컴퓨터 그래픽스를 이용한 로보트 매니퓰레이터의 구현 방법

(Realization of Robot Manipulators Using Computer Graphics)

張 源*, 鄭 明 振*, 卞 增 男*

(Won Jang, Myung Jin Chung and Zeungnam Bien)

要 約

본 논문에서는 로보트의 구조를 기술하는 한 방법을 제안하고, 제안한 방법을 로보트의 설계를 돕는 도구로 사용할 수 있음을 보였다.

로보트를 여러 링크들이 모여서 구성하는 결합체로 파악하였으며, 각 링크들은 多角柱의 형상으로 근사화하였다. 근사화된 링크들을 모아 하나의 로보트를 기술하기 위하여, homogeneous transformation과 관련된 명령어들을 정의하여 사용하였다. 정의된 명령어들을 사용하면, 복잡한 결합구조를 갖는 각종 로보트에 대하여, 그 결합관계를 손쉽게 표현할 수 있었다.

Abstract

In this paper, we developed a method of describing the structure of robot arms and a systematic way to use it as a robot-design-aid tool.

To describe the structure of robot arms, a robot was regarded as a collection of various links in the form of polygonal pillars. With the aid of the homogeneous transformation, a set of commands for describing the information on how all the links in the robot are connected was defined and used in graphically realizing complex link-based structures including robot manipulators.

I. 서 론

최근에 이르러 생산성의 향상을 위한 공장자동화가 진행되면서, 산업현장에서 로보트를 사용하는 양이 급격히 증가하고 있다. 이와 같은 로보트 사용량의 증가로 인해 로보트의 사용자와 생산자들 모두 새로운 문제점에 직면하게 되었다.[1,2]

산업현장의 로보트 사용자가 지적하는 문제점으로는 우선 플레이-백 방식의 교시방법으로는 고가의 첨단기술 장비인 로보트의 효율적인 사용이 어렵다는 것이다.[2,3,4] 손 쉬운 교시를 위한 다양한 로보트 언어가 개발되고 있으나, 원하는 움직임을 해석적으로 기술하기가 쉽지 않다는 점을 보완해야 하는 문제점이 있다.[6]

로보트를 설계·생산하여 공급하는 측면에서는, 설계에 필요한 여러가지 변수들을 한번에 고려할 수 없었던 까닭에 종래의 시행착오 방식의 설계과정을 답습할 경우 날로 증가하는 사용자의 다양한 요구조건을 충족시키기 위한 설계 및 생산에 소요되는 기간 및 비용이 증가한다는 것이 지적되고 있다.[2,4,6]

앞에서 언급한 문제점들을 해결하기 위한 연구가 그동안 활발히 진행되어 왔으며, 그 해결책들 중, 컴퓨터 그래픽스를 이용하여 시뮬레이션을 하는 방법이 제안되어 많은 각광을 받고 있다.[1,4~7]

*正會員, 韓國科學技術院 電氣및電子工學科
(Dept. of Elec. Eng., KAIST)
接受日字 : 1986年 6月 11日

그러나 개발측면을 고려한 그래픽 시뮬레이터에 대한 연구는 미진한 편이며, 그 이유는 다음과 같은 몇가지 선결과제가 있기 때문이다.[2,6] 첫째, 로보트를 구성하는 링크들을 묘사하는 방법과, 여러 링크들이 모여서 로보트를 구성하는 과정을 기술하는 방법이 결정되어 있어야 한다.[2,4,6] 둘째, 로보트의 동력학적인 특성을 명확히 표현할 수 있는 방법이 있어야 한다.[2,4,6] 셋째, 설계된 로보트의 각종 정격이 원하는 대로 되었는지를 판단할 수 있는 방법이 준비되어 있어야 한다.[6]

본 논문에서는 로보트를 강체인 링크들이 모여서 이루어진 결합체로 파악하고,[2] 그 결합체를 기술하는 새로운 방법을 제안하였다.

제안한 방법의 내용에 대해서는 제 2 장에서 상세히 설명하였다. 또한 제안한 방법을 사용하여 임의의 형태를 갖는 로보트를 구성하는 과정과 그 결과를 제 3 장에서 보였다.

Ⅱ. 로보트 구조의 일반적 기술방법

본 논문에서 제안하는 로보트 구조의 기술방법을 세 부분으로 나누어 설명한다.

처음에, 링크를 모델링하는 방법과, 다음으로 어떤 링크들이 모여서 로보트를 구성하는지를 지정하는 방법을 설명하고, 마지막으로, 로보트를 구성하는 여러 링크들의 상호 결합관계를 기술하기 위해 정의한 명령어들에 대해 설명한다.

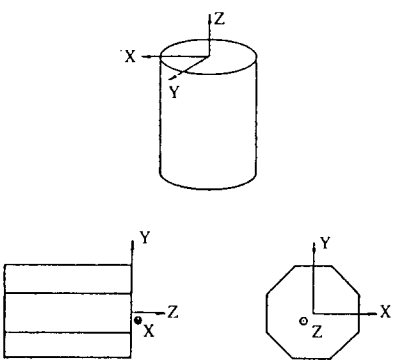
1. 링크의 모델링

로보트를 구성하는 여러 링크들은 그 형태가 매우 다양하고 복잡하기 때문에 간단한 모양의 기본적 primitive들만을 사용하여 모델링하는 경우, 그래픽 디스플레이 결과가 조악해진다는 문제점이 있다. 반면에, 그래픽 디스플레이의 충실도를 높이기 위해서 여러가지 형태의 primitive들을 사용하여 모델링하는 경우에는 추가로 요구되는 데이타 및 알고리즘들로 인한 계산량의 증가라는 부담을 안게 된다.

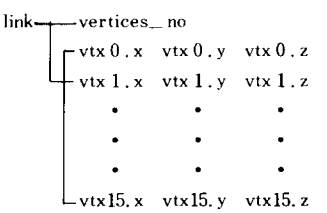
이에, 계산량을 줄이면서도 그래픽 디스플레이의 충실도를 유지할 수 있는 모델링 방법으로서 '임의의 형태의 多角柱를 이용한 근사화 방법'을 고안하였다. 이 방법은 N개의 角을 갖는 임의의 형태의 角柱를 써서 링크들을 모델링하되 링크의 형태가 복잡한 정도에 따라 角의 수를 변화시켜 사용하는 것을 내용으로 하고 있다.

이 방법의 장점으로는, 임의의 형태를 갖는 角柱라는 통일된 primitive를 사용하기 때문에, 각종 연산처리를 위한 알고리즘이 매우 간단해진다는 점을 들 수 있다. 아울러, 복잡한 형태의 링크에 대해서는 많은 角을 갖는 角柱를 써서 모델링하지만(그림 2 참조), 단순한 형태의 링크에 대해서는 三角柱 내지는 四角柱 정도의 간단한 primitive를 사용하기 때문에(그림 6 참조), 전체적으로는 모델링을 충실히 하면서도 계산에 소요되는 데이타 양을 최소한으로 줄일 수 있다는 것도 큰 장점의 하나이다.

'임의의 형태의 多角柱를 이용한 근사화 방법'을 써서 圓柱를 正八角柱로 모델링하는 경우를 그림1(a) 에 보였다. 그림 1 (b)에 보인 것은 그때 생성되는 자료구조이다. 이때 데이타로서 입력되는 것은 각 꼭지점들의 x, y, z 좌표들인데, 그 입력순서는 우선 윗면을 구성하는 꼭지점들을, 위에서 보아서 반시계 방향으로 입력시킨 뒤, 같은 방향의 순서대로, 밑면을 구성하는 꼭지점들의 좌표를 입력한다. 여기서 아래 윗면의 크기가 다른 角柱의 경우도 포함시키기 위해서 모든 꼭지점의 좌표를 필요로 하고 있다.



(a)



(b)

그림 1. 圓柱의 모델링
Fig. 1. Modelling a cylinder.

좌표계산의 기준이 되는 직교좌표계의 기준축들은 설계자의 임의로 설정할 수도 있게 하였다.

산업용 로보트에 자주 사용되는 형태의 링크들을 모델링하여 데이타 베이스로 만들어 놓은 몇가지 예를 그림 2 에 보였다.

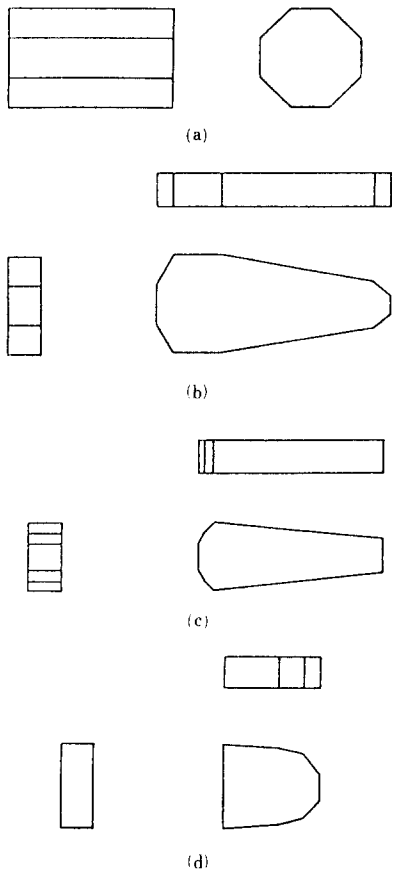


그림 2. 모델링된 여러 링크들의 예
Fig. 2. Some of modelled links.

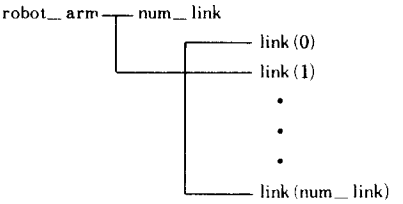


그림 3. 로보트 하나에 대한 자료구조
Fig. 3. Data structure for a robot.

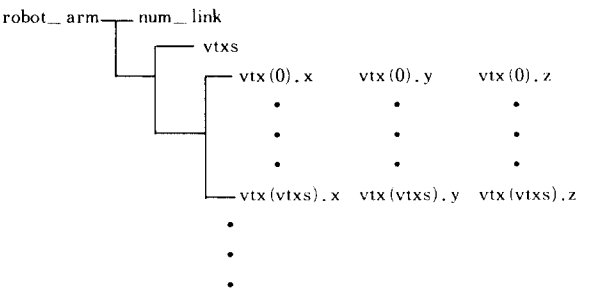


그림 4. 로보트에 대한 상세한 자료구조
Fig. 4. Data structure for a robot in detail.

2. 링크들의 collection

본 절에서는 이미 모델링되어 데이타 베이스에 축적되어 있는 여러 링크들 중에서 어떤 링크들을 써서 로보트를 구성할 것인가에 대한 정보를 기술하는 방법에 대해서 논한다.

현재 생산되고 있는 산업용 로보트만 보더라도 그 구성형태 및 링크의 갯수 등이 무척 다양하다. 이런 이유로 인해서, 어떤 링크들을 사용한다고 지정하는 것 뿐만 아니라, 전체를 몇 개의 링크를 써서 구성하려 한다는 것을 나타내는 정보가 또한 필요하다.

이에 본 논문에서는 하나의 로보트를 기술하기 위한 자료구조를 그림 3과 같이 설정하였다. 가장 아래요소인 'link'에는 II-1절에서 설정한 형태의 데이타들이 위치하게 되어 전체적으로는 그림 4와 같아진다. 이때 'num-link'라는 변수는 정수로서 로보트의 구성에 소요되는 링크의 수를 지정하는데 사용된다.

이 방법의 장점은 우선 로보트의 구성에 소요되는 링크의 갯수에 제한을 받지 않는다는 점이다. 사용자가 임의의 형태를 갖는 로보트를 설계할 수 있도록 한다는 목적에 비추어 생각해 볼 때, 구성에 소요되는 링크의 수에 제한받지 않을 것은 필수적인 조건이라 할 수 있다.

본 논문에서는 그림 4의 구조를 갖는 변수들을 'vertex-descriptor'라 지칭한다.

3. 링크들의 connection

본 절에서는 모아진 링크로 로보트를 구성하기 위해서는 각 링크들이 어떻게 결합되어야 하는가 하는 것을 기술하는 방법에 대해 논하고자 한다.

이미 설계 제작되어 있는 로보트를 해석적으로 연구하기 위한 구조표현방법에 대해서는 많은 연구가 진행되어 왔고 좋은 방법도 많이 제안된 바 있다.[12,13] 그러나 원하는 구조를 갖는 로보트를 합성한다는 측면에서, 로보트라는 구조물의 특성을 살려서, solid object의 링크들로 구성된 로보트를 표현하는 방법에 대해서는 제시된 바가 매우 드문 형편이다. 그러나 임의의 형태를 갖는 로보트를 설계할 수 있으려면, 로보트 구성에 소요되는 여러 링크의 결합관계를 제한없이 기술할 수 있는 방법이 필요하다.

이에 본 논문에서는 로보트 구성에 소요되는 여러 링크사이의 결합관계를 기술할 수 있는 명령어들을 정의하여 사용하였다.

로보트를 구성하는 여러 링크들의 결합관계는 일견 매우 복잡한 것으로 파악된다. 그러나 아무리 복잡한 형태의 결합이라도 몇 가지의 기본결합을 써서 분해해 볼 수 있고, 역으로 몇 가지 기본적인 결합형태를 갖고서 아무리 복잡하게 결합된 관계도 재현해 낼 수 있다는데 착안점을 두었다.

결합관계를 기술하는데 사용되는 명령어들의 기능으로는, x축·y축·z축을 중심으로 하는 rotation, x축·y축·z축 방향으로의 translation - scaling이 필요하다고 보고 그에 해당하는 명령어들을 표 1과 같이 정의하였다. 이 명령어들은 해당되는 link의 기준 좌표계에서의 x·y·z 축을 중심으로 하고 있으므로, link의 좌표계를 변환시킨 후의 operation은 임의의 축을 기준으로 하는 operation의 효과가 나타난다.

표 1. 정의된 명령어들
Table 1. Defined commands.

Rotation op [deg] about x axis	RRX op
Rotation op [deg] about y axis	RRY op
Rotation op [deg] about z axis	RRZ op
Translation op [mm] along x axis	TTX op
Translation op [mm] along y axis	TTY op
Translation op [mm] along z axis	TTZ op
Scaling op along x axis	SSX op
Scaling op along y axis	SSY op
Scaling op along z axis	SSZ op

여러 명령어들을 모아서 하나의 결합관계를 묘사하는 경우에 가장 혼동의 가능성이 높은 부분은 명령어의 수행순서이다. Translation후의 rotation 결과가 rotation후의 translation 결과가 다르다는 것 등, commutative law가 성립하지 않으므로 확실한 규칙을 정해 놓을 필요가 있다. 본 논문에서는 정의된 명령어들이, 같은 기능의 4×4 homogeneous transformation matrix와 대응하도록 하였고 명령어들의 복합연산은 matrix의 product에 해당하도록 하였다. 그 사용예를 표 2에 보였다.

표 1에서 정의한 명령어들만 갖고서도 어떤 형태의 물체든지 그 정적인 결합관계를 기술하는데는 아무 지장이 없다. 그러나 로보트와 같은 경우에는 joint variable에 따라서 로보트 자신의 형상이 시시각각으로 변화한다는데 문제가 있다. 이를 해결하기 위해서는 링크사이의 정적인 결합관계 외에도 joint variable에 따라 규칙적으로 변화하는 동적인 결합관계도 기술할 수 있어야 한다.

본 논문에서는 rotation, translation, scaling 들의 operand에 원하는 joint variable을 assign할 수 있도록 한, joint variable에 관련되는 명령어들을 추가로 정의하여 이 문제를 해결하였다. 새로 정의한 명령어들은 표 3과 같다.

표 2. 사용 예
Table 2. Operation example.

*TTY 330
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 330 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

*RRY 37
$$\begin{bmatrix} \cos 37 & 0 & \sin 37 & 0 \\ 0 & 1 & 0 & 0 \\ -\sin 37 & 0 & \cos 37 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

*RRX 23
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 23 & -\sin 23 & 0 \\ 0 & \sin 23 & \cos 23 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

*RRY 37. TTY 330. RRX 23
$$\begin{bmatrix} 0.799 & 0.235 & 0.554 & 0 \\ 0 & 0.921 & -0.391 & 330 \\ -0.602 & 0.321 & 0.736 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

표 3. Joint variable에 관계되는 명령어 群
Table 3. Commands related to joint variables.

Rotation about x axis as much as n-th joint variable	JRX n
Rotation about y axis as much as n-th joint variable	JRY n
Rotation about z axis as much as n-th joint variable	JRZ n
Translation along x axis as much as n-th joint variable	JTX n
Translation along y axis as much as n-th joint variable	JTY n
Translation along z axis as much as n-th joint variable	JTZ n
Scaling along x axis as much as n-th joint variable	JSX n
Scaling along y axis as much as n-th joint variable	JSY n
Scaling along z axis as much as n-th joint variable	JSZ n

표 3에서 정의된 명령어들의 operand에는 정수가 오게 되는데, 이 정수에 의해 지정되는 joint variable의 값을 rotation, translation, 또는 scaling의 operand로 삼는다. Joint variable을 operand로 assign 받는다는 것 외에는 모든 것이 표 1에서 정의된 명령어들과 똑같도록 하였다.

명령어의 operand와 여러 joint variable 사이의 matching 문제는 다음과 같이 해결하였다. 우선 base에서부터 gripper 쪽으로 나가면서 각 joint variable에 번호를 붙인다. 그리고 나서 명령어의 operand에 오는 정수가 N인 경우에는, N이라는 번호가 붙여진 joint variable을 assign한다. 그 사용예를 표 4에 보였다.

표 4. 사용 예
Table 4. Example.

1st joint variable····· 23.7
2nd joint variable·····130.0
3rd joint variable····· 13.3

$$*JRZ-1=\begin{bmatrix}\cos(-jv1) & -\sin(-jv1) & 0 & 0\\ \sin(-jv1) & \cos(-jv1) & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}0.916 & 0.402 & 0 & 0\\ -0.402 & 0.916 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$*JTY-2=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & -jv2\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & -130\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$*JRY-3=\begin{bmatrix}\cos(-jv3) & 0 & \sin(-jv3) & 0\\ 0 & 1 & 0 & 0\\ -\sin(-jv3) & 0 & \cos(-jv3) & 0\\ 0 & 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}0.973 & 0 & -0.230 & 0\\ 0 & 1 & 0 & 0\\ 0.230 & 0 & 0.973 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$*RRX\ 23.JRY-3.TTY\ 30=\begin{bmatrix}0.973 & 0 & -0.230 & 0\\ -0.090 & 0.921 & -0.380 & 27.63\\ 0.212 & 0.391 & 0.896 & 11.73\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Operand로 오는 정수 N은 양수뿐만 아니라 음수일 수도 있도록 하였다. 음수일 때는 joint variable의 부호를 바꾸어 assign하도록 하였기 때문에 양수일 때와는 그 움직임의 방향이 반대가 된다.

또한, 여러 개의 링크로 구성된 로보트의 움직임을 능률적으로 계산하기 위하여 다음과 같은 사항이 고려되었다. 우선 각 링크에 대한 연산처리 과정을 보면, 여러 명령어들 중에는, joint variable이 변화함에 따라 매번 새로이 계산하여야 하는 부분이 있는 반면, 처음에 한번만 계산해 주면 그 결과를 그대로 사용할 수 있는 부분도 있다. 또한, 여러 링크가 연결되어 있는 상황에서는 앞의 링크의 움직임의 영향을 고려한, 뒷 링크의 움직임을 계산할 수 있어야 한다.

이상의 고려사항을 다음과 같은 방법을 사용하여 조치하였다.

우선 연산의 능률을 높이기 위하여 명령어들을 쓰임새에 따라서 setup descriptor와 joint descriptor라는 이름의 두 그룹으로 나누되, 매번 계산해야 하는 명령어들은 joint descriptor에, 처음에 한번만 계산하면 되는 명령어들은 setup descriptor에 포함시킴으로써, 매번 계산하는 명령어의 수를 줄이도록 하였다.

또한 joint descriptor에 속하는 명령어들의 계산때는, 이전 링크까지 계산된 결과를 앞에 덧붙여서 계산하도록 하여서, 앞의 링크가 움직이는 영향을 뒤의 링크들에게 능률적으로 전파할 수 있도록 하였다.[11]

Ⅲ. 로보트 설계

앞에서 제시한 방법들을 사용하여 원하는 로보트를 설계하는 과정을 논한다. 우선 두 개의 링크로 구성되어 있는 간단한 형태의 로보트를 설계하는 경우를 대상으로 하여 그 과정을 논하고, 이를 잘 알려진 복잡한 형태의 산업용 로보트에 적용하는 예를 보인다.

1. 로보트 설계 과정

본 절에서는 두 개의 링크로 구성되는 간단한 로보트를, 앞에서 제안한 방법을 사용하여 설계하는 과정을 설명한다. 설계하려는 로보트의 형태는 그림 5와 같다.

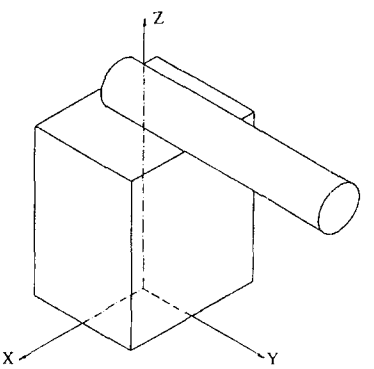
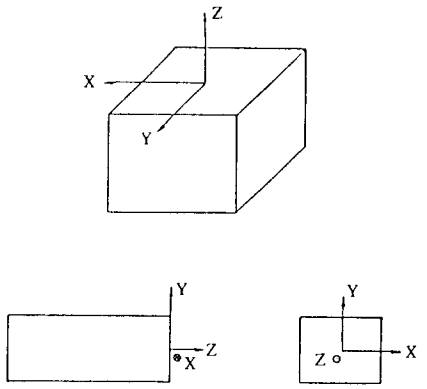


그림 5. 두 링크로 구성된 로보트 팔
Fig. 5. Two link robot arm.

로보트를 설계하는 첫 과정은 각 링크를 모델링하는 일이다. 그림 5에서 알 수 있듯이, 설계하려는 로보트는 四角柱 형태의 링크 1과 圓柱 형태의 링크2로 구성되어 있다. 링크 1의 경우에는 좌표축을 그림 6 (a)와 같이 설정해 놓고 四角柱로 모델링할 수 있다. 그 결과는 그림 6 (b)와 같다. 링크 2의 경우에는 좌표축을 그림 7 (a)와 같이 설정한다. 이에 따라 모델링한 결과는 그림 7 (c)와 같다.

링크를 모델링할 때 모든 데이타를 사용자 자신이 계산하여 입력하는 것이 가장 융통성있고 자유로운 방법이다. 그러나 일정한 형태의 링크들을 모델링하는 경우, 단순한 계산을 반복해야 하는 불편함을 해소하기 위해, 많이 쓰이는 형태의 링크를 모델링하는 경우에는 그 특징을 기술할 수 있는 몇개의 parameter들을 입력시킴으로써 모델링이 완료되게 하였다.

설계의 두번째 과정은 링크 collection에 해당한다. 모델링된 데이타들은 데이타 베이스에 축적되어 있으므로 이들 중에서 어떤 것을 선택하여 로보트를 구성

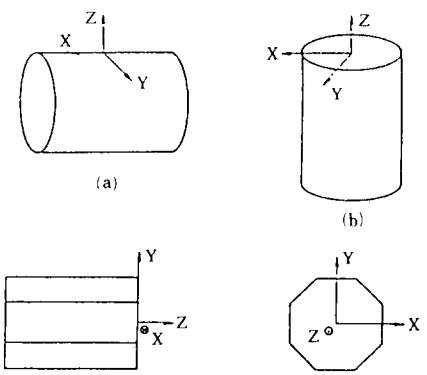


(a)

50	−40	0
50	40	0
−50	40	0
−50	−40	0
50	−40	−500
50	40	−500
−50	40	−500
−50	−40	−500

(b)

그림 6. Two link arm의 첫번째 링크
Fig. 6. Link 1 of the two link arm.



(a) (b)

(c)

5.74	−13.8	0	5.74	−13.8	−300
13.8	− 5.74	0	13.8	− 5.74	−300
13.8	5.74	0	13.8	5.74	−300
5.74	13.8	0	5.74	13.8	−300
− 5.74	13.8	0	− 5.74	13.8	−300
−13.8	5.74	0	−13.8	5.74	−300
−13.8	− 5.74	0	−13.8	− 5.74	−300
− 5.74	−13.8	0	− 5.74	−13.8	−300

(d)

그림 7. Two link arm의 두번째 링크
Fig. 7. Link 2 of the two link arm.

하겠다는 것을 지정한다.

그림6, 그림7과 같이 모델링된 링크들을 사용하여 그림5와 같은 two link arm을 설계하려는 경우에는, 문자열을 써서 링크1과 링크2에 대한 데이타 파일의 이름을 지정함으로써 원하는 링크를 선택하도록 하였다. 지정하는 순서는 편의상 base 쪽의 링크부터 먼저 지정하는 것을 원칙으로 하였다.

Two link arm을 위해 두개의 링크, 링크1과 링크2를 선택하고 나면 그림8과 같은 자료구조가 형성된다.

이상의 두 과정을 통해서, 그림6과 그림7처럼 모델링된 두 링크를 갖고 two link arm을 구성한다는 것까지는 기술이 되었으나, 두 링크가 어떻게 결합되는지, joint variable에 따라서 어떤 운동을 하는지에 대한 정보는 아직 언급되지 않은 상태이다.

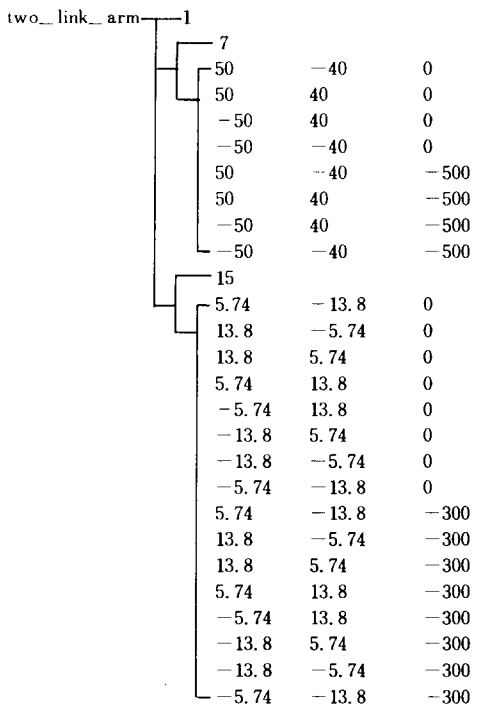


그림 8. Two link arm에 대한 자료구조
Fig. 8. Data structure of two link arm.

이제, 세번째 과정을 통해서 각 링크의 결합관계를 기술한다. 그림5와 같은 구조를 갖기 위해서는 각 링크에 표5와 같은 연산을 해 주면 된다. 이 연산을 능률적으로 처리하기 위해 Ⅱ-3절에서 언급한 방법을 사용한 결과는 표6과 같다.

2. 적용 예

앞 절에서 간단한 형태의 로보트를 예로 하며 설명

표 5. Two link arm을 기술하기 위한 명령어 群
Table 5. Commands for the desired two link arm.

link 1	TTZ 500
link 2	RRX −90 TTY 270 TTZ 513.8 JRZ 1

표 6. 표 5의 개량형
Table 6. Improved version of table 5.

	Joint D.	Setup D.
link 1	TTX 0.0	TTZ 500
link 2	JRZ 1	RRX −90 TTY 270 TTZ 513.8

한 설계과정을, 널리 알려진 산업용 로보트에 적용한 결과를 보인다.[11]

대상 로보트로는 R-R-R, R-R-P type의 로보트를 선택하였다. R-R-R type을 위해서는 Puma 560을 기초로 삼았고, R-R-P type을 위해서는 Stanford arm을 기초로 삼았다.[11]

적용결과는 그림 9와 그림10에 보였다. 각 화면 좌측 상단의 숫자 및 기호는 각 joint variable 및 hand의 상태를 표시한다. 또한 화면 우측 상단의 x, y, z 축 표시는 universal coordinate frame을 표시하는 것으로서 현재의 관찰자 위치를 알려 주는 역할을 하고 있다.

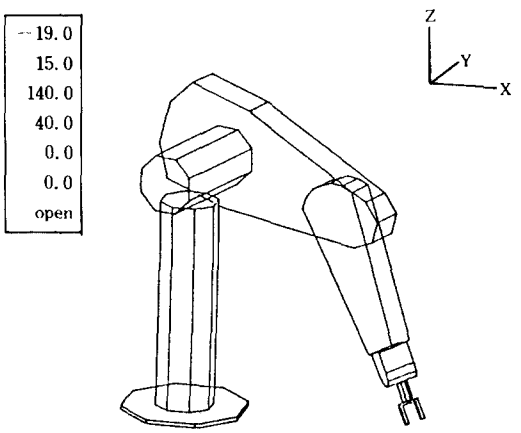


그림 9. Puma 560 arm에 대한 결과
Fig. 9. Approximated Puma 560 arm.

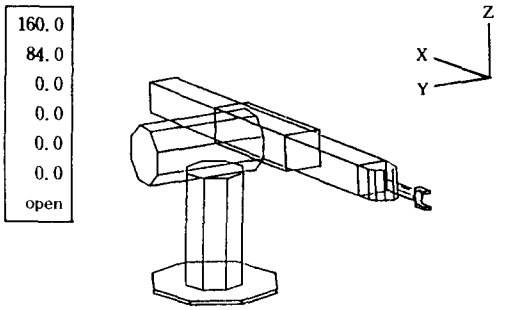


그림10. Stanford arm에 대한 결과
Fig. 10. Approximated Stanford arm.

Ⅳ. 결 론

본 논문에서는 로보트의 링크들을 多角柱 형태의 primitive들을 써서 모델링하는 방법을 제안하고, 링크 상호간의 결합관계를 기술할 수 있는 명령어들을 정의하였다. 그리고 몇가지 예를 통해서, 제안된 방법이 일반적 구조를 갖는 로보트의 설계에 사용될 수 있음을 보였다.

제안된 방법을 사용하면 일반적인 로보트의 구조를 통일된 형태의 자료구조를 써서 기술할 수 있게 된다. 이는 앞으로 로보트의 그래픽 시뮬레이터 등 로보트에 관계되는 각종 소프트웨어를 개발시에, 하나의 자료구조에만 맞추면 손쉽게 모든 형태의 로보트에 적용할 수 있게 된다는 장점을 갖는다.

參 考 文 獻

[1] Rita R. Schreiber, "How to teach a robot", *Robotics Today*, Jun. 1984.

[2] C.C. Thomson, "Robot modelling-the tools needed for optimal design and utilization.", *Computer-Aided Design*, vol.16, no.6, Nov. 1984.

[3] Shunji Mohri, Takashi Kogawa, "Robot language", *Hitachi Review*, vol.34, no.1, 1985.

[4] Masaharu TAKANO, "Development of simulation system of robot motion and its role in task planning and design systems", *Robotics Research, The 2nd International Symposium*, MIT Press, 1985.

[5] Tamio Arai, "A robot language system with a color graphic simulator", *Advanced Software in Robotics*, Elsevier Science Publishers, 1984.

[6] Kunwoo Lee, "A CAD system for designing robotic manipulators, *IEEE International Conference of Robotics and Automation*, 1985.

[7] Robert N. Stauffer, "Robot system simulation", *Robotics Today*, Jun 1984.

[8] H.E. Bez, "Homogeneous coordinates for computer graphics", *Computer-Aided Design*, vol.15, no.6, Nov. 1983.

[9] R.F. Reisenfeld, "Homogeneous coordinates and projective planes in computer graphics", *IEEE*, CG&A, Jan. 1981.

[10] 장 원, 장 칠, "컴퓨터 그래픽스를 이용한 로보트 시뮬레이터의 설계", 대한전기학회, 계측제어 시스템 연구회 제22회 학술발표회, 1985.

[11] 장 원, "컴퓨터 그래픽스를 이용한 로보트 시뮬레이터의 설계", 한국과학기술원, 석사학위논문, 1986.

[12] J. Denavit and R.S. Hatenberg, "A kinematic notation for lower-pair mechanisms based on matrices", *Journal of Applied Mechanics*, pp.215-221, 1955.

[13] C.S.G. Lee, "Robot arm kinematics", Tutorial on Robotics, *IEEE Computer Society Press*, 1983.