論文 88-25-1-2

로보트와 제어기의 개발을 위한 로보트 시뮬레이터의 설계

(Design of a Robot Simulator for the Development Robot and its Controller)

張 哲,* 張 源,* 鄭明振,* 卞增男*

(Cheol Chang, Won Jang, Myung Jin Chung and Zeungnam Bien)

要 約

본 논문은 로보트의 기하학적인 모델링을 효과적으로 할 수 있고, 로보트 및 구동기의 동적특성들을 쉽게 이해할 수 있으며 여러가지 제어방식들의 성능을 평가할 수 있는 로보트 시뮬레이터에 관하여 논하였다. 이를 위하여 기구학, 동력학 및 삼차원 공간상에서 로보트의 움직임을 시각적으로 보기 위하여 컴퓨터 그래픽스에 관한 연구가 수행되었다. 개발된 로보트 시뮬레이터는 다이네믹 시뮬레이터와 그래픽 시뮬레이터 두가지로 구성이 된다. 전체시스템의 평가를 위하여 PUMA-560, Stanford arm, RHINO 로보트가 고려되었고, MV/10000 수퍼 미니 컴퓨터와 IBM-PC/XT 개인용 컴퓨터가 사용되었다.

Abstract

This paper describes a robot simulator which enables a user to model a robot geometrically, and to evaluate performances of various robot control algorithms as well as to obtain physical understanding of robot and actuator dynamics. To achieve these goals, the kinematics and dynamics of a robot and interactive 3-D computer graphics which visulaize the motion of the robot were studied. The developed robot simulator consists of two parts:a dynamic simulator and a graphic simulator. To evaluate the robot simulator PUMA-560, Stanford arm, and RHINO robot were considered and a DG MV/10000 super mini-computer and an IBM-PC/XT personal computer were used.

I. 서 론

산업의 규모가 커지고 다양하게 발달함에 따라 작업환경이 수시로 변화하게 되고, 또한 사람이 직접 일하기에는 위험한 경우가 많이 생기게 되었다. 이에 따라 새로운 작업요구에 신속히 대응하기 위하여 필요한 로보트를 체계적으로 설계할 수 있고, 로보트의 교시, 작업 배치 등을 위해 소모되는 시간과 비용을 줄일 수 있으며, 또한 작업자의 안전을 도모할 수 있는 방법들에 대한 연구가 필요하게 되었다.

그러나 종래의 경험적인 설계방식을 사용할 경우 설계 및 생산에 소요되는 시간과 비용이 많이 들게 되므로 사용자들의 다양한 요구조건을 만족시키기가 힘들게 된다. 그리고, 산업체에서 일반적으로 사용하는 플레이-백 방식으로는 로보트의 교시에 소요되는 시간을 줄이기 어려울 뿐더러, 작업자의 안전을 보장하기가 힘들다. 이러한 문제점들을 해결하기 위하여 많은 연구

*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 9月 22日

가 진행되어 왔으나, 그중 컴퓨터를 이용한 로보트 시뮬레이션 방식이 크게 관심을 모으고 있다.[1~6]

위에 언급된 방식을 이용하여 로보트와 제어기의 설계 및 성능을 정확히 평가하기 위해서는 로보트의 정확한 모델식들을 사용하여야 하며, 로보트의 교시 및 작업배치 등을 위하여는 컴퓨터 그래픽스를 이용하여 로보트와 주위 작업환경을 시각적으로 나타낼 필요가 있다. 본 논문에서는 이를 위하여 로보트의 기하학적, 역학적 면들을 고려한 로보트 시뮬레이터의 개발에 촛점을 두고 있으며, 이는 로보트의 역학적 특성 및 제어기의 반응 등을 알아보는 다이네믹 시뮬레이터(dynamic simulator)와 로보트의 기하학적 모델링 기능과 다이네믹 시뮬레이터에서 얻어진 데이타를 가지고 로보트의 움직임을 시각적으로 보여줌으로써 삼차원 공간에서의 궤적등을 쉽게 볼 수 있도록 하는 기능을 갖는 그래픽 시뮬레이터(graphic simulator)로 나누어진다(그림 1 참조).

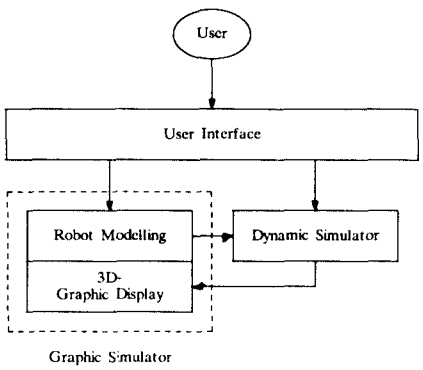

그림 1. 로보트 시뮬레이터의 구성
Fig. 1. The structure of robot simulator.

로보트 시뮬레이터의 개발시 특히 고려하여야 할 점은 사용자가 사용하기에 편한, 즉 사용자 위주의 시스템이어야 하는데, 지금까지 나온 많은 로보트 시뮬레이션 시스템들이 널리 사용되고 있지 못한 이유중의 하나가 바로 이러한 점의 충분한 고려가 되어있지 않기 때문이다. 그리고 로보트 시뮬레이터가 범용으로 사용되어지기 위해서는, 로보트와 관련된 여러 기능들이 독자적으로, 혹은 서로 조화를 이루며 운영되어져야 하기 때문에 로보트 자체뿐만 아니라 데이타 구조, 프로그램 구조 및 파일 조작(file operation) 등이 효율적으로 구성되어야 한다.

개발된 로보트 시뮬레이터에서 많은 계산량이 요구되어지는 다이네믹 시뮬레이터는 MV/10000 컴퓨터에 구현되었고, 그래픽 시뮬레이터부분은 IBM-PC/XT에 구현하였다. 사용되어진 프로그래밍 언어는 Pascal과 8088 assembly 언어이다.

Ⅱ. 다이네믹 시뮬레이터

다이네믹 시뮬레이터는 로보트의 기구학적, 동력학적 특성을 시뮬레이션하는 부분으로, 설계되어진 로보트의 동작이 만족할 만한지, 그리고 어떠한 구조의 제어기가 적당하며, 그 제어기의 파라미터 값들은 어떻게 정해야 하는지 등을 시뮬레이션해 봄으로써 원하는 로보트에 대한 설계 파라미터들을 쉽게 얻을 수 있도록 하는 기능을 갖는다.

먼저 기구학에 관하여 논하고, 로보트의 기계부분 및 구동기로 많이 사용되고 있는 DC 모터의 다이네믹스에 대해 알아보고, 이들의 시뮬레이션 과정에 대해 논하기로 한다.

1. 로보트의 기구학 및 역기구학

로보트 링크들 간의 상호관계는 그림 2 와같이 Denavit-Hartenberg표현 방식에 의해 4×4 homogeneous transform matrix로 표현될 수 있다.[12]

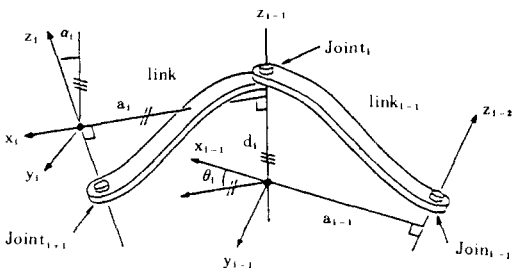

그림 2. 링크 좌표계와 파라미터들
Fig. 2. Link coordinate system and parameters.

위에서 정의된 조인트, 링크 파라미터들을 이용하여 그림 2 에서 (i-1)번째 링크 좌표계에서 본 i번째 링크 좌표계의 상대적 위치를 나타내려면 2 번의 translation과 2 번의 rotation에 의하여 다음과 같이 표현된다.

$$T_{i-1}^{i} = \mathrm{Rot}(z, \theta_i) \cdot \mathrm{Trans}(z, d_i) \cdot \mathrm{Trans}(x, a_i) \cdot \mathrm{Rot}(x, \alpha_i)$$

$$= \begin{bmatrix} c\theta_i & -c\alpha_i \cdot s\theta_i & s\alpha_i \cdot s\theta_i & a_i \cdot c\theta_i \\ s\theta_i & c\alpha_i \cdot c\theta_i & -s\alpha_i \cdot c\theta_i & a_i \cdot s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

여기서

$c\theta_i=\cos\theta_i$, $s\theta_i=\sin\theta_i$, $c\alpha_i=\cos\alpha_i$, $s\alpha_i=\sin\alpha_i$

로보트의 자유도가 6 이라 가정할 때 로보트손의 위치 및 방향은 T_0^1부터 T_5^6까지의 곱으로 표현되어진다. 즉, $T_H=T_0^1\cdot T_1^2\cdot T_2^3\cdot T_3^4\cdot T_4^5\cdot T_5^6$

$$=\begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2)$$

식(2)에서 로보트손의 좌표계와 방향을 나타내는 세 벡터 $\underline{n}$, $\underline{s}$, $\underline{a}$는 그림 3 과 같이 주어진다.

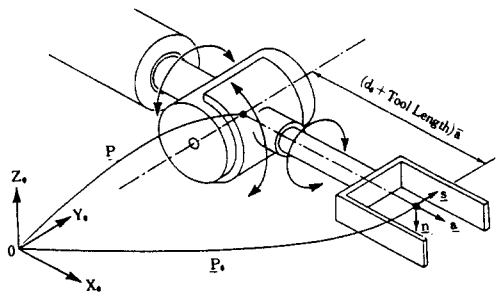

그림 3. 로보트손의 좌표계
Fig. 3. Hand coordinate system.

로보트의 기구학을 직접 푸는 경우에 대한 해는 식(1)과 (2)와 같이 일반적으로 표현될 수 있으나, 역기구학 문제는 로보트의 구조에 따라 기하학적,[12] 대수적,[17] 반복적[13] 방법중에서 특정한 방법을 사용하여 해를 구하는 것이 보통이다. 본 논문에서는 PUMA-560, Stanford 로보트들과 같은 기하학적 구조가 복잡하지 않은 경우에 쉽게 해를 구할 수 있는 기하학적 방법을 사용하였으며, 로보트의 자세를 나타내기 위해 다음과 같이 정의된 4 개의 자세 지시기들을 사용하였다.

$$\text{ARM} = \begin{cases} 1 : \text{right arm} \\ -1 : \text{left arm} \end{cases} \qquad (3)$$

$$\text{ELBOW} = \begin{cases} 1 : \text{above arm} \\ -1 : \text{below arm} \end{cases} \qquad (4)$$

$$\text{WRIST} = \begin{cases} 1 : \text{wrist down} \\ -1 : \text{wrist up} \\ 0 : \text{if in the degenerate case} \end{cases} \qquad (5)$$

$$\Omega \begin{cases} = \dfrac{\underline{s}\cdot(\underline{z}_3\times\underline{a})}{\|\underline{z}_3\times\underline{a}\|} : \text{if } \underline{s}\times(\underline{z}_3\times\underline{a})=0 \\ \dfrac{\underline{n}\cdot(\underline{z}_3\times\underline{a})}{\|\underline{z}_3\times\underline{a}\|} : \text{otherwise} \end{cases} \qquad (6)$$

앞에 정의된 자세 지시기들중 Ω는 로보트손의 현재 방향상태를 나타내고, WRIST는 다음 원하는 손목의 방향상태를 나타내며, Stanford arm과 같은 구조를 갖는 로보트의 경우에는 ELBOW 지시기가 필요없게 된다. 역기구학을 기하학적인 방법을 사용하여 푸는 과정을 간단하게 나타내면 그림 4 와 같다.

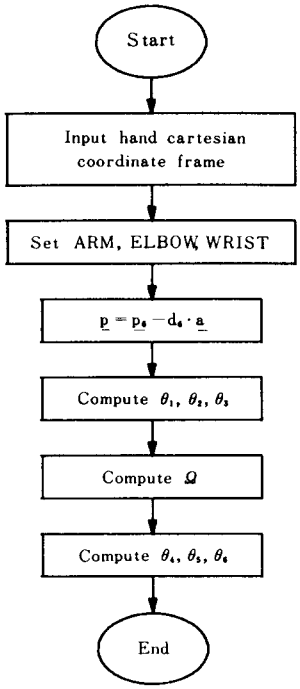

그림 4. 역기구학 해를 구하는 순서도
Fig. 4. The flow chart solving inverse kinematics.

2. 로보트 및 구동기의 동력학

로보트의 동력학은 기계부분과 구동기부분으로 구분되며, 대개의 경우 구동기의 동력학을 기계부분에 포함시켜 각 조인트의 입력 토크에 대한 로보트의 응답을 다루나, 제어기 및 구동기의 성능을 제대로 평가하기 위해서는 구동기의 동력학을 분리하여 다룰 수 있도록 하고 또한 구동기의 입력전류 혹은 전압에 대한 로보트의 다이네믹스를 구하여야 한다. 입력 토크에 대한 로보트의 다이네믹스는 Lagrange-Euler식을 이용하면 식(7)과 같이 나타낼 수 있고,[12] 각 조인트에서의 DC모터의 식은 식(8)과 같이 나타낼 수 있다.

$$G(\underline{q})\cdot\ddot{\underline{q}}+H(\underline{q},\dot{\underline{q}})+G(\dot{\underline{q}})+F_m\cdot\dot{\underline{q}}=\underline{\tau} \qquad (7)$$

여기서

$\underline{q}$: N× 1 generalized joint vector.
$D(\underline{q})$: N×N intertial acceleration-related symmetric matrix.
$H(\underline{q},\dot{\underline{q}})$: N× 1 nonlinear Coriolis and Centrifugal force vector.

$G(q)$: N×1 gravity loading force vector.

F_m : N×N diagonal friction coefficients matrix.

τ : N×1 applied force/torque vetor to joints.

$$d_{ai}+f_{ai}\cdot\dot{q}_i+\tau_i/n_{ai}=k_{ai}\cdot u_i \quad \text{at joint } i \tag{8}$$

여기서

d_{ai} : inertia coefficient

f_{ai} : damping coefficient

n_{ai} : gear ratio

k_{ai} : torque gain

u_i : input current

식(7)과 (8)을 이용하여 입력전류와 로보트의 조인트 변수값들 사이의 관계식은 식(9)와 같이 주어진다.

$$[D(q)+J_a]\cdot\ddot{q}+H(q,\dot{q})+[F_m+F_a]\cdot\dot{q}+G(q)$$
$$=J(q)\cdot\ddot{q}+H(q,\dot{q})+F\cdot\dot{q}+G(q)=K_a\cdot U \tag{9}$$

여기서

$$J_a=\begin{bmatrix} d_{a1}\cdot n_{a1}^2 & & & & & \\ & d_{a2}\cdot n_{a2}^2 & & & & \\ & & d_{a3}\cdot n_{a3}^2 & & & \\ & & & d_{a4}\cdot n_{a4}^2 & & \\ & & & & d_{a5}\cdot n_{a5}^2 & \\ & & & & & d_{a6}\cdot n_{a6}^2 \end{bmatrix}$$

$$F_a=\begin{bmatrix} f_{a1}\cdot n_{a1}^2 & & & & & \\ & f_{a2}\cdot n_{a2}^2 & & & & \\ & & f_{a3}\cdot n_{a3}^2 & & & \\ & & & f_{a4}\cdot n_{a4}^2 & & \\ & & & & f_{a5}\cdot n_{a5}^2 & \\ & & & & & f_{a6}\cdot n_{a6}^2 \end{bmatrix}$$

$$K_a=\begin{bmatrix} k_{a1}\cdot n_{a1}^2 & & & & & \\ & k_{a2}\cdot n_{a2}^2 & & & & \\ & & k_{a3}\cdot n_{a3}^2 & & & \\ & & & k_{a4}\cdot n_{a4}^2 & & \\ & & & & k_{a5}\cdot n_{a5}^2 & \\ & & & & & k_{a6}\cdot n_{a6}^2 \end{bmatrix}$$

식(9)는 각 조인트의 구동기에 입력 전류가 주어졌을 때 조인트 변수들의 값을 구하는데 사용되어진다. 역으로 조인트 변수들이 주어졌을 때 필요한 입력전류를 구하기 위해서는 Lagrange-Euler방식보다 상대적으로 계산량이 적은 Netwton-Euler방식을 사용하는 것이 바람직하다.[12] 구동기의 동력학 특성을 고려한 Newton-Euler식은 Lagrange-Euler식을 유도할 때와 마찬가지로 식(8)을 사용하면 쉽게 구할 수 있으므로 자세한 유도과정은 생략하기로 한다. 로보트에 입력전류가 인가되었을 때 조인트 변수들의 값을 구하기 위하여 4 차 Runge-Kutta-Gill 방법을 사용하였다. 이를 위하여 식(9)를 다음과 같이 상태 방정식으로 나타내었다.

$$x_i = q_i \tag{10a}$$

$$x_{i+6}=\dot{q}_i \tag{10b}$$

이때 $i=1,\cdots,6$

$$\dot{x}_i = x_{i+6} \tag{11a}$$

$$\dot{x}_{i+6}=f_i(\underline{x}) \tag{11b}$$

여기서

$\underline{x}=[x_1, x_2, \cdots, x_{12}]^T$

$f_i(\underline{x})$: i^{th} component of $[-J^{-1}\cdot(H+G+F\cdot\dot{q}-K_a\cdot U)]$

식(10)과 (11)을 사용하고 Runge-Kutta-Gill 방법을 이용하여 로보트의 다이네믹스를 시뮬레이션하는 과정을 그림 5 에 나타내었다.

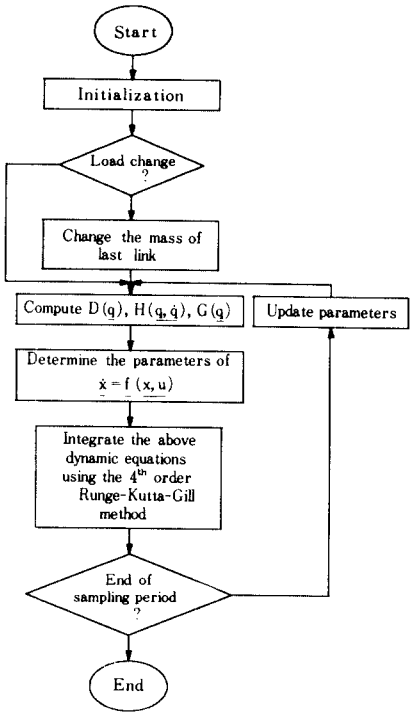

그림 5. 다이네믹스 시뮬레이션 순서도
Fig. 5. The flow chart of dynamics simulation.

Ⅲ. 그래픽 시뮬레이터

개발된 그래픽 시뮬레이터의 기능은 크게 두가지로 나누어 지는데, 그중 하나는 로보트의 기하학적인 모델링에 관한 것이고, 다른 하나는 다이네믹 시뮬레이터에서 얻어진 동작 데이타를 받아서 원하는 여러시각 각도에서의 동작모습을 3 차원적으로 그래픽 티미널에 디스플레이 하는 기능이다.

1. 로보트의 모델링

모델링 기능에서는 로보트를 여러 링크들이 모여서 구성되는 결합체로 보고 각 링크들을 다각주로 근사화하여(그림6참조), 이들 근사화된 링크들을 모아 하나의 로보트을 기술하기 위하여 homogeneous transfor-

mation과 관련된 명령어들을 정의하고(표 1 참조), 이들을 사용하여 복잡한 결합구조를 갖는 각종 로보트에 대하여 그 결합관계를 손쉽게 표현할 수 있도록 하였다.[8,10]

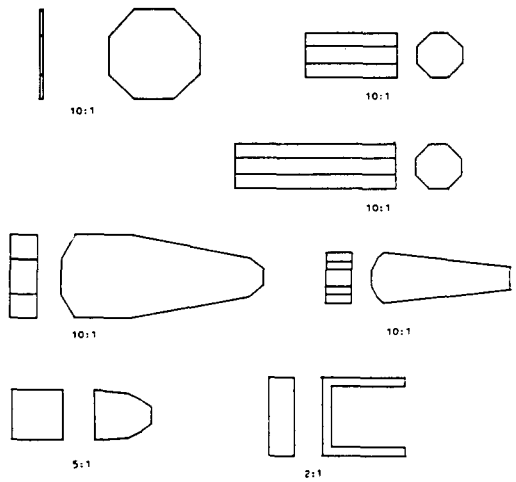

그림 6. 모델링된 여러 링크들의 예
Fig. 6. Some of modelled links.

표 1. 정의된 명령어들
Table 1. Defined commands.

Command	Operation
RRX op	Rotation op〔deg〕about x axis
RRY op	Rotation op〔deg〕about y axis
RRZ op	Rotation op〔deg〕about z axis
TTX op	Translation op〔mm〕along x axis
TTY op	Translation op〔mm〕along y axis
TTZ op	Translation op〔mm〕along z axis
SSX op	Scaling op along x axis
SSY op	Scaling op along y axis
SSZ op	Scaling op along z axis

2. 그래픽 디스플레이

그래픽 디스플레이 기능에는 관찰자 좌표계의 변환, 링크에서의 숨은선 제거, 원근효과, 계산된 좌표값들이 화면에서 그릴 수 있는 범위를 넘어서는 경우 이들을 처리하는 디바이스 클리핑 및 물체가 관찰자의 뒤에 있는 경우 이를 처리하는 삼차원 클리핑 기능등이 있다.

관찰자 좌표계의 변환은 관찰자의 위치가 변화됨으로써 나타나는 효과를 화면상에 나타내기 위한 것으로써, 공간상에 주어진 점이 화면상에 어떻게 나타날 것인가를 고려하는 기능이다. 이를 위해서 공간상에 기준이 되는 좌표계(universal coordinate system)를 기준으로 표시된 점을 관찰자의 눈을 기준으로 하는 좌표계(eye coordinate system)으로 변환시켜 주어야 한다. 본 논문에서는 좌표계의 변환을 위하여 4×4 homogeneous transformation matrix를 사용하였다.

링크에서의 숨은선 제거는 각 링크를 구성하는 면이 모두 평면이고, 그 링크들의 꼭지점들을 안다는 점에 착안하여 다음과 같은 방법을 사용하였다.

(a) 평면상의 서로 다른 세 점의 좌표들로부터 평면상의 두 vector를 구한다.

(b) 두 vector의 cross product를 구하여 평면의 normal vector를 구한다.

(c) Normal vector를 사용하여 평면이 관찰자를 향하고 있는가를 판단한다.

이상의 (a), (b), (c)들을 링크를 구성하는 모든 면에 적용하여, 앞면에 속한 가장자리선들은 화면에 그리고, 뒷면에 해당되는 선들은 제외함으로써 링크단위의 숨은선 제거를 할 수 있다.

Real world에서 나타나는 원근효과로는 다음과 같은 것들을 들을 수 있다.

(a) 평행선들이 지평선에서 만나는 것처럼 보인다.

(b) 똑같은 물체라도 멀리 있는 것이 작게 보인다.

컴퓨터 그래픽스를 이용하여 위와 같은 원근효과를 내기 위해서는 그림 7 에서 보듯이 관찰자 E와 공간상의 점P를 잇는 직선이 평면 PL과 만나는 점P′에 P의 영상을 디스플레이 하여야 한다.

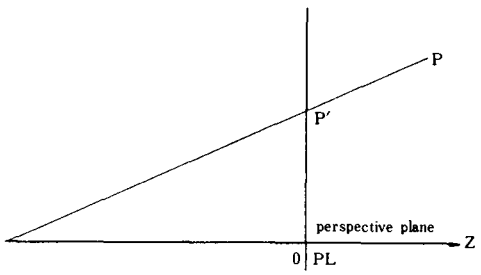

그림 7. Perspective plane으로의 투영
Fig. 7. The projection of perspective plane.

디바이스 클리핑은 real world의 점들을 관찰자의 좌표계로 변환시켜 화면상에 나타낼 때 화면상에 실제로 그릴 수 없는 범위의 값을 가지는 경우에 이러한 점들을 그리지 않도록 처리하는 기능이다. 시스템에 따라서는 이러한 경우를 처리해 주지 않는 경우 오동

작을 일으키는 원인이 됨으로 사전에 처리해 놓아야 한다. 이를 수행하는 알고리즘은, 우선 한직선의 두끝점이 모두 화면 내부에 있는지를 판단하고, 그렇지 않을 경우 화면의 테두리와 만나는 점들을 계산하여 화면 내부에 해당하는 부분만 그리도록 하는 것이다. 이와 같은 방식으로 클리핑하는 알고리즘들이 상당수 발표되었는데, 본 논문에서는 Cohen-Sutherland clipping algorithm을 사용하였다.[16]

삼차원 클리핑을 하기 위해서는 6 개의 평면으로 구성되는 view volume[15]을 정의하여 직선과 평면들과의 위치관계를 이들 6 개의 평면에 대해 계산해야 하고, 교점계산에 있어서도 직선과 평면사이의 점을 계산해야 하기때문에 요구되는 계산량이 상당히 많아지게 된다. 이러한 문제점을 개선하기 위해 view volume을 규정하는 평면의 갯수를 하나로 줄이는 새로운 방법을 사용하여 관찰자의 눈뒤에 있는 부분을 지운 다음, 기존 디바이스 클리핑 방법을 적용하여 화면 밖으로 벗어나는 부분을 처리하였다. 이 방식의 장점으로는 클리핑 대상을 6 개의 평면에서 하나의 평면으로 줄인 결과 계산량이 대폭 감소된다는 잇점이 있다.[2]

Ⅳ. 로보트 시뮬레이터의 구성

개발된 로보트 시뮬레이터의 구성을 그림 8 에 나타내었다. 그 기능들을 살펴보면, 우선 User interface는 사용자의 명령을 받아 원하는 작업을 수행할수 있도록 여러 부프로그램들을 제어하는 기능을 갖고 있다.

다이네믹 시뮬레이터는 크게 두가지 기능을 갖는데, 그중 하나는 로보트의 기구학 및 역기구학등을 이용하여 로보트의 궤도를 계획하는 궤도계획 기능으로

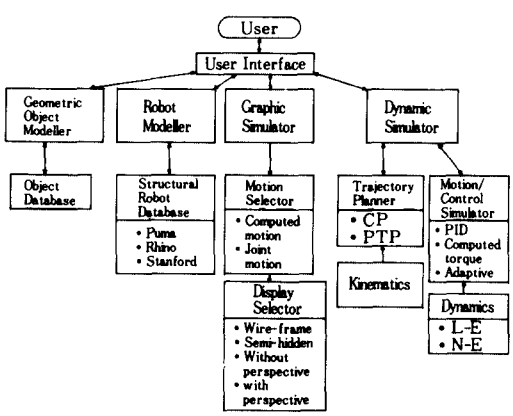

그림 8. 개발된 로보트 시뮬레이터의 구성
Fig. 8. The structure of the developed robot simulator.

써 현재는 직선운동 및 점대점(point to point) 운동에 대한 알고리즘이 구현되어 있다. 다른 하나는 로보트의 다이네믹스를 이용하여 로보트의 동작 및 제어방식들을 시뮬레이션할 수 있는 동작/제어 시뮬레이션(motion and control simulation) 기능이다. 구현되어진 다이네믹스에는 Largrange-Euler 및 Netwton-Euler 방식들이 있다. 그리고 제어방식에는 PID, computed torque 및 적응제어 방식들이 구현되어 있다.

Geometric object modeller는 로보트를 구성하는 링크, 그리퍼(gripper), 작업물들을 모델링하며, robot modeller에서는 모델링되어진 링크 및 그리퍼들을 결합하여 로보트의 구조를 완성한 다음 이들을 데이타베이스에 저장한다. 이와 같이 모델링 기능을 나눈 이유는 로보트의 구조중에서 일부만이 변할 때 불필요하게 처음부터 끝까지 다시 모델링하는 폐단을 없애기 위한 것이다. 즉, 기존 로보트의 구성중에서 수정이 필요한 부문만 고침으로써 전체적인 설계시간을 단축할 수 있는 잇점을 갖게된다. 현재 로보트들의 데이타 베이스로는 PUMA, Stanford, RHINO 등이 있다.

그래픽 시뮬레이터에서는 사용자가 설계된 로보트의 여러 자세들을 볼 수 있도록 쉽게 볼 수 있도록 keyboard상의 key들을 이용하여 로보트의 각 조인트를 임의로 움직여 볼 수 있는 joint motion mode와 다이네믹 시뮬레이터에서 계산 되어진 동작 데이타를 이용하여 연속적인 동작을 볼 수 있도록 하는 computed motion mode중에서 하나를 선택하는 motion selector가 있고, 로보트를 화면상에 디스플레이할때 어떠한 효과를 고려하여 디스플레이할 것인가를 선택하는 display selector가 있다. 디스플레이 모우드에는 숨은선 제거가 고려되지 않는 wire-frame mode, 숨은선 제거가 고려되어지는 semi-hide mode, 원근효과가 고려되지 않는 without perspective mode, 원근효과가 고려되어지는 with perspective mode 등이 있다. Wire-frame mode는 로보트의 자세한 모습보다는 애니메이션(animation) 시 속도를 빨리하기 위한 모우드이며, semi-hide with perspective mode는 애니메이션 속도는 늦어지나 로보트의 모습을 자세히 그릴 수 있다. 그러므로 사용자는 원하는 기능을 강조할 수 있는 모우드를 선택하여 사용할 수 있는 유연성을 갖게 된다.

사용 컴퓨터 시스템으로는 MV/10000 수퍼미니 컴퓨터, IBM-PC/XT 기종이며, 전체 하드웨어 구조는 그림 9 와 같다. 프로그래밍 언어로는 Pascal과 8088 assembly 언어를 사용하였다. 이는 CRT와 같은 각종 주변기기들을 제어하기에는 Pascal과 같은 고급언어로는 힘들기 때문에 이들에 관련된 기능들

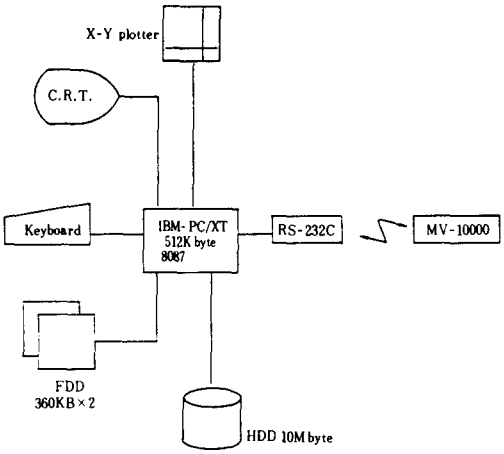

그림 9. 하드웨어 구조
Fig. 9. Hardware structure.

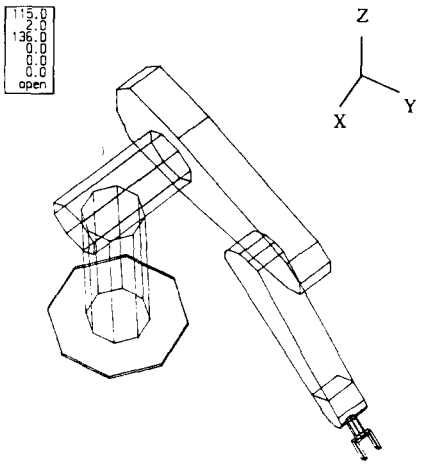

그림11. PUMA-560의 예
Fig. 11. An example of PUMA-560.

은 assembly 언어를 사용하여 구현하였다.

V. 적 용 예

개발된 로보트 시뮬레이터의 적용예로서 우선 그래픽 시뮬레이터의 각종 모우드에 해당하는 응용예들을 그림11에서 15까지에 나타내었고, 다이네믹 시뮬레이터의 응용예로서는 연구용으로 널리 사용되어지는 PUMA-560 로보트에 대하여 직선운동에 관한 시뮬레이션 결과들을 그림(16~20)에 보이고, 그래픽 시뮬레이터를 이용하여 다이네믹 시뮬레이터의 결과들에 대응하는 3차원상의 연속동작 모습을 그림21에 나타내었다. 사용되어진 로보트의 링크 좌표계들은 그림10과 같이 주어지고, 파라미터들의 값들은 표2와 3에 주어졌다. 표3에 주어진 값들은 임의로 정한 값들이다. 직선운동의 궤도계획은 본 논문에서는 시점과 종점 사이의 직선구간을 움직여 갈 때 출발과 정지시

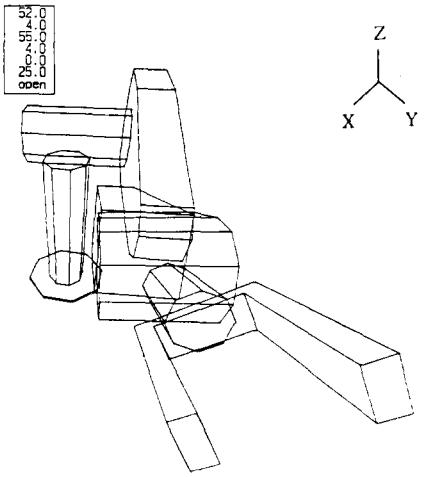

그림12. 원근효과가 고려된 PUMA-560
Fig. 12. PUMA-560 with perspective effect.

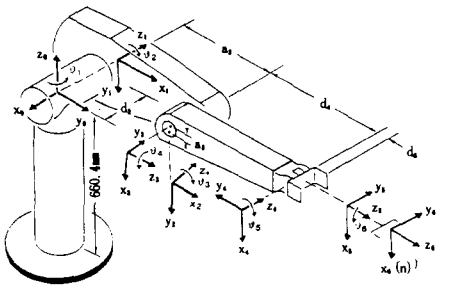

그림10. PUMA-560의 링크 좌표계
Fig. 10. Link coordinate system of PUMA-560.

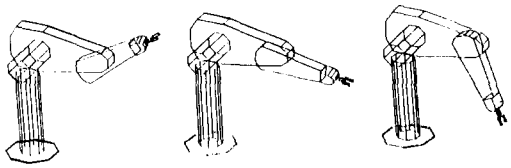

그림13. PUMA-560의 조인트 모션
Fig. 13. A joint motion of PUMA-560.

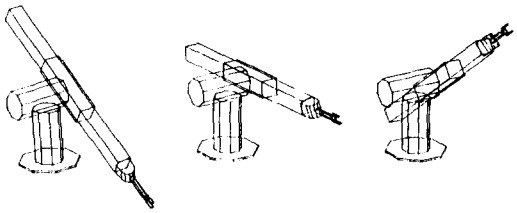

그림14. Stanford arm의 조인트모션
Fig. 14. A joint motion of stanford arm.

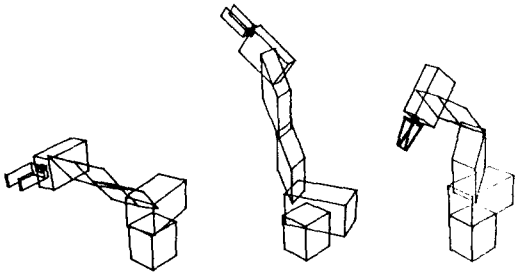

그림15. RHINO-XRIII의 조인트 모션
Fig. 15. A joint motion of RHINO-XRIII.

구동기에 무리가 가지 않도록 구동기가 낼 수 있는 최대토크 및 최대 토크변화율을 넘지 못하도록 하는 제한을 두어 가감속 구간을 구한 다음 이 구간들에 대해서 보간함수를 만드는 방식을 사용하였다.[7] 주어진 시점과 종점의 homogeneous transform matrix값들은 다음과 같다.

$$X_1 = \begin{bmatrix} -0.39 & -0.77 & 0.5 & -0.1 \\ 0.58 & 0.217 & 0.784 & 0.3 \\ -0.71 & 0.6 & 0.362 & 0.5 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \quad (12)$$

$$X_2 = \begin{bmatrix} -0.335 & -0.915 & 0.225 & 0.1 \\ 0.585 & -0.015 & 0.811 & 0.5 \\ -0.739 & 0.0403 & 0.54 & 0.1 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \quad (13)$$

정상상태에서 주어진 로보트손의 직교 좌표계에서의 등속도는 0.35m/sec이다. 그림16은 직교 좌표계에서 직선운동의 속도 프로파일이며 이에 대응하는 조인트 공간에서의 속도 프로파일을 그림17에 나타내었다.

Ⅵ. 결 론

로보트 시스템을 개발하는데 도움이 되는 로보트 시뮬레이터에 관하여 논하였다. 전체 시스템의 동작은

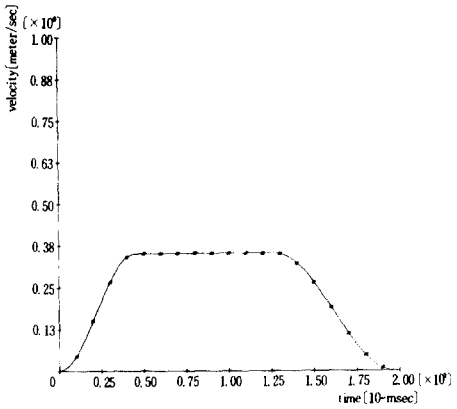

그림16. 직교좌표계에서의 속도 프로파일
Fig. 16. The velocity profile in cartesian space.

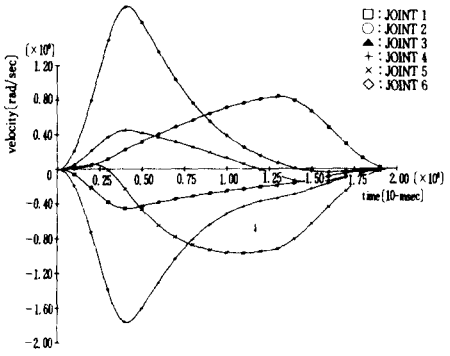

그림17. 조인트 공간에서의 속도 프로파일
Fig. 17. The velocity profile in joint space.

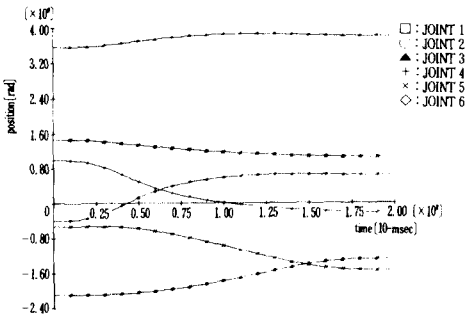

그림18. 조인트 공간에서의 궤도들
Fig. 18. The trajectories in joint space.

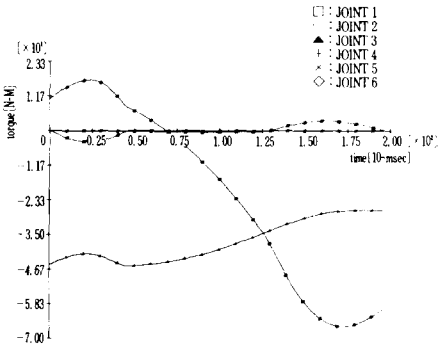

그림19. 각 조인트에 인가된 토크들
Fig. 19. The torques applied at each joint.

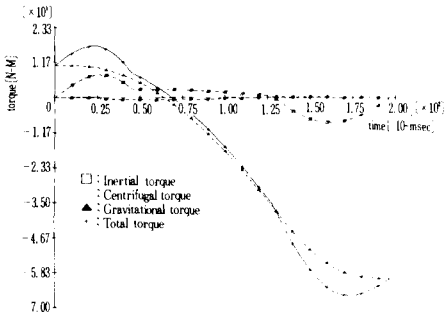

그림20. 조인트 2 에서 토크 컴퍼넌트들의 영향
Fig. 20. The contributions of the torque components at joint2.

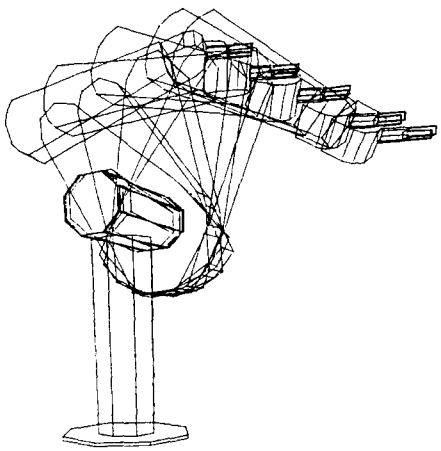

그림21. PUMA-560의 활동그림
Fig. 21. An animation of PUMA-560.

표 2. PUMA-560의 링크 파라미터
Table 2. Link coordinate parameters of a PUMA-560.

joint	θ_i [deg]	α_i [deg]	a_i [mm]	d_i [mm]
1	90	-90	0.0	0.0
2	0	0	431.8	149.09
3	90	90	-20.32	0.0
4	0	90	0.0	433.07
5	0	90	0.0	0.0
6	0	0	0.0	56.25

표 3. 링크질량과 무게중심
Table 3. Link mass and center of mass.

link	mass (kg)	x axis (m)	y axis (m)	z axis (m)
1	3.00	0.00	0.35	0.00
2	15.00	0.25	0.00	0.00
3	12.00	0.00	0.00	0.25
4	1.08	0.00	0.01	0.00
5	0.63	0.00	0.00	0.00
6	0.51	0.00	0.00	0.15

off-line 형식으로 다이네믹 시뮬레이터에서 필요한 사항에 대해서 시뮬레이션해 본 다음, 그 결과 데이타를 그래픽 시뮬레이터로 전송하여 CRT에 디스플레이 하도록 되어 있다. 결과적으로 지금까지 경험적으로 취급하던 로보트의 설계 및 성능 평가를 체계적으로 할 수 있게 되었다. 앞으로 구동기의 정확한 사양에 따른 정확한 시뮬레이션과 또한 로보트 언어의 개발, 작업환경에 관한 데이타 베이스를 구성방법과 실시간 애니메이션 효과, off-line programming system에 대한 연구가 되어져야 할 것으로 생각된다.

參 考 文 獻

[1] M. 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.

[2] M.S. Pfeifer, C.P. Neuman, "An adaptive simulator for robot arm dynamics," *CIME*, 1984.

[3] R.N. Stauffer, "Robot system simulation", *Robotics Today*, Jun. 1984.

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

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

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

[7] 장철, "로보트 팔과 제어기 설계를 위한 다이네믹 시뮬레이터의 설계," 한국과학기술원, 석사학위논문, 1986.

[8] 장원, "컴퓨터 그래픽스를 이용한 로보트 시뮬레이터의 설계," 한국과학기술원, 석사학위논문, 1986.

[9] 장원, 장철, "컴퓨터 그래픽스를 이용한 로보트 시뮬레이터의 설계," 대한전기학회, 계측제어시스템 연구회 제22회 학술발표회, 1985.

[10] 장원, 정명진, 변증남, "컴퓨터 그래픽스를 이용한 로보트 매니퓰레이터의 구현방법," 전자공학회지, 1987.

[11] S. Mohri, T. Kogawa, "Robot language," Hitachi Review, vol. 34. no. 1. 1985.

[12] C.S.G. Lee, R.C. Gonzalez, K.S. Fu, *Tutorial on Robotics*, IEEE Computer Society Press, 1983.

[13] J.J. Uicker, J. Denavit, and R.S. Hartenberg, "An iterative method for the displacement analysis of spatial mechanism", *Trans. of ASME, Journal of Applied Mechanics*, vol. 31, Series E.

[14] J.D. Foley, A. Van Dam, *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, 1982.

[15] D.F. Rogers, *Procedural Elements for Computer Graphics*, McGraw-Hill Book Company, 1985.

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

[17] R.P. Paul, Robot Manipulator, *Mathematics, Programming, and Control*, MIT Press, 1981.

[18] M. Brady, J.M. Hollerbach, T. L. Johnson, T. Lozano-Perez, and M.T. Mason, *Robot motion*, MIT Press, 1982.

[19] K.R. Symon, *Mechanics*, Addison-Wesley Publishing Company, 1979.