여 코드의 재사용율을 높이도록 하였다.

2. 객체 지향방식의 개발

Autolisp 은 객체 지향형의 언어가 아니다. Autocad 에서 이용되는 다른언어로서 ADS(Autocad Development System) 은 현재 주목받고 있는 C++ 의 객체 지향적 프로그램을 이용하려는 의도에서 개발 되었다. 이는 컴파일러 형태의 개발툴인데 경우에 따라서는 interpreter형태의 개발툴인 Autolisp이 더욱 유용한 경우가 있다. 즉 로봇의 부품을 개발하고 추가, 삭제, 수정등의 작업을 요하는 설계과정에 대하여는 수시로 현재의 작업상태에 대한 응답식 환경을 제공하는 interpreter 형의 언어가 더욱 유리하다. 이러한 Autolisp 에 객체 지향적 기능을 부여 하기위해서 추가 하여야할 내용은 세 가지로 들 수 있다.

- Data Encapsulation

하나의 객체를 설계할 때에 원하는 정보를 인수화 한다. 예로서 표1의 프로그램을 살펴 보자. 하나의 직사각형을 객체의 형태로 표시하는 방식을 보여준다. 직사각형은 길이와 높이로 표시된다. 즉 객체 rect 는 width, length, low_corner 등의 데이타들의 집합으로 표현되며, 각각의 데이타들은 상호간의 값들을 reference 할수 있는 방법을 갖고있다. 현재의 프로그램에서는 " the" 라는 이름의 함수로 작성되었다. 즉 직사각형은 높이, 길이, 중심이 주어진 상태에서 좌하단의 꼭지점은 이들의 데이타를 이용함으로서 구할 수 있다. ((the 'low_corner) 의 명령은 좌하단 꼭지점의 값을 구한다) 또한 (the 'draw) 의 명령은 도형을 그리게 된다. 즉 draw인수에는 모든 인수들의 값에 근거하여 도형을 그리는 기능을 집약하여 둔다. 하나의 객체는 단순히 그것을 구성하는 인수들의 동일한 집합으로서 표현가능하며 이러한것이 결과적으로 프로그램의 해석및 확장을 용이하게 하여주는 역할을 한다.

```
(rect
   (center (0 0 0))
   (low_corner (list (- (car (the 'center))
                        (/ (the 'width) (float 2)))
```

```
                (- (nth 1 (the 'center))
                (/ (the 'height) (float 2)))
                 (nth 2 (the 'center))))
(width 1)
(height 1)
(draw (command "line" (the 'low_corner)
                (list (+ (car (the 'low_corner))
                (the 'width))
                (nth 1 (the 'low_corner))
                (nth 2 (the 'low_corner)))
                (list (+ (car (the 'low_corner))
                      (the 'width))
                (+ (nth 1 (the 'low_corner))
                      (the 'height))
                (nth 2 (the 'low_corner)))
                    (list (car (the 'low_corner))
               , (+ (nth 1 (the 'low_corner))
                      (the 'height))
                (nth 2 (the 'low_corner)))
                    "c")))
```

표 1. 직사각형의 객체지향적 표현방식

- Inheritance

하나의 객체를 생성하고 이것을 이용하여 또 다른 객체를 생성할 경우 원래의 객체가 가지고 있던 형질을 대부분 다시 이용하는 경우 가 있다. 결국 이 것은 객체의 인수들의 정보와 그 상호간의 연결관계를 유전 받는것을 의미한다. 원하는 객체의 인수가 현재의 객체에서 정의 되지 않은 경우에는 이의 모 객체를 찾아서 올라가는 기능을 제공 하여야한다. 이러한 결합관계는 하나의 객체내에서는 parents 라는 인수로 다른 객체에서는 children 이라는 인수로 서로 연결하도록 한다. 표2는 그 예로서 하나의 관절은 다른관절과 어떠한 관계에 있는가를 설명하는 parents 와 children 의 인수를 포함하고 있음을 알 수 있다.

```
(basedraw
      (local_point (0 0 0))
      (model_point (0 0 0))
      (rotate 0))

(robot
      (a-kind-of (basedraw
            rect))
      (children (arm1 body
            head))
      (width 5)
      (height 8)
      (length 5)
      (rotatez 0))

(arm1
      (a-kind-of (box0))
      (children (arm2))
      (parents (robot))
      (width 5)
      (height 0.3)
      (length 1)
      (local_point (-2 0 0))
      (model_point (0 0 2))
      (rotatez 45))

(arm2
      (a-kind-of (arm1))
      (parents (arm1))
      (model_point (2 0 0))
      (rotatez 90))
```

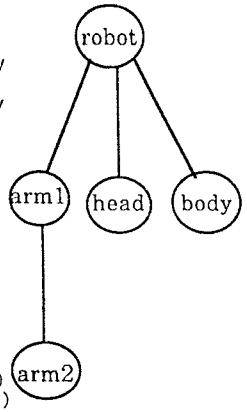

표2. Inheritance 의 예제

- 객체 상호간의 위치 제어

Autocad는 여러가지의 좌표를 지정할 수 있는 기능을 갖고있다. 이러한 기능을 이용함으로써 모 객체의 좌표내에서 정의 되는 자손객체들의 좌표를 효과적으로 제어할 수 있는데 이는 model_point 와 local_point 의 인수 이다. 이는 다음 절에서 상세히 설명하도록 한다.

이상의 항목들을 포함하며 이를 운용하기위하여 패키지가 갖추어야할 기능은 다음과 같다.

1) 작성된 프로그램을 읽어 들이는 기능
2) 각각의 객체의 이름에 대하여 인수들을 할당하 는 기능
3) 객체의 인수를 inherit 받는 기능
4) 필요한 객체에서 특정인수의 값을 회수하는 기능
5) 하나의 객체를 독립적으로 화면에 출력하는 기능
6) 하나의 객체를 관련 객체와 함께 화면에 출력하는 기능

3. Autocad 의 3차원 좌표계 변환

Autocad 는 도형요소를 표현하기위하여 UCS(User Coordinate System)을 사용하는데 이를 이용하여 각각의 관절에 대한 좌표계의 변환이 가능하다. 표3 은 이러한 방식을 이용하여 좌표계를 변환한 뒤에 각 관절을 그리는 프로그램을 나타낸다.

```
(defun expand_n (nlist done / tx ty tx mo lo)
   (setq *c_part* (car nlist))
   (setq *instan* (eval *c_part*))
   (setq *instan* (r_add_frame *instan* (getmixinlist
         *instan*)))
   (setq tx (the 'rotatex) ty (the 'rotatey)
       tz (the 'rotatez) mo (the 'model_point)
       lo (the 'local_point))
   (command "ucs" "or" mo)
   (if tz (command "ucs" "z" tz))
   (if ty (command "ucs" "y" ty))
   (if tx (command "ucs" "x" tx))
   (command "ucs" "or" (mapcar '- '(0 0 0) lo))
   (if () (the 'quantity) 0)
      (multiprocess)
      (the 'draw))
   (princ *c_part*)
   (getpoint)
   (cond
     ((null nlist) nil)
     (T (expand_n (append (the 'children)
                 (cdr nlist))
                 (append done (list (car nlist))))))

   (if tz (command "ucs" "z" (- 0 tz)))
   (if ty (command "ucs" "y" (- 0 ty)))
   (if tx (command "ucs" "x" (- 0 tx)))
   (command "ucs" "or" (mapcar '- '(0 0 0) mo))
)
```

표 3. 3차원 좌표변환을 위한 프로그램

model_point는 현재의 객체의 어느 부분에 자객체의 원점이 위치하는가를 나타낸다. local_point 는 자객체의 어느부분이 자객체좌표의 원점에 놓이게 되는가를 지정하는 인수 이다. expand_n 는 지정된 객체로 부터 모든 후손들을 depth first search 방식에의하여 평가하며 평가과정에서 각각의 객체 고유좌표에서 도형을 그리게된다.

4. 결과 및 고찰

시범적인 로봇을 완성하기 위한 객체지향 프로그램은 표 4 에 주어졌고, 이의 실행 결과는 그림에 주어졌다. 그림은분할된 Autocad 의 화면에 생성되는 3가지의 다른 관절의 위치 각각에 대한 4가지 viewport 의 투시도를 보여준다.

```
(robot
   (a-kind-of (basedraw box0))
   (children (arm1))
   (width 2)
   (height 0.5)
   (length 3)
   (rotatez 0))
```

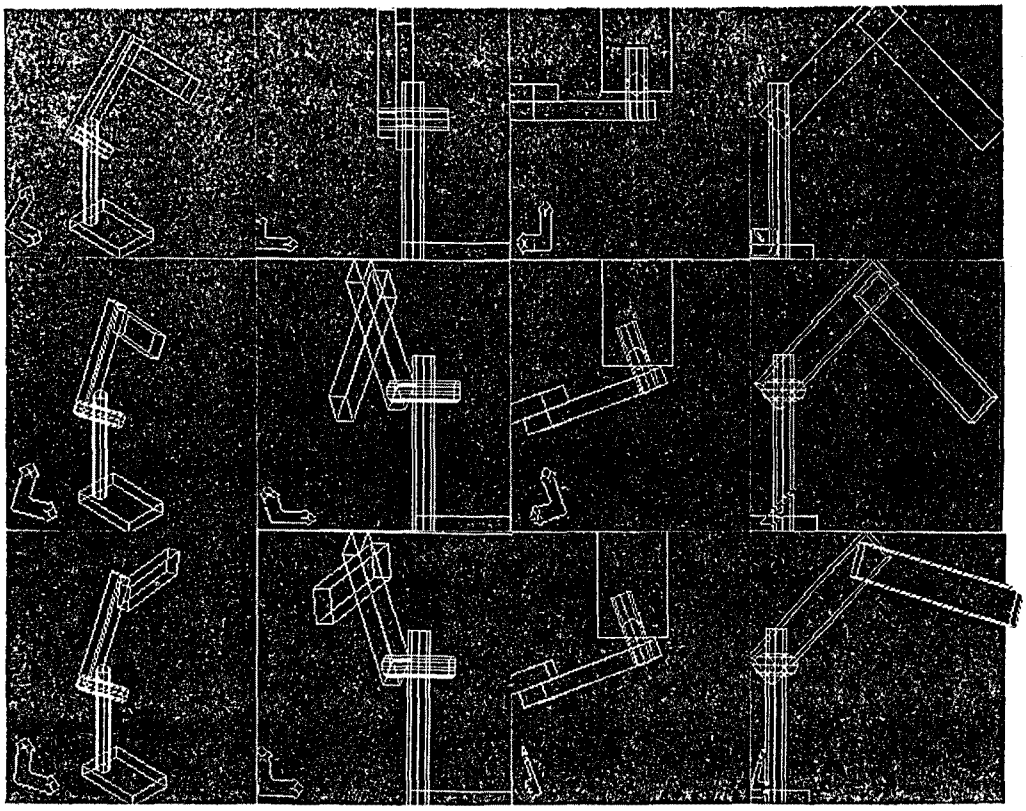

그림. 로봇 시뮬레이터에 의한 실행결과
상단: 기준위치, 중간단: 주축의 회전, 하단: 최종관절의 회전

```
(arm1
  (a-kind-of (cyl))
  (children (arm2))
  (parents (robot))
  (depth 5)
  (radius 0.3)
  (local_point (0 0 0))
  (model_point (0 0 1)))

(arm2
  (a-kind-of (cyl))
  (children (arm3))
  (parents (arm1))
  (depth      2)
  (radius 0.3)
  (model_point (0 0 4))
  (local_point (0 0 1))
  (rotatex 90))

(arm3
  (a-kind-of (box0))
  (parents (arm2))
  (children (arm4))
  (width 1)
  (height 0.5)
  (length 5)
  (model_point (0 0 -0.5))
  (local_point (0 0 0))
  (rotatez 45))

(arm4
  (a-kind-of (arm3))
  (parents (arm3))
  (children ())
  (model_point (0 4 -0.5))
  (rotatez 90))
```

표 4. 실험로봇의 제작 프로그램

5. 결론

Turn-key방식 이 아닌 일반적인 로봇 시뮬레이터를 개발하였다. 현재는 운동학적인 측면에서의 이용만이 가능하나. Autocad의 Solid기능으로 확장하고 개발되어 있는 Third Party Program을 연결하게 될 경우 질량 해석을 고려한 동역학적인 해석도 용이하게 될 것이다. 한편 Autocad 와 연결된 animation 패키지를 이용할 경우 경로추출에 대한 3차원에서의 시각적 확인이 용이할 것이다. 현 단계에서는 GUI 부분을 보강할 필요가 있으며, 더욱 다양한 종류의 관절객체를 확보할 필요성이 있다. 현재 까지 개발된 유틸리티는 무상으로 공개할 예정이다.

6. 참고 문헌

[1] John F. Nethery and Mark W. Spong, " Robotica: A Mathematica Package for Robot Analysis" , IEEE Robotics & Automation Magazine, pp13~20, March, 1994

[2] David J. Miller & R. Charleene Lennox, " An Object-Oriented Environment for Robot System Architectures" , IEEE Control Systems, Vol 11, No 2, pp 14~23, Feb. 1991

[3] Robert B. White, R. K. Read, M. V. Koch, and R. J. Schilling, " A Graphics Simulator for a Robotic Arm" , IEEE Transactions on Education, pp 417~429,Vol. 32, No. 4, Nov. 1989

[4] Rusty Gestner & Joseph Smith, " Maximizing AutoLisp" , New Riders Publishing, 1992