한국시뮬레이션학회 '99추계학술대회 논문집 1999.10.9 한양대학교

로봇의 기구학적 3차원 그래픽 시뮬레이션

조병학
(한국전력공사 전력연구원)

3-D Graphic Simulation of Robot Kinematics

Byung-Hak Cho
(Korea Electric Power Research Institute)

한국전력공사 전력연구원은 로봇 원격제어기술과 가상현실기술을 접목하는 과정의 일환으로 로봇작업을 3-D 그래픽으로 시뮬레이션하는 연구를 수행하였다. 본 논문에서는 로봇언어를 이용하여 이동경로를 생성하고 이동된 로봇의 위치와 자세정보를 그래픽변수와 연결하여 가시화하는 일련의 과정을 다룬다. 개발된 시뮬레이션 패키지는 실시간으로 로봇의 움직임을 관찰하는 목적 이외에 로봇작업경로의 생성과 로봇설계 등의 엔지니어링 목적으로도 활용할 수 있는 기능을 갖추고 있다. 이 시스템은 사용자의 편이성과 로봇의 신뢰성 및 가격적인 면 등을 고려하여 팬티엄-II급 산업용 PC와 Windows NT로 구성되어 있고, 그래픽 툴로는 Microsoft가 멀티미디어용으로 개발한 DirectX를 활용하였다. DirectX는 다양한 예제 프로그램을 포함하고 있는 Software Design Tool을 제공하므로 소프트웨어 개발시간을 단축시키는 장점을 가지고 있다.

I. 서론

로봇은 사람을 대신해서 일상적인 반복작업 또는 위험한 지역에서의 작업을 자동으로 수행하도록 활용목적에 따라 다양한 형태로 개발되어 왔다. 초기에는 자동으로 움직이는 장치를 모두 로봇이라고 했지만 현대의 기준에 의하면 프로그램이 가능하고 3개 이상의 독립된 방향으로 움직일 수 있는 것이 로봇의 범주에 속한다. 현대적인 산업로봇은 1961년 제너널 모터스사에 의해 최초로 사용된 후 1980년대의 기술적 성숙기를 거쳐 1990년대에는 일반산업은 물론 원자력산업을 포함한 극한작업분야와 의료, 서비스 등의 다양한 분야로 응용범위가 급속히 확대되고 있다. 최근에는 일반산업에 응용되는 단순반복작업용 로봇의 경우 물체를 인식하는 지능을 부여하여 정확한 위치제어를 가능케 하는 등의 방법으로 제품의 품질을 향상시키는 연구가 진행되고 있고, 극한작업분야와 의료산업에서는 로봇원격제어기술에 가상현실기술을 접목하여 원격조종자에게 현실감을 부여하는 연구가 수행되고 있다. 가상현실기술은 크게 시각, 청각과 촉감으로 나뉘어지는데 이들 중 시각적인 것이 가장 많은 정보를 제공하는 것으로 알려져 있다. 로봇의 거동은 기구학적인 면과 동력학적인 면에서 고려된다. 로봇기구학은 관절의 회전축의 방향과 회전각도 및 링크길이 등의 구조적인 특성으로부터 로봇이 움직임을 묘사할 때 적용되므로 3차원 그래픽 처리와 밀접한 관계가 있다. 동력학으로 관절에 가해지는 힘을 계산할 수는 있지만 로봇의 작업여건에 따라 불확실성이 많으므로 보통은 로봇의 손목에 힘센서를 붙여서 이 신호로부터 원격조종자가 느끼는 촉감을 얻는다. 그러나, 청각적인 효과는 계산이 불가능하므로 로봇에 두 개 이상의 음향센서를 붙여서 이를 3차원 그래픽과 결합하여 3차원 음향효과를 얻는 방법이 사용하고 있다. 로봇을 3차원으로 그래픽 시뮬레이션하는 상용 소프트웨어는 RobCAD, Workspace 등이 있지만 이들은 가격이 비싸고, 더욱이 개발건수 마다 로열티를 지불해야 한다. 일반적으로 가상현실(VR) 툴의 경우 5만여개 이상의 다각형(Polygon, 보통은 삼각형으로 그래픽엔진이 그림을 그리는 기본단위임)을 처리하지만 로봇 시뮬레이션의 경우 5천여개로 제한되고, 그래픽 오브젝트(Object, 6면체, 구, 실린

더, 원뿔, ... 등의 물체를 형성하는 기본요소)의 수도 100개 이내이므로 Windows와 DirectX가 제공하는 기본적인 기능만으로도 충분히 구현이 가능하다. 아울러 Windows의 멀티미디어 환경은 VR을 구성하기에 필요한 Audio 기능을 제공하고 있고, 로봇에 가해지는 힘을 원격조종자에게 전달하기 위한 힘궤환(Force Feedback) 조이스틱을 완벽하게 지원하는 강점을 가지고 있다. 본 논문에서는 로봇모션의 3-D 그래픽 처리과정에 초점을 맞추고, 로봇언어를 이용하여 로봇의 이동경로를 생성한 후 이동된 로봇의 위치와 자세정보를 DirectX의 그래픽변수와 연결하여 기구학적으로 시뮬레이션하는 일련의 과정을 다룬다. 개발된 시뮬레이터는 전력연구원이 보유하고 있는 실험용 로봇시스템인 KEPRO-II에 적용하여 로봇이 문자를 쓰는 과정에 적용하여 보았다.

II. 로봇 기구학

로봇은 엔드-이펙터(End-Effector)가 임의의 위치에서 주어진 자세(Orientation)를 취할 수 있도록 다수의 관절로 구성된다. 로봇은 기구학적으로 축(Joint)의 배열과 축의 각도 및 축과 축사이의 길이인 링크(Link)로 표현된다.

2.1 Kepro-II의 기구학 (Direct Kinematics)

시뮬레이션 대상인 Kepro-II 로봇의 좌표 및 기구학적 변수는 다음의 그림 1과 표 1에 보였다.

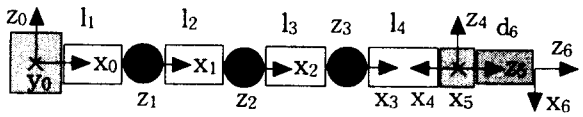

그림 1. Kepro-II 로봇의 좌표설정

표 1. Kepro-II 로봇의 링크와 좌표설정 변수

Joint Parameters			Link Parameters	
Link	Variable	Constant	Twist Angle	Link Length
1	θ_1	0	-90°	l_1
2	θ_2	0	0	l_2
3	θ_3	0	0	l_3
4	θ_4	0	-90°	l_4
5	θ_5	0	90°	0
6	θ_6	d_6	0	0

로봇 기구학은 로봇의 축 각도를 알고 있을 때 이로부터 로봇 링크의 위치와 자세 벡터를 구하기 위해 사용된다. 일반적으로 Denabit-Hartenberg식의 표현방법이 활용되고 있는데, 이 방식에서는 임의의 i번째 축의 위치가 이전(i-1)축의 위치와 i번째 축의 변환 매트릭스(Transformation Matrix)에 의해 정의된다. 이 과정을 수식으로 표현하면 다음과 같다.

$$T^i_{base} = A^1_o A^2_1 \cdots A^i_{i-1} = \prod_{j=1}^{i} A^j_{j-1} = \begin{bmatrix} n_i & s_i & a_i & p_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \text{for } i = 1, 2, \cdots n. \tag{1.a}$$

여기서, A^i_{i-1}는 축의 회전과 이동을 표현하는 변환 매트릭스이고 다음과 같이 주어진다[1].

$$A^i_{i-1} = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.b}$$

T^i_{base}는 로봇 베이스 (x_o, y_o, z_o)에서 본 로봇의 i번째 축과 링크의 자세를 나타내며 Normal 벡터 n_i, Sliding 벡터 s_i, Approach 벡터 a_i 및 Position 벡터 p_i로 구성된다.

2.2 로봇 역기구학(Inverse Kinematics)

역기구학은 기구학과는 반대로 엔드-이펙터의 자세로부터 각 축의 각도를 계산할 때 사용한다. 일반적으로 로봇의 자세에 따라 여러 개의 해를 얻을 수 있으며, 로봇의 구조와 Above/Below Arm, Left/ Right Arm 또는 Wrist의 Flip/No Flip 등의 자세에 따라 여러 개의 해를 얻을 수 있다. 역기구학을 구하는 방법으로는 고전적인 기하학적 방법, Dual Number 방법과 Iteration 방법 등이 있다. 이들 중 Dual Number 방법은 회전과

이동을 3x3 매트릭스로 분리하여 체계적으로 쉽게 해를 구하는 방법으로 알려져 있고 Iteration 방법의 경우 실시간으로 계산하기에는 계산량이 많아 적용에 어려움이 있다. Dual Number 방법 [2]에 의한 Kepro-II 로봇의 역기구학을 구하면 다음과 같다.

$$\theta_1 = \tan_2^{-1}\frac{S_1}{C_1} = \tan_2^{-1}\frac{p_y}{p_x} = \tan_2^{-1}\frac{p_y^o - d_6 a_y}{p_x^o - d_6 a_x}.$$

(2.a)

$$p_x = p_x^o - d_6 a_x, \quad p_y = p_y^o - d_6 a_y,$$

$$p_z = p_z^o - d_6 a_z.$$

$$\theta_3 = 2\delta \tan^{-1}\sqrt{\frac{(l_2+l_3)^2 - (E^2+F^2)}{(E^2+F^2) - (l_2-l_3)^2}}.$$

(2.b)

δ=+/-1 determines lower and upper arm.

$$\theta_{234} = \tan_2^{-1}\frac{S_{234}}{C_{234}} = \tan_2^{-1}\frac{a_z}{a_x C_1 + a_y S_1}.$$

$$E = -p_z - l_4 S_{234}, \quad F = p_y S_1 + p_x C_1 - l_1 - l_4 C_{234}.$$

$$\theta_2 = \tan_2^{-1}\frac{S_2}{C_2} = \tan_2^{-1}\frac{(l_3 C_3 + l_2)F - (l_3 S_3)E}{(l_3 C_3 + l_2)E + (l_3 S_3)F}.$$

(2.c)

$$\theta_4 = \theta_{234} - \theta_2 - \theta_3.$$

(2.d)

$$\theta_5 = \tan_2^{-1}\frac{a_x C_1 C_{234} + a_y S_1 C_{234} - a_z S_{234}}{a_y C_1 - a_x S_1}.$$

(2.e)

$$\theta_6 = \tan_2^{-1}\frac{n_x C_1 S_{234} + n_y S_1 S_{234} + n_z C_{234}}{s_x C_1 S_{234} + s_y S_1 S_{234} + s_z C_{234}}.$$

(3.f)

여기서, s_i와 c_i는 $\sin(\theta_i)$와 $\cos(\theta_i)$를 s_{ijk}와 c_{ijk}는 $\sin(\theta_i+\theta_j+\theta_k)$와 $\cos(\theta_i+\theta_j+\theta_k)$를 각각 나타낸다. 로봇 기구학과 역기구학의 관계를 도식으로 표현하면 아래의 그림 2와 같다. 시뮬레이션에 필요한 로봇의 기구학적 정보는 로봇 기구학식(1)로부터 계산되어 진다.

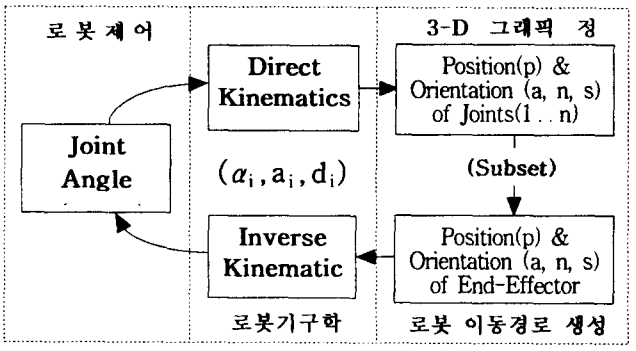

그림 2. 로봇 기구학과 역기구학의 도식적 표현

2.3 로봇작업경로 생성

작업경로의 생성은 크게 Joint Space와 Cartesian Space에서 고려되어 진다. Joint Space 방식은 컴퓨터 기술이 그다지 발전하지 않은 때에 사용된 방법으로 이 방법에서는 시작점과 종료점 사이의 조인트각도를 이동시간에 대해 선형으로 분할하여 이동하는데 여러 개의 조인트를 동시에 움직일 경우 엔드-이펙트의 이동경로(Path)를 예측할 수 없는 단점을 가지고 있다. 한편 Cartesian Space에서의 경로생성은 정지모션으로 표현되는 시작점과 끝점 사이의 최단거리를 이동경로를 택하기 때문에 이들 이동경로에서 로봇이 장애물과 충돌없이 자세를 취할 수 있는지가 관건이 된다. 따라서 정지모션은 이들을 충분히 고려하여 설정하여야 한다. 본 시뮬레이션에서는 정지모션 기법을 적용하고 있으며, 로봇의 이동경로는 Cartesian 공간에서 엔드-이펙터의 위치와 접근벡터(Approach Vector)로 정의되는 일련의 정지모션으로부터 자동으로 생성된다.[3]

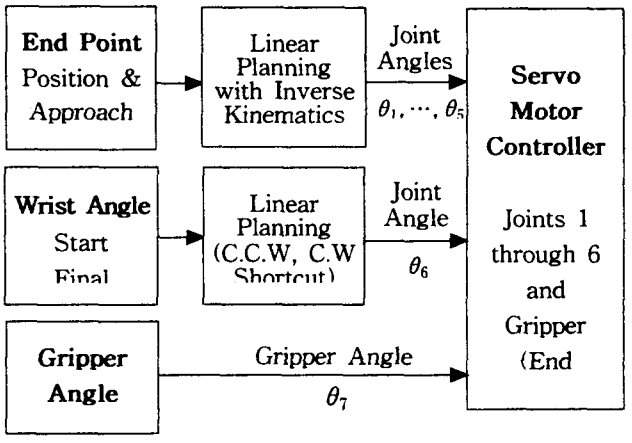

그림 3. 이동경로 생성의 도식적 표현

엔드-이펙터의 회전 방향에 자유도를 부여하기 위해 Wrist의 회전은 별도로 처리하였는데, 이 과정을 그림 3에 보였다. 엔드-이펙터의 위치와 접근벡터로 정의된 두 개의 정지된 모션으로부터 Cartesian Space에서 Path를 생성하는 과정은 그림 4에 보였다.

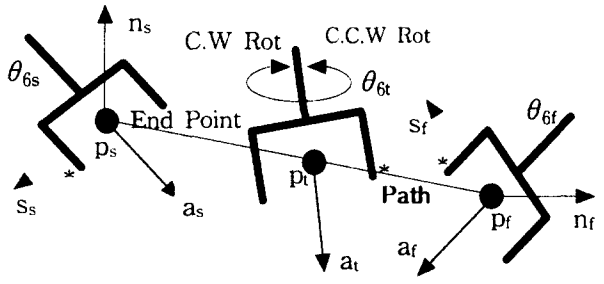

그림 4. 두 개의 정지모션으로부터의 Cartesian space에서의 이동경로 생성

2.4 상대 자세(Relative Orientation)

3차원 그래픽은 절대좌표계(World Coordinate System)에서 이루어진다. 그러나, 로봇의 기구학적 계산은 보통 첫 번째 축의 좌표계(그림 1의 (x_o,y_o,z_o))를 중심으로 한다. 따라서, 로봇이 절대좌표계의 (0,0,0)에 위치하지 않을 경우에는 로봇의 좌표를 절대좌표로 환산할 필요가 있다. 그림 5는 로봇 좌표계와 절대좌표계의 관계를 보이고 있다.

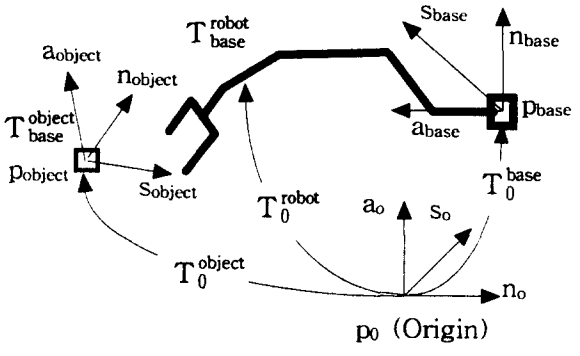

그림 5. 로봇 좌표계와 절대 좌표계의 관계

예를 들어, P_{base}에 위치한 로봇이 절대좌표 P_{Object}에 위치한 물건에 접근하기 위해서는 기구학적 계산을 수행하기 위해 로봇을 기준으로 바꾸어야 하며, 로봇 베이스에서 본 물건의 좌표 T^{object}_{base}는 아래의 식(4)와 같다. 로봇의 그래픽 처리를 위해서는 로봇 축과 링크의 절대좌표를 알아야 하고, 이 좌표는 로봇기구학에서 구한 상대좌표 T^{robot}_{base}와 로봇 베이스의 절대좌표 T^{base}_0로부터 구해지며 식(5)와 같다.

$$T^{object}_{base} = (T^{base}_0)^{-1} T^{object}_0, \quad (4)$$

$$T^{robot}_0 = T^{robot}_{base}\, T^{base}_0. \quad (5)$$

여기서,

$$(T^{base}_0)^{-1} = \begin{bmatrix} n_{base\,x} & n_{base\,y} & n_{base\,z} & -n^T_{base} p_{base} \\ s_{base\,x} & s_{base\,y} & s_{base\,z} & -s^T_{base} p_{base} \\ a_{base\,x} & a_{base\,y} & a_{base\,z} & -a^T_{base} p_{base} \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

표 2는 로봇과 물건(Object)의 좌표변환 관계를 요약해서 보이고 있다.

표2. 로봇과 Object의 좌표변환 관계

Coordinate System	Robot	Object
로봇 상대좌표	T^{robot}_{base}	T^{object}_{base}
절대좌표	T^{robot}_0	T^{object}_0
Remarks: T^{robot}_{base} : Solution of Direct Kinematics T^{object}_0 & T^{base}_0 : Known		

III. 로봇모션의 가시화(Visualization)

KeproSim은 전력연구원에서 개발한 Windows 환경의 로봇 시뮬레이터이다. 이 시뮬레이터는 로봇의 축과 링크 등의 기구학적 사양을 입력받아 로봇기구학을 계산하는 환경을 만들고 로봇언어로 명령을 내리면 로봇이 명령에 따라 움직이는 과정을 보여준다. 이 시뮬레이터의 3-D Visualization은 Microsoft사의 DirectX[4]로 구현하였으며 사용자는 방향과 거리 및 망원렌즈를 조종할 수 있는 가상 카메라를 통해 로봇의 작업과정을 실시간으로 볼 수 있다. 따라서, 정지모션에 의한 작업경로 생성시 이동 중에 발생하는 로봇바디와 구조물의 충돌 또는 역기구학의 해가 존재 여부를 확인할 수 있다. 본 절에서는

Kepro-II 로봇의 글자 쓰기 실험을 위해 작성한 로봇언어와 이를 처리하여 로봇이 명령을 실행하는 과정을 다룬다.

3.1 로봇언어와 처리기

가. 로봇언어 번역기

Kepro-II 로봇에 명령을 주는 언어는 크게 3개의 방법으로 입력이 가능하다. 먼저 축제어기(모터제어기)에서 키보드로 명령을 주는 방법이 있고, 화일에 저장된 명령을 순차적으로 수행하는 방법이 있으며 마지막으로 주제어 컴퓨터에서 TCP/IP 또는 UDP/IP 컴퓨터 통신에 의해 원격으로 명령을 주는 방법이 있다. 입력된 명령은 번역기로 입력되어 주어진 로봇언어에 해당하는 명령을 코드화하고 명령의 수행에 필요한 데이터를 입력한 후 이동경로생성 프로그램으로 보내져서 로봇 이동이 수행된다. 로봇언어에 오류가 발생한 경우 이를 CRT에 표시하거나 경고음으로 사용자에게 알리는 기능도 포함되어 있다.

나. 로봇언어 명령의 종류

본 실험용 로봇 시스템은 아래의 표3과 같은 로봇 언어 명령을 가지고 있으며 필요한 경우 다양한 명령을 생성할 수 있도록 설계되어 있다.

표 3. 로봇언어 명령

No	로봇언어	입력변수
	home	
	pause	
	resume	
0	speed	<n/a>
1		<n/a>
2	tool	<n/a>
3	turn	<(+)float:mm/sec>
4		<n/a>
5	mov_a	<+/-float:turns for gripper>
6	mov_s	<+/-float:mm, a-벡터로 이동>
7		<+/-float:mm, s-벡터로 이동>
8	mov_n	<+/-float:mm, n-벡터로 이동>
9		<0,1>: close or open gripper
10	grip	<+/-float:mm, x-축으로 이동>
11	mov_x	<+/-float:mm, y-축으로 이동>
12		<+/-float:mm, z-축으로 이동>
13	mov_y	<0,1,2>:회전방향 정의
14		<+/-float:angle, s-벡터를 >
15	mov_z	<+/-float:angle to be tilted about n>
16		<int:joint#, +/-float:absolute angle>
17	rot_type	<int:joint#, +/-float:relative angle>
18	tilt_n	<int:joint#, zero for all joints>
19	tilt_s	<int:joint#, float:angle reference>
20	joint_a	<float:mm, float:mm move n,s-axis>
21	joint_r	<char:'character' write on n,s-axis>
22	reset	<float:font size>
23	set_ref	<n/a>
24	mov_sn	<int:time delay in msec>
	write	
	fsize	
	up_arm	
	delay	

3.2 글자쓰기 실험

글자쓰기 실험을 위해 "write" 라는 로봇언어 명령을 정의하였다. 한 개의 글자는 여러 개의 벡터 성분으로 구성된다. 따라서 "write"는 로봇언어의 번역과정에서 일련의 "move" 명령들의 집합이며 이 과정은 로봇언어 번역기의 내부처리기(Internal Processor)에서 처리된다. "fsize" 명령은 글자폰트의 크기를 결정하며, 글자폰트는 Windows가 제공하는 Small Font를 사용하였다. "Welcome to Robotics Lab."을 쓰기 위한 로봇언어의 리스트는 아래와 같다.

```
@ Process 0.0  Initialization of Robot Motion
%
speed      1.0          % setup initial speed=1
cm/sec
rot_type   0        % wrist rotation is short cut
home                % move to home motion
```

```
grip      1       % open gripper
turn      0.5     % turn gripper 180 degree
turn     -1.0     % turn gripper -360 degree
turn      0.5     % turn gripper 180 degree
grip      0       % close gripper
%
@ Process 1.0 Write "Welcome to robotics lab."
%
% Get a pen
%
tool              % move to tool getting motion
grip      1       % open gripper
pause             % halt - Ctrl+Enter to resume
grip      0       % close gripper
%
% Move pen to paper
%
up_arm            % move to up_arm motion
mov_x   130.0     % move 130 mm through x-axis
mov_y   -30.0     % move -30 mm through y-axis
mov_z   -95.0     % move -95 mm through z-axis
%
% Write characters
%
fsize     0.8     % setup font size
write W           % write 'W'
write e           % write 'e'
write l           % write 'l'
write c           % write 'c'
write o           % write 'o'
write m           % write 'm'
write e           % write 'e'
write ^           % write ' '
write t           % write 't'
write o           % write 'o'
%
% Line feed
%
mov_x   220.0     % move 220 mm through x-axis
mov_y    50.0     % move 50 mm through y-axis
%
write R % write 'R'
write o           % write 'o'
write b           % write 'b'
write o           % write 'o'
write t           % write 't'
write i           % write 'i'
write c           % write 'c'
write s           % write 's'
write ^           % write ' '
write L           % write 'L'
write a           % write 'a'
write b           % write 'b'
write .           % write '.'
%
up_arm            % move to up_arm motion
%
@ Job Completed!!!
```

위의 로봇언어 리스트는 파일로 저장되며 제어화면의 "COMMAND"에서 "SOURCE"를 "FILE"로 선택한 후 실행하면 자동으로 문자쓰기가 시작된다. 펜(화이트보드용 펜)과 종이의 접촉점에서 발생하는 과도한 힘을 감쇄하기 위해 로봇의 펜은 프라스틱 원통에 스프링을 넣은 형태로 제작하였다. 로봇의 문자쓰기 장면과 그래픽 시뮬레이션을 각각 그림6과 그림7에 보였다. 로봇 엔드-이펙터의 이동 속도는 10mm/초이며 이 정도의 속도에서 +/-1mm 이내의 위치추적 오차를 보인다.

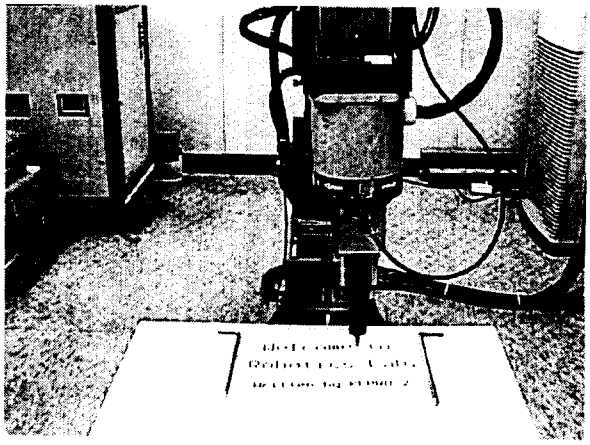

그림 6. 글자쓰기를 실행하고 있는 Kepro-II 로봇

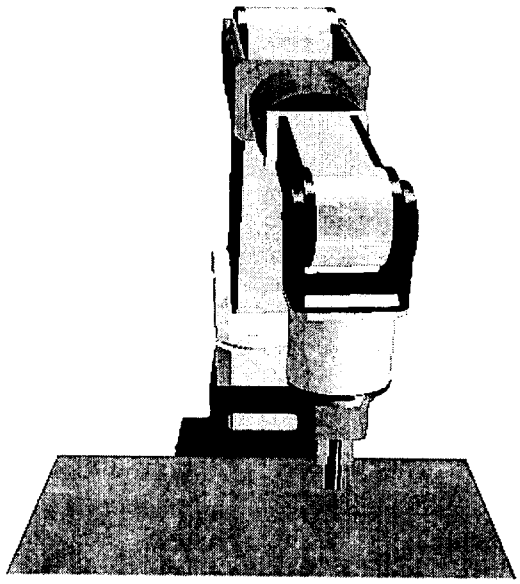

그림 7. 글자쓰기 공정의 비쥬얼 시뮬레이션

Visualization 과정을 좀더 상세히 설명하면 다음과 같다. 먼저, 로봇기구학(Direct Kinematics)이 필요로 하는 로봇 축의 각도가 주어져야 한다. 로봇 축의 각도를 얻는 경로는 크게 실시간 Visualization과 시뮬레이션으로 나뉘어진다. 실시간 Visualization의 경우 로봇 축제어기가 명령을 수행하는 도중에 로봇의 축각도를 생성하며, 시뮬레이션의 경우 두 개의 정지모션 사이의 위치를 Linear

Interpolation하여 로봇의 이동자세를 구한 후 역기구학 수식(2)를 풀어서 축각도를 구한다. 축각도가 주어지면 로봇기구학 식(1)을 풀어 로봇 축과 링크의 자세인 T^{i}_{base}를 구하고 이를 식(5)를 통해 절대좌표로 변환한 후 로봇축과 링크의 좌표 {JP[.].p, JP[.].a, JP[.].n, JP[.].s}와 {LP[.].p, LP[.].a, LP[.].n, LP[.].s}를 DirectX에 등록된 해당 오브젝트의 좌표와 Mapping하면 AutoCallBack(.) 함수에 의해 동영상 랜더링(Rendering)이 실현된다. 로봇축과 링크의 기구학적 크기인 JP[.].d[0..2]와 LP[.].d[0..2]는 DirectX의 초기화 프로그램에서 오브젝트를 생성하는 단계에서 설정된다. 아래의 그림 8은 로봇기구학 변수와 DirectX 오브젝트 좌표계의 Mapping 관계를 보이고 있다.

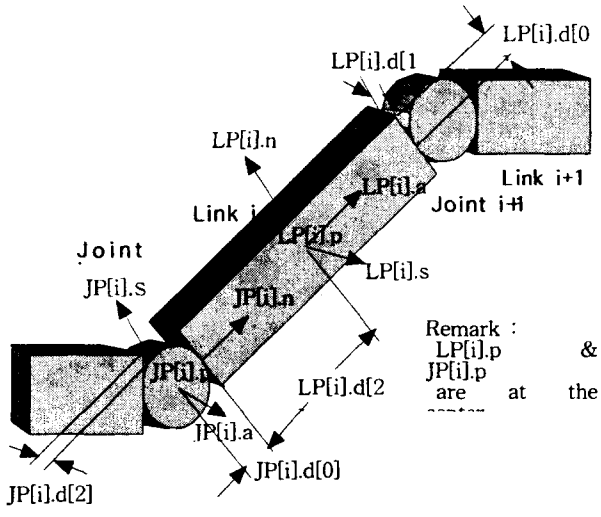

그림 8. 로봇기구학 변수와 DirectX 오브젝트 좌표계의 Mapping

Visualization에서의 가장 중요한 것은 사용자가 원하는 곳을 쉽게 볼 수 있어야 한다는 점이다. 이를 위해 DirectX에서는 카메라 오브젝트를 별도로 지원하고 있다. Kepro-Sim은 다수의 카메라를 설치하고 각각의 카메라를 Pan과 Tilt로 표현되는 회전과 이동을 통해 앵글을 바꿀 수 있도록 설계되어 있고, Zoom 기능은 카메라 위치를 앞뒤로 바꾸는 기능과 망원렌즈에 의한 확대 축소기능을 포함하고 있다.

3.3 가상현실과 로봇원격제어 기술

로봇 원격조종자는 작업현장에 설치한 카메라에 비친 영상 이미지만으로는 충분한 정보를 얻을 수 없고, 이러한 문제로 Visualization과 가상카메라의 개념이 도입되지만 그래픽모델의 오차 또는 환경의 변화 등으로 인해 정확한 위치정보를 제공하기가 어려운 실정이다. 또한 사용자가 집중력을 가지고 사물을 지속적으로 응시하는 것은 인간공학적으로 어려움이 따른다. 그러나, 오디오(음향신호)는 조종자가 조종에 집중할 수 없는 상황에서도 이벤트성의 정보를 제공한다. 이것은 동전이 떨어졌을 때 시선이 집중되는 것과 같은 이치이다. 따라서, 향후 로봇 원격제어에 가상현실기능이 더욱 도입될 전망이다. 가상현실은 본 논문에서 다룬 Visualization 이외에 오디오와 촉감이 추가된다. 전력연구원의 로보틱스팀은 현재 오디오 기능을 Visualization과 결합하는 연구와 원격조종자에게 촉감을 부여하는 연구를 병행하여 추진하고 있다.

IV. 결론

현대의 로봇에서는 하드웨어로 표현되는 로봇 본체와 일렉트로닉스 및 엔드-이펙터 이외에 원격조종자에게 보다 정확한 정보를 제공하여 작업의 품질을 높이려는 일련의 소프트웨어적인 면이 더욱 강조되고 있다. 이의 일환으로 본 연구에서는 로봇과 작업환경을 3D 그래픽으로 표현하는 Visualization 툴의 개발에 초점을 맞추고, 3차원 그래픽 렌더링 엔진인 Microsoft사의 DirectX를 이용하여 로봇의 움직임과 그래픽 오브젝트를 Mapping하는 방법 및 가상 카메라를 효율적으로 회전/이동하는 방법을 체계적으로 확립하였다. 개발된 Kepro-Sim은 로봇적용을 위한 경로생성과 시뮬레이션에 의한 로봇 모션의 확인 등의 Off-Line 엔지니어링 작업은 물론 실시간 Visualization에서도 실시간 처리와 사용자 편이성 면에서 성능의 우수성이 입증되었다. Kepro-Sim은 Windows와 PC 환경에 기초하고 있으므로 개발비용이 저렴하고 개발기간이 단축되는 장점과 상용툴을 사용할 때 개발 건마다 지불하여야 하는 로열티를 지불할 필요가 없어 다수의 시스템을 개발할 때에 유리한 장점을 가지고 있다.

References

[1] C.S.G. Lee, R. C. Gonzalez, K. S. Fu, Tutorial on Robotics, IEEE computer society, 1983, pp. 47-65, and pp. 300-318

[2] G. R. Pennock and A. T. Yang, "Application of Dual-Number Matrices to the Inverse Kinematics Problem of Robot Manipulators," Transaction of the ASME Journal of Mechanisms, Transmissions, and Automation in Design, Vol. 107, June 1985, pp. 201-208

[3] 조병학, "Simulation Studies on Robot System Applied to Nozzle Dam Installation", TM. 96EW03.P1998.482, 전력연구원

[4] Microsoft Direct3D Retained Mode, Microsoft, 1998