〈技術論文〉

PC에서 운용되는 스카라형 로보트의 오프-라인 프로그래밍 시스템

박민조* · 손 권** · 안두성***

(1994년 1월 26일 접수)

A PC Operated Off-Line Programming System for SCARA Robots

Min Jo Park, Kwon Son and Doo Sung Ahn

Key Words: Off-Line Programming System(오프-라인 프로그래밍 시스템), SCARA Robot(스카라형 로보트), Three-Dimensional Graphic(3차원 그래픽), Trajectory Planning(궤적계획), Graphic User Interface(그래픽 사용자 인터페이스), Robot Programming Command(로보트 프로그래밍 명령어)

Abstract

An off-line programming(OLP) system was proposed and developed in order to save cost and time in adjusting a robot to new workcells or applying new algorithms to actual trajectory planning. The developed OLP system was especially designed to be operated in a PC level host computer. A SCARA robot with four axes was selected as an objective robot. The OLP system developed in this study consisted of such modules as data base, three-dimensional graphics, kinematics, trajectory planning, dynamics, control, and commands. Each module was constructed to form an independent unit so that it can be easily modified or improved. The OLP system was programmed for a graphic user interface in Borland C++ language. Some of system operating commands and an interpreter were devised and used for more convenient programming of robot simulations.

1. 서 론

현재 산업계에서 겪고 있는 인력난을 해결하고 동시에 생산성을 향상시키기 위해 공장 자동화가 산업분야 전반에 걸쳐 진행되고 있으며, 이에 따라 로보트의 도입이 급증하고 있다. 시장변화에 따른 제품의 다양화와 제품 수명(life cycle)의 단축 등으로 작업환경과 내용이 빈번히 변화할 뿐만 아니라, 필요에 따라서 로보트 시스템의 일부분인 제어기나 궤적계획을 변경시킬 수 있는 시스템의 유연성이 점점 더 요구되고 있다. 온-라인 방식으로 교시·성능실험을 하기 위해서는 별도의 개발 라인이 요구되며, 작업 경험이 풍부한 전문가에 의하여 교시·성능실험이 이루어져야 함에 따라 많은 시간과 비용이 소요되는 문제점을 갖고 있다.

온-라인 작업상의 문제점은 실제작업과 유사한 환경을 가진 시뮬레이터인 오프-라인 프로그래밍(off-line programming, OLP)시스템을 이용함으로써 해결될 수 있다.[1] OLP시스템을 이용하면 로보트의 가동 중에도 동적시뮬레이션이 가능하며, 작

*부산대학교 대학원 기계공학과
**정회원, 부산대학교 기계공학과 및 기계기술연구소
***정회원, 부산수산대학교 기계공학과

업교시, 궤적계획, 제어 알고리즘 등의 개발 및 성능평가를 소프트웨어로 수행할 수 있다.[2,3] 따라서 오프-라인 방식으로 교시와 성능실험을 수행하여 생산라인의 작업중단을 방지하고, 복잡한 작업에의 적용과 성능평가도 용이하게 된다. 뿐만 아니라 OLP시스템의 개발은 원자력 발전소와 같은 극한 작업환경에서 모니터 화면과 원격조정으로 가동되는 로보트의 개발에도 활용될 수 있다.

이미 개발된 OLP시스템에는 ROSI 2,[4] STAR,[5] SILMA사에서 개발한 CimStation,[6] McDonnell Dougless사의 McAuto,[7] Deneb Robotics사에서 개발한 IGRIP[8] 등이 있으나, 이들은 워크스테이션에서 지원하는 그래픽 환경이나 도구를 이용하여 개발·운용되고 있다. PC가 대중화됨에 따라 PC에서 운용되는 OLP시스템의 개발도 요구되고 있다. 워크스테이션에서 개발한 경우에는 여러가지 그래픽 환경과 도구가 지원되나, PC에서는 그래픽 처리에 많은 어려움이 따르게 되며 계산성능 차이로 인한 한계점도 극복해야 한다.

본 논문에서는 스카라형 로보트에 대한 3차원 그래픽 시뮬레이터인 OLP시스템의 개발을 다루었다. 해석대상으로 4축 스카라형 로보트를 택하여 기구학적, 역학적 모델링과 그래픽을 위한 데이터베이스를 구성하였다. 궤적은 직각좌표계에 대한 연속경로법과 관절좌표계에 대한 PTP법(point to point method)에 의해 계획되었고, 제어방법은 계산토크법에 의한 PC 비선형 보상제어와 로보트의 파라미터가 변할 경우에 이를 추정할 수 있는 기존의 적응제어기를 사용하였다. 개발된 OLP시스템은 Borland사의 C++언어를 이용하여 그래픽 사용자 인터페이스가 가능하도록 프로그램되었으며, PC(486 microprocessor)에서 수행이 가능하고, 로보트 거동에 대한 3차원 그래픽 애니메이션이 가능하며, 적용한 로보트에 대한 궤적계획과 제어 알고리즘을 시뮬레이션한 결과를 평가할 수 있는 기능을 갖추었다. 또한 로보트 프로그래밍의 기초작업으로 로보트 동작에 대한 명령어와 그 명령어에 따라 시뮬레이션할 수 있는 해석기(interpreter)도 개발하였다.

2. OLP의 구조

본 논문에서 개발한 OLP시스템은 Fig. 1에서와 같이 크게 그래픽, 시뮬레이션, 로보트 프로그래밍의 세 모듈로 구성되어 있다. 각 모듈은 세부적인 구성모듈을 포함하고 있으나 총체적인 기능을 수행하기 위해 서로 연관되어져 있다. OLP시스템을 통해서 로보트의 구성 요소 개개에 대한 시뮬레이션을 할 수 있으며, 로보트 시스템 구성요소 사이의 관계와 영향도 살펴볼 수 있다.

그래픽 모듈에서는 선정된 로보트를 모델링하고 3차원 그래픽 알고리즘을 통해 2차원 화면에 그래픽 처리를 하며, 시뮬레이션 모듈에서는 주어진 정보로 궤적계획과 제어알고리즘을 통해 동적시뮬레이션을 수행하고, 로보트 프로그래밍 모듈에서는 앞으로 설명할 기능을 이용하여 사용자가 보다 쉽게 사용할 수 있도록 로보트 명령어와 이를 해석하는 기능을 갖추고 있다.

Fig. 1에서 알 수 있듯이 그래픽 모듈에는 데이타 베이스와 3차원 그래픽의 세부모듈이 있고, 시뮬레이션 모듈에는 입력생성, 궤적계획, 기구학, 동역학 및 제어 등의 세부모듈이 있으며, 로보트

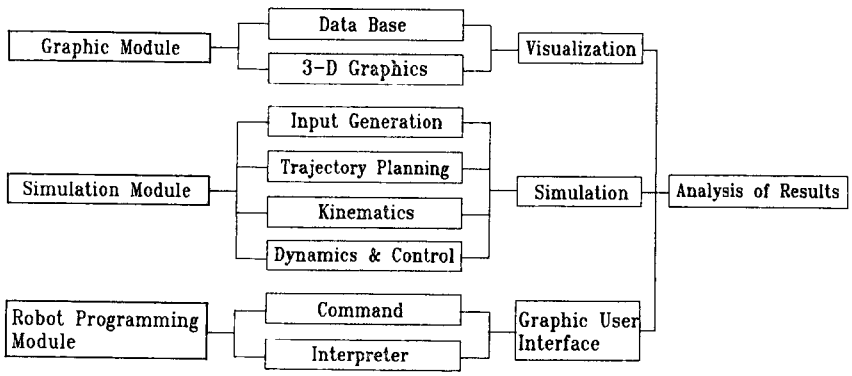


Fig. 1 Structure of off-line programming system

프로그래밍 모듈은 명령어와 해석기의 세부모듈을 가지고 있다. 이러한 세부모듈은 OLP시스템의 각 구성요소에 해당하며, 그 자체가 실제 온-라인 상태의 로보트 시스템 구성요소와 유사한 기능과 역할을 담당한다.

2.1 그래픽 모듈

여기서는 데이터베이스 모듈에서 얻은 정보와 추후에 기술할 시뮬레이션 모듈에서 얻은 로보트 운동에 대한 정보를 이용하여 그래픽기법을 통해 스크린 상에 이미지를 가시화하고, 실제로 움직이는 것 같은 현실감을 주기 위해 로보트의 운동을 애니메이션한다. 또한 애니메이션하는 매 순간마다 로보트의 각 링크에 해당하는 정보를 화면에 나타낸다.

2.1.1 데이터베이스

로보트의 형상을 그래픽으로 처리하기 위해서는 로보트에 대한 정보가 필요하다. 이 정보는 로보트의 종류와 사양에 따라 구분되어 데이터베이스 모듈에서 정의되며, 그래픽 처리기법에 따라 저장방식이나 메모리량이 다르다. 현 시스템 그래픽 처리기법은 하드웨어 시스템의 한계인 메모리의 부족으로 와이어프레임(wire frame)기법을 사용함에 따라 물체의 꼭지점을 기준점으로 하였다. 시뮬레이션을 하는 경우에는 관절 각도를 변수로 하는 동차변환행렬을 이용하여, 물체가 운동을 하는 경우의 형상에 대한 꼭지점의 좌표를 구한다.

2.1.2 3차원 그래픽

2차원 화면에 3차원 물체를 현실감있게 표현하기 위한 개념도는 Fig. 2와 같다. 첫번째 단계에서는 실제좌표계에서의 물체 좌표값을 데이터베이스에 정의하고, 두번째 단계에서는 시각좌표계의 위치와 방향을 결정하는 변수들을 이용하여 앞에서 정의한 좌표값들을 시각좌표계에서의 값으로 변환시키며, 마지막 단계에서는 원근투영원리를 적용하여 스크린상에서 3차원 이미지를 구현시킨다. 물체가 운동을 할 경우에는 매 시점에 대해 물체의 실제좌표값들을 계산하고 앞에서 설명한 단계를 통해 가시화시킬 수 있다.[9]

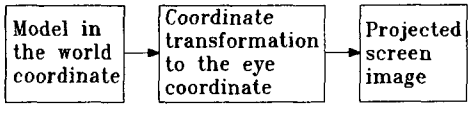


Fig. 2 Three-dimensional graphic process

화면상에서 3차원적 표현을 위해서는 실제좌표계를 시각좌표계로 변화시켜야 한다. 이를 위해 관찰자의 눈이 시각좌표계의 원점인 Fig. 3의 O_E에 있다고 간주하여 관찰자의 입장에서 물체의 좌표값을 구하였다. 실제좌표계(X_W, Y_W, Z_W)와 시각좌표계(X_E, Y_E, Z_E)의 기하학적 관계는 Fig. 3과 같으며, 그림에 표시된 변수 θ, φ, ρ를 이용하여 관찰자의 위치를 변경할 수 있다.

실제좌표계 벡터 $[V_W]$를 시각좌표계 벡터 $[V_E]$로 변환하기 위한 식은 다음과 같다.

$$[V_E]=A[V_W] \tag{1}$$

여기서, A는 좌표변환을 위한 동차 변환행렬로 다음과 같다.

$$A=\begin{bmatrix} -\sin\theta & \cos\theta & 0 & 0 \\ \cos\theta\cos\phi & \sin\theta\cos\phi & -\sin\phi & 0 \\ -\cos\theta\cos\phi & -\sin\theta\cos\phi & -\cos\phi & \rho \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

2.2 시뮬레이션 모듈

여기에는 로보트를 구동하기 위한 요소인 입력생성, 궤적계획, 기구학, 동역학 및 제어 등의 세부모듈이 있다. 입력생성 모듈은 시뮬레이션에 필요한 정보인 경유점, 각 구간에서의 경유시간, 단말효과기(end-effector)의 각도 등을 제공한다. 이러한 정보를 이용하여 궤적계획 모듈에서 선택된 경로법에 따라 궤적을 생성하고, 기구학 모듈을 이용하여 자료값들을 변환하고, 선택된 제어 알고리즘을 이용하여 시뮬레이션을 하게 된다.

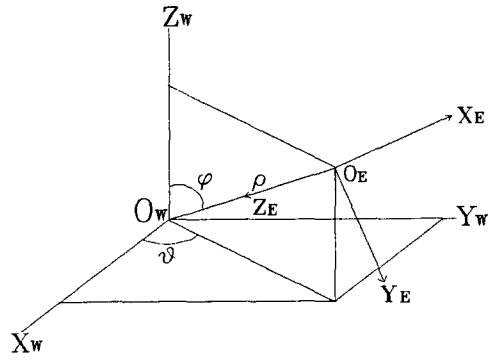


Fig. 3 World and eye coordinates

2.2.1 궤적계획

로보트의 최초 위치와 자세에서 최종 위치와 자세로 움직이는 궤적을 계획하기 위하여 관절좌표계에서는 PTP(point to point)[3]법을, 직각좌표계에서는 연속경로법(continuous path method)[3]을 사용하였다.

PTP법에서는 작업계획이 직각좌표계상에서 이루어져 경유점들이 직각좌표값으로 주어지면, 경유점들은 역기구학을 이용하여 각 관절의 좌표계인 각도나 변위로 변환해야 한다. 이 계획법은 관절공간상에서 이루어지기 때문에 실제 작업공간인 직각좌표계에서의 경로는 고려되지 않고 직각좌표계에서의 경유점은 반드시 지나가는 특성을 지니고 있다. 본 논문에서는 두번 미분 가능한 3차 다항함수를 각 관절에 대해 각 구간의 보간함수로 사용하였다.

연속경로법에서는 작업좌표계 상의 경로를 생성시키기 위해 직선과 포물선을 혼합한 함수를 사용하였다. 경로점 사이를 직선으로 연결하고 경로점 부근의 혼합구간을 포물선을 이용하여 연결함으로써 속도와 가속도가 연속이 되도록 하였다. 사용자가 경유점들과 구간 사이의 경과시간을 지정하면, 허용되는 가속도의 한도 내에서 최대속도로 운동하게 되며, 이때 가속도의 최대한계는 각 관절구동 액츄에이터의 최대 토크값에 의해 결정된다. 작업좌표계의 궤적은 역기구학을 이용하여 관절좌표계로 변환하였다.

2.2.2 기구학

일반적으로 사용되어지는 Denavit-Hartenberg (D-H)표시법[3,10]을 각 링크의 좌표계와 파라미터의 설정에 이용하였다. Fig. 4는 스카라형 로보트에 대해서 D-H표시법을 이용하여 각 링크의 좌표계를 설정한 것이다.

Table 1 Link parameters based on Denavit-Hartenberg representation

Joint	θ_i	α_i	a_i	d_i
1	θ_1	0	L_1	d_1
2	θ_2	0	L_2	0
3	0	180°	0	d_3
4	θ_4	0	0	d_4

스카라형 로보트의 링크 파라미터를 구하면 Table 1과 같다. 스카라형 로보트는 3개의 회전관절과 하나의 병진관절로 구성(RRTR)되기 때문에 표에서 θ_1, θ_2, d_3, θ_4가 링크의 관절변수가 된다. Table 1을 이용하여 기준 직각좌표계와 단말효과가 좌표계 사이의 동차변환행렬을 구하면 다음과 같다.

$$A_4^0=\begin{bmatrix} \cos\theta_{1+2-4} & \sin\theta_{1+2-4} & 0 & L_1\cos\theta_1+L_2\cos\theta_{1+2} \\ \sin\theta_{1+2-4} & -\cos\theta_{1+2-4} & 0 & L_1\sin\theta_1+L_2\sin\theta_{1+2} \\ 0 & 0 & -1 & d_1+d_3-d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

대부분의 로보트 제어는 관절좌표계에서 행하여지나 작업계획, 작업교시, 궤적계획 등은 직각좌표계에서 표현되어진다. 본 논문에서 해석한 로보트는 형상이 단순하기 때문에 기하학적 방법으로 역기구학 문제를 해결하였다. Fig. 5는 스카라형 로보트에서 두개의 링크만을 나타낸 그림이다. 여기

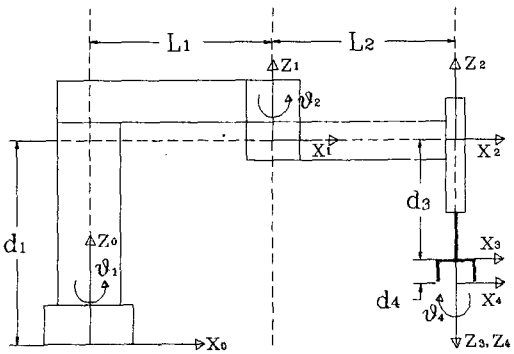


Fig. 4 Link coordinates based on Denavit-Hartenberg representation

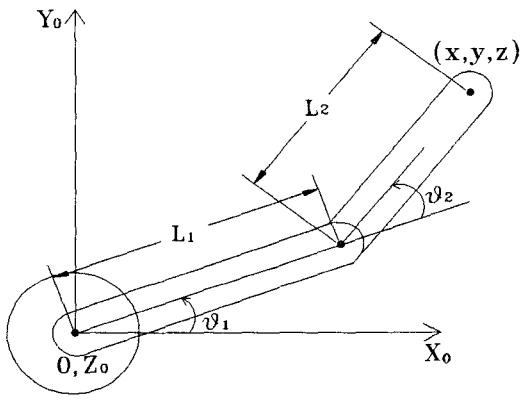


Fig. 5 Coordinate system of a two-link robot

서, 링크 1, 2, 4의 각도에 대한 식을 유도할 수 있으며 병진운동을 하는 링크 3에 대해서는 Fig. 4에서 쉽게 구할 수 있다. 직각좌표계에서의 단말효과기의 좌표값(x, y, z)와 로보트의 링크길이 L_1, L_2, d_1을 이용하여 관절변수 θ_1과 θ_2를 구하는 식을 유도하면 다음과 같다.

$$\theta_1=\tan^{-1}(\frac{y}{x})-\theta^* \tag{4}$$

$$\theta_2=\tan^{-1}(\frac{\pm\sqrt{1-d^2}}{d}) \tag{5}$$

여기서,

$$\theta^*=\tan^{-1}\frac{L_2\sin\theta_2}{L_1+L_2\cos\theta_2}$$

$$d\equiv\frac{x^2+y^2-L_1^2-L_2^2}{2L_1L_2}\text{이다.}$$

식(5)에서 링크 1과 2의 자세에 따라 두가지 해를 구할 수 있으며, 이것은 식(5)의 오른쪽 항의 ±부호에 의해 결정되어진다. Fig. 5는 부호가 +인 경우에 오른쪽 팔(right arm)형태를 나타내고 있다. 실제 OLP시스템은 작업공간의 한계와 관절의 운동범위를 고려하여 궤적계획이 행해지기 때문에, 두가지 해에서 적절한 값을 선택해야 하고 경로의 연속성을 고려하여 링크 1과 2의 자세(right, left arm)가 변화하여야 한다.

병진운동을 하는 링크 3의 변수 d_3는 유일하게 z 축으로 운동하기 때문에 쉽게 구할 수 있으며, 단말효과기의 각도 θ_4도 식(4)와 (5)를 이용하여 다음과 같이 구할 수 있다.

$$d_3=d_1-z \tag{6}$$

$$\theta_4=\theta_1+\theta_2-a \tag{7}$$

여기서, a는 직각좌표에서 결정되어지는 단말효과기의 각도를 의미하며, 일반적으로 사용자에 의해 값이 정해지게 된다.

2.2.3 동역학 및 제어

OLP시스템에서 동적 시뮬레이션을 수행하기 위해서는 적용 로보트의 운동방정식과 제어 알고리즘이 필요하다. 본 논문에서는 스카라형 로보트의 운동방정식을 유도하고, 제어방법으로는 기존의 계산토크법과 계산토크법을 이용한 적응제어법을 적용하였다.

(가) 운동방정식

매니퓰레이터를 강성체로 가정하고, 운동방정식을 라그랑지법[11]으로 유도하면 다음과 같다.

$$H(\Theta)\ddot{\Theta}+h(\Theta,\dot{\Theta})+G(\Theta)=\tau \tag{8}$$

여기서, 첫번째 항은 관성행렬, 두번째 항은 원심력과 코리올리력, 세번째 항은 중력항, 오른쪽 항은 토크값을 각각 나타낸다. 각 링크에 대한 운동방정식은 부록에 정리하였다.

(나) 계산토크법

계산토크법의 알고리즘은 원하는 각도와 그 순간의 속도 및 가속도를 이용하여 동적 모델식에서 제어입력항인 토크나 힘을 계산한다. 제어 알고리즘의 블록선도는 Fig. 6과 같으며, 그림에서 번호 1, 2, 3은 각각 관성모멘트, 코리올리력, 중력항의 보상을 의미하고, K_P와 K_V는 각각 비례게인과 미분게인을 나타낸다. 입력으로는 각 시간에 대한 관절의 각도, 각속도, 각가속도이며, 이 값들은 궤적계획과 기구학에 의해 구해진다. 이 제어법칙에 의한 제어입력은 다음과 같다.

$$\tau(t)=H(q)[\ddot{q}^d(t)+K_v\{\dot{q}^d(t)-\dot{q}(t)\}+K_p\{q^d(t)-q(t)\}]+h(q,\dot{q})+G(q) \tag{9}$$

제어입력항인 식(9)를 운동방정식 (8)에 대입하면 다음과 같다.

$$\begin{aligned}&H(q)\ddot{q}+h(q,\dot{q})+G(q)\\&=H(q)[\ddot{q}^d(t)+K_v\{\dot{q}^d(t)-\dot{q}(t)\}\\&\quad+K_p\{q^d(t)-q(t)\}]+h(q,\dot{q})+G(q)\end{aligned} \tag{10}$$

식(10)은 관성력, 코리올리력, 중력 등을 모두 보상한 경우에 해당되며, 이 식을 정리하면 다음과 같은 오차방정식을 구할 수 있다.

$$H(q)[\ddot{e}(t)+K_v\dot{e}(t)+K_pe(t)]=0 \tag{11}$$

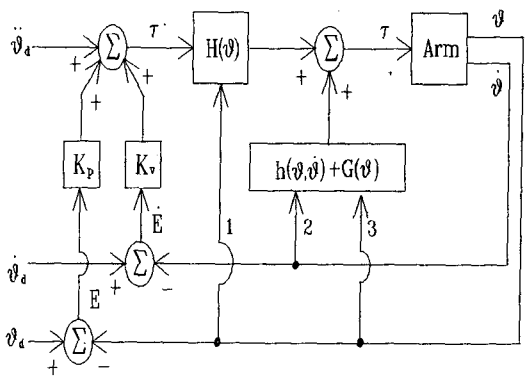


Fig. 6 Block diagram of computed torque method

단, $e(t)=q^d(t)-q(t)$이며, $\dot{e}(t)=\dot{q}^d(t)-\dot{q}(t)$이다.

오차방정식 (11)은 동적모델식이 정확하다는 가정하에서 성립한다. 비선형 연립미분방정식 (11)의 해는 4위의 Runge-Kutta법[12]을 이용하여 수치적분으로 구하였다.

(다) 적응제어

적응제어는 동적 모델링과 파라미터와 오차, 운전중의 부하변동, 고속운전에 따른 성능저하와 불안정을 해결할 수 있는 한가지 방법이며,[13,14] 본 논문에서는 원하는 이상적인 동특성을 지니는 기준모델을 설정하여, 이 기준모델의 출력이 일치하도록 플랜트의 매개변수를 조종하는 방식인 기준모델 적응제어방식(MRAC)을 사용하였으며, 추정자(estimator)로 불확실한 플랜트의 파라미터를 추정하고, 이 정보를 바탕으로 제어입력항을 계산하는 간접제어방식을 택하였다.

Fig. 7은 적응제어기의 블록선도이며, 그림에서 알 수 있듯이 앞 절의 계산토크법의 블록선도에 단지 적응요소만 첨가될 뿐 거의 같은 구조를 가지고 있다. 적응요소는 서보오차를 관측하고 이것을 이용하여 관성행렬과 코리올리력 및 원심력를 추정한다. 리아프노브(Lyapunov)이론을 이용하여 적응법칙을 유도하였으며 제어기의 안전성도 증명하였다.[15]

2.3 로보트 프로그래밍 모듈

로보트 프로그래밍 모듈은 앞에서 언급한 그래픽 모듈과 시뮬레이션 모듈을 실제 OLP시스템에서 실행하기 위한 부분으로 프로그래밍 명령어와 해석기를 세부모듈로 가지고 있으며, 그래픽 사용자 인터페이스의 한 종류인 버턴과 마우스를 이용할 수 있게 구성하였다. 입력생성 모듈에서 얻은 정보와 이미 정의된 명령어를 입력하게 되면, 해석기 모듈에서 명령어를 해석하고, 이에 따라 시뮬레이션하게 된다. 명령어에는 로보트의 거동과 관련된 명령어와 화일관리와 그래픽에 관련된 보조 명령어가 있다. 이와 같은 명령어를 조합하면 작업수준의 로보트 프로그래밍도 가능하게 될 것이다. 본 논문에서 개발한 해석기는 로보트 프로그래밍의 컴파일러와 같이 명령어의 조합을 해석할 수는 없으나, 그 전 단계로 한 개의 명령어에 대한 해석하고 그 명령어에 맞게 시뮬레이션할 수 있도록 하였다. 명령어와 해석기를 계속 확장·개선한다면, 개발한 OLP시스템을 효율적으로 운영할 수 있을 것이다.

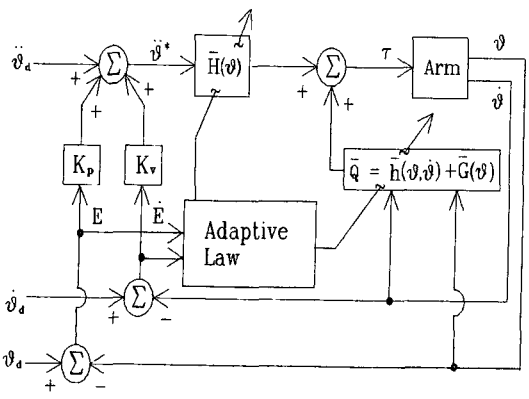


Fig. 7 Block diagram of adaptive controller

3. OLP시스템의 구성

개발된 OLP시스템은 Fig. 8에서와 같은 기본 메뉴화면을 가지며, 셋업모드, 교시모드, 프로그래밍 모드가 있다. 또한 각 모드의 기본적인 기능 이외에 화일관리, 편집기 등과 같은 여러가지 부수적인 기능을 가지고 있다. Fig. 9는 실제 OLP시스템의 구성도를 나타낸다. 그림에서 셋업모드는 프로그래밍모드의 각 옵션을 결정하고, 교시모드는 입력기능의 정보를 제공하며, 프로그래밍모드에서 버턴기

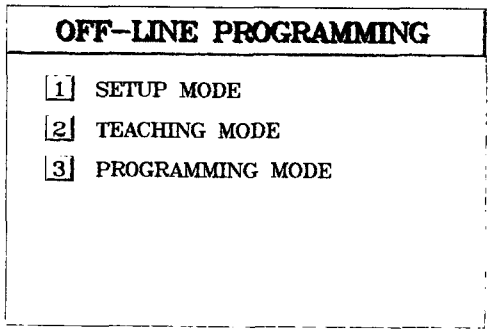


Fig. 8 Main menu screen of OLP system

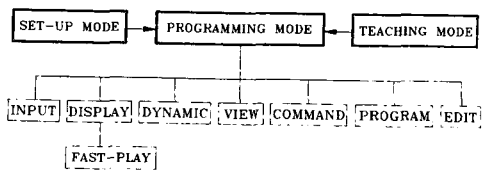


Fig. 9 Functional structure of OLP system

능을 이용하여 로보트 시뮬레이션을 수행한다.

3.1 교시모드

교시모드에서는 Fig. 10과 같은 화면을 지원한다. 그림에서 x, y, z축과 단말효과기의 각도에 대한 +와 −버튼을 이용하여 단말효과기의 위치와 로보트의 자세를 변경할 수 있으며, 각 위치와 자세에 해당하는 정보를 알 수 있다. 단말효과기가 로보트의 작업영역을 벗어나는 경우에는 경고음과 함께 작동이 되지 않도록 OLP를 구성하여 작업영역에서의 정보만을 얻을 수 있으며, 화면 오른쪽 하단에 여러 화일기능을 이용하여 원하는 위치와 자세의 정보를 화일에 저장할 수도 있다.

3.2 프로그래밍모드

프로그래밍모드를 통해 3차원 표현기법으로 로보트의 거동을 보이고, 로보트의 각 관절과 실제 좌표계에서 궤적계획, 동역학, 제어 등의 성능평가를 할 수 있으며, 프로그래밍모드의 부수적인 기능을 통해 편리하게 시뮬레이션을 할 수 있도록 하였다. 본 논문에서 개발한 OLP시스템의 각 기능버턴에 대해서 기술하면 다음과 같다. 이 기능들은 시뮬레이션 순서와 연관이 있으며 그래픽 사용자 인터페이스의 관점에서 개발되었다.

3.2.1 INPUT기능

단말효과기의 초기위치, 최종위치, 경유점, 시간, 단말효과기의 각도 등의 정보를 Fig. 11에서와 같이 화면의 오른쪽 하단의 창에서 입력하면 셋업 모드에서 정한 궤적계획법으로 입력에 따른 궤적을 생성하게 되고, 그것에 해당하는 직각좌표계와 관절좌표계에서의 데이터값을 화일에 저장하게 된다. 만약 입력항이 궤적계획조건에 맞지 않으면, 경고음과 메시지를 통해 궤적계획 조건에 적합하지 않는 이유를 표시한다.

3.2.2 DISPLAY와 FAST-PLAY기능

DISPLAY기능은 INPUT기능에서 계획한 경로를 따라 로보트가 거동하도록 셋업 모드에서 설정한 제어 알고리즘을 이용하여 동적 시뮬레이션한다. 설정한 자료추출시간(sampling time)에 대해 미분운동방정식을 수치적분하여 풀고 이 값을 이용하여 그래픽 모듈을 통해 3차원적으로 로보트가 거동하는 모습을 애니메이션한다. Fig. 12는 DISPLAY기능을 실행한 경우의 시뮬레이션 결과를 애니메이션하는 화면이다. 그림의 하단부분에 매 시

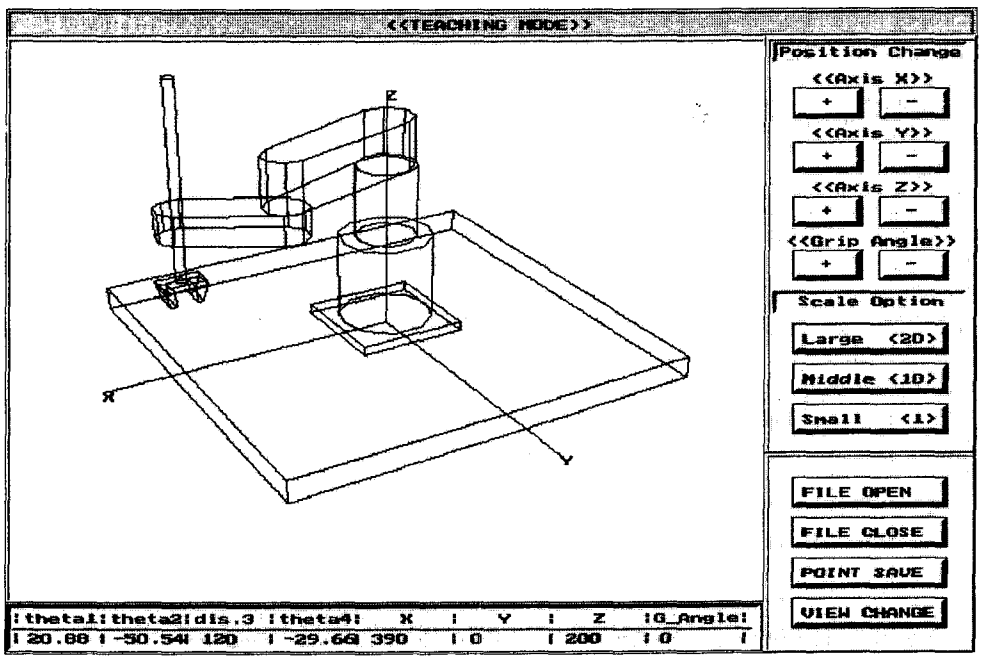


Fig. 10 Screen in TEACHING MODE

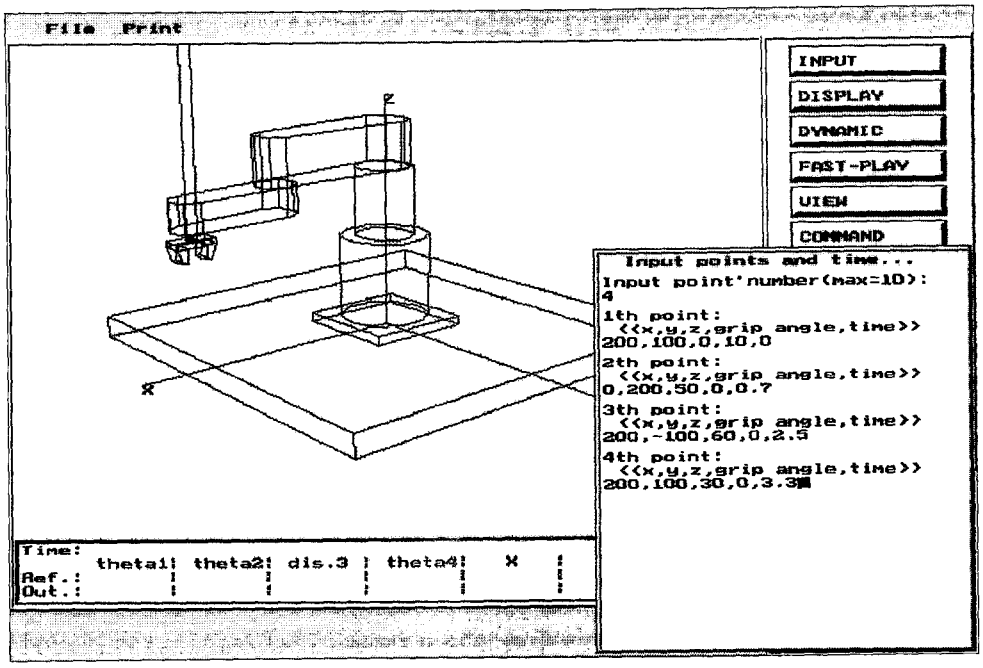


Fig. 11 Screen in INPUT FUNCTION

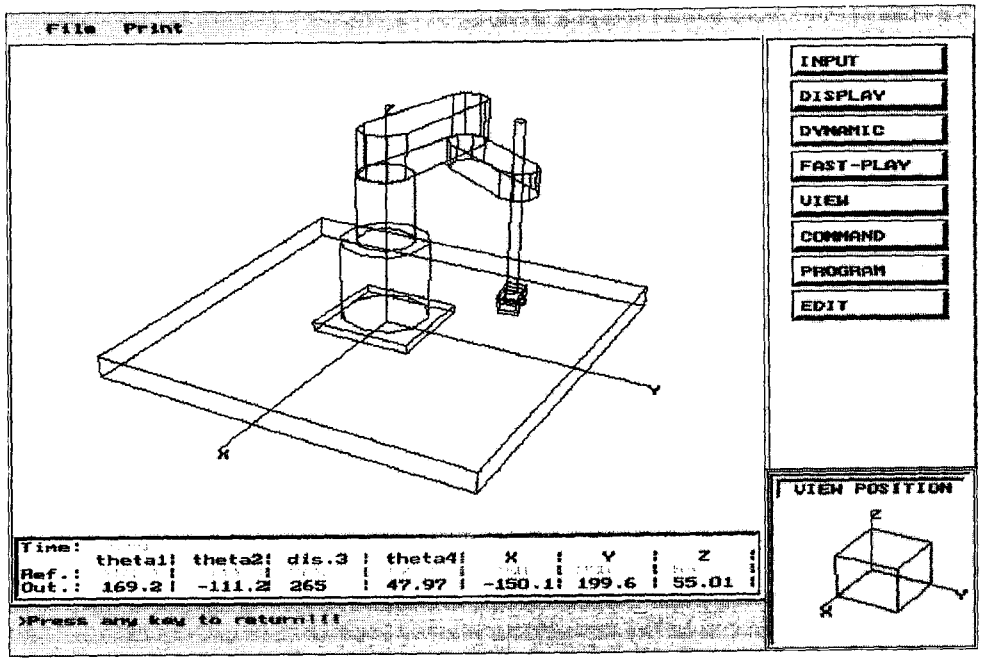


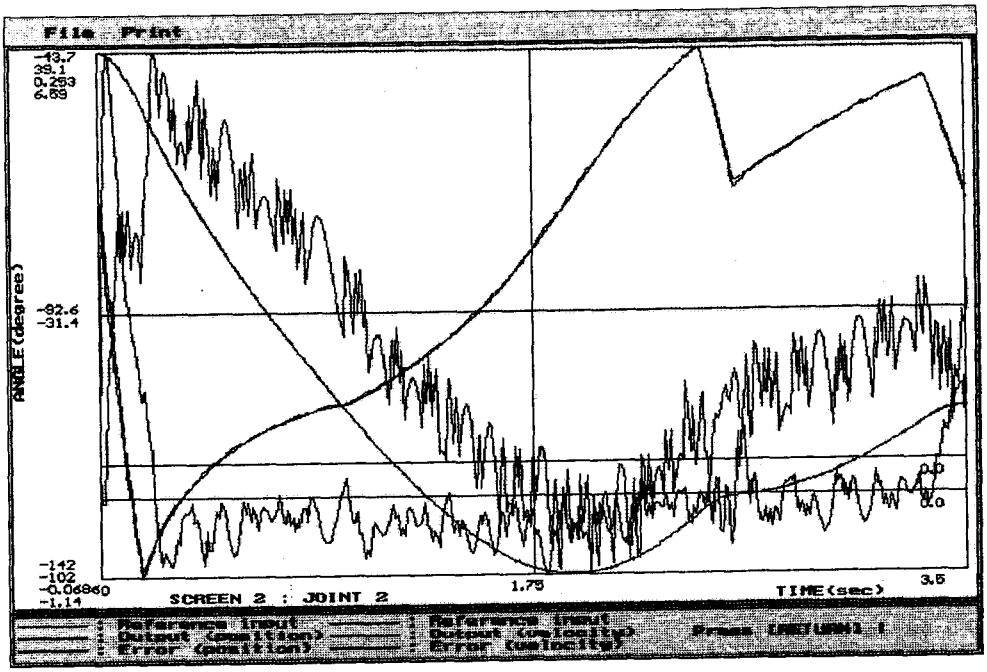
Fig. 12 Screen in DISPLAY FUNCTION

점에서의 링크와 단말효과기에 대한 위치정보가 표시됨을 알 수 있다.

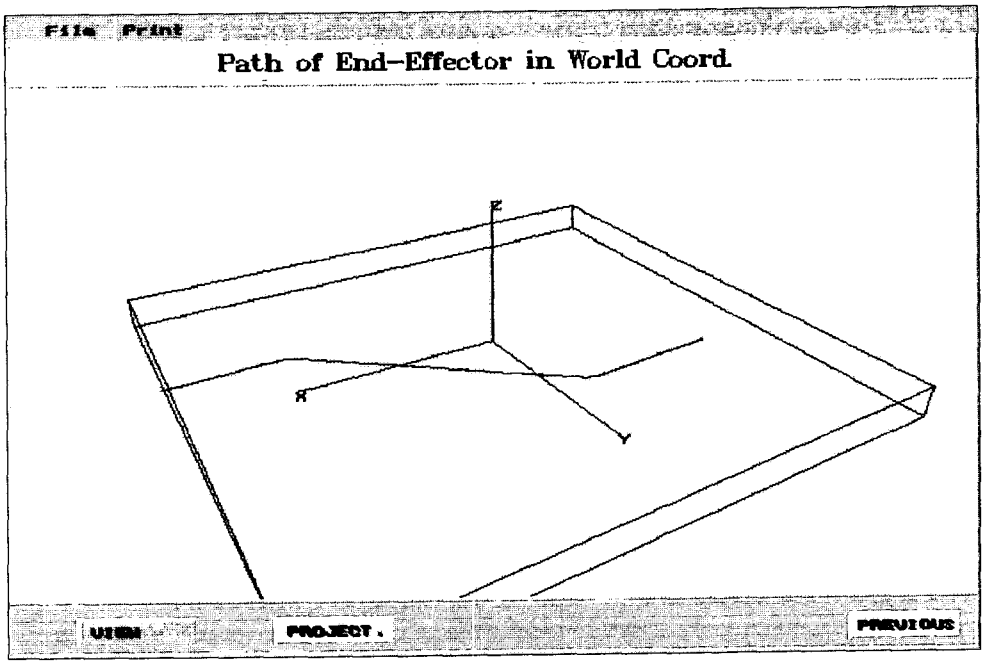
FAST-PLAY 기능은 DISPLAY기능시 화일로 저장되어 있는 동적 시뮬레이션 결과를 이용하여 사용자가 원하는 시각위치에 대해 수행된 결과를 여러번 반복하여 볼 수 있게 한다.

3.2.3 DYNAMIC기능

DISPLAY기능에서 구한 동적 시뮬레이션 결과를 사용자가 쉽게 평가할 수 있는 기능을 제공한다. 즉 관절좌표계에서의 각 링크의 궤적과 속도를 비교 평가할 수 있는 기능과 직각좌표계에서의 단말효과기의 경로를 평가할 수 있는 기능을 가지고



(a) Results in the joint coordinate



(b) Trajectory in the cartesian coordinate

Fig. 13 Screen in DYNAMIC FUNCTION

있으며, VIEW기능을 이용하여 시뮬레이션한 결과를 여러 위치에서 관찰할 수 있다. Fig. 13(a)는 관절좌표계에서 기준입력, 출력, 오차를 나타내며, Fig. 13(b)는 직각좌표계에서의 궤적을 나타낸다.

3.2.4 VIEW기능

VIEW기능을 이용하여 시각위치를 변화시킬 수 있으며 θ, φ, ρ(Fig. 3참조)는 시각좌표계를 결정하는 변수로 원하는 화면에 맞게 사용자가 임의로

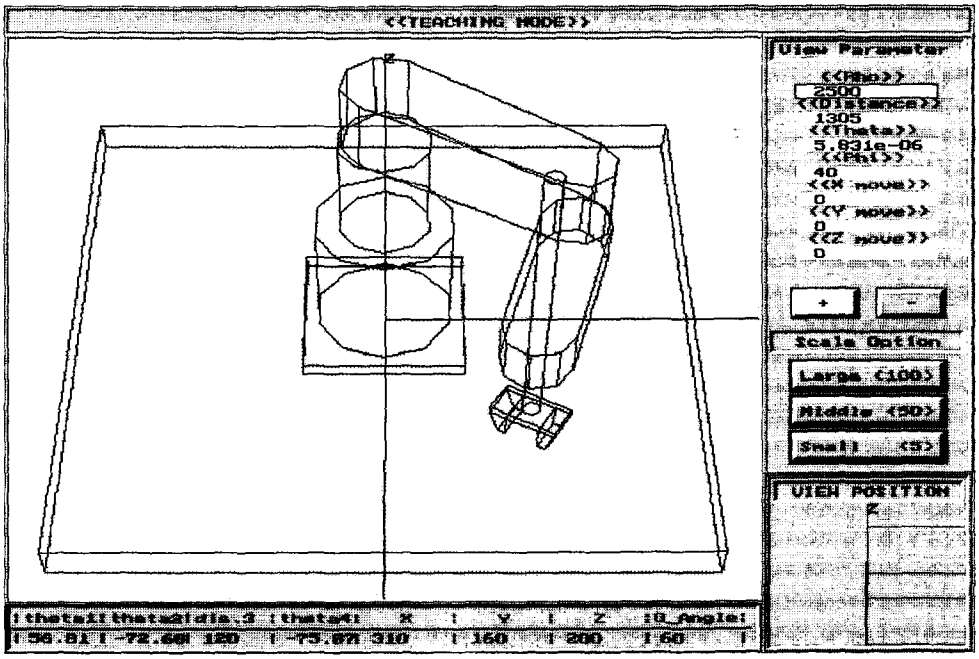


Fig. 14 Screen in VIEW FUNCTION

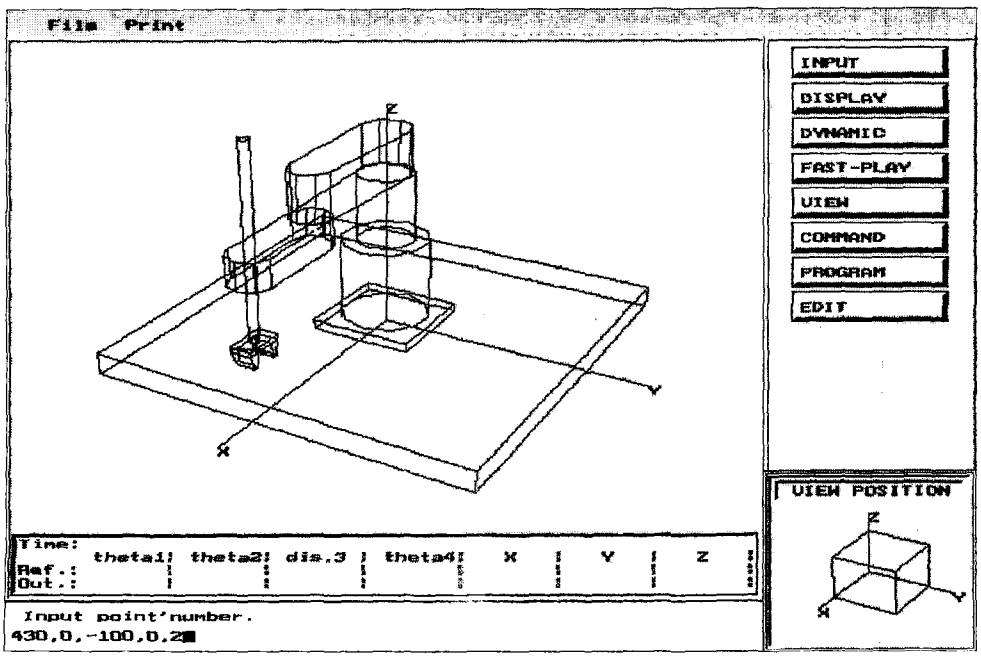


Fig. 15 Screen in COMMAND FUNCTION

조정할 수 있다. Fig. 14는 실제 프로그램상에서의 화면 예이다. 변수 θ, φ, ρ는 화면에서 Theta, Phi, Rho로 표시되고, distance는 시각위치와 화면 사이의 거리를 나타낸다. X, Y, Z move의 값은 화면 전체가 시각좌표계의 X, Y, Z축 방향으로 이동시키는 역할을 한다. 사용자는 키보드나 마우스를 이용하여 위의 변수를 변화시켜 시각좌표계의 위치를 원하는대로 변경할 수 있다.

3.2.5 COMMAND기능

효율적인 프로그래밍을 위해 인터프리터용 명령어를 개발하고, 이를 사용하여 시뮬레이션이 가능하도록 하였다. COMMAND기능을 실행하게 되면 화면의 왼쪽 아래에 있는 영역에 커서가 생기고 여기서 키보드를 이용하여 명령어를 입력할 수 있다. 현재까지 만들어진 명령어로는 MOVE(단말효과기를 이동시키는 명령어)와 화일관리 명령어인 NEW FILE 등이 있으며, 화면상의 로보트의 위치를 초기화하는 명령어도 있다. 이렇게 입력한 명령어를 시뮬레이션할 수 있도록 처리하는 해석기도 개발하였는데, 이 해석기는 단지 한 명령어에 대해 처리하고 시뮬레이션하는 능력만을 갖추고 있다. Fig. 15는 화면의 아래 영역에 인터프리터용 명령어를 입력하는 화면 예이다.

4. 결 론

본 논문에서는 PC에서 운용되는 4축 스카라형 로보트에 대한 OLP시스템을 개발하였다. 3차원 그래픽, 궤적계획, 기구학, 동역학, 제어 알고리즘, 명령어, 해석기, 그래픽 사용자 인터페이스(GUI)등으로 구성되는 OLP시스템을 개발하였으며, 이를 통해 로보트의 작동과정과 구성요소의 특성을 화면상에서 평가할 수 있도록 하였다. 개발된 시스템은 다음과 같은 특징을 지니고 있다.

(1) 일반적으로 널리 보급된 PC(486 microprocessor)에서 운용되도록 프로그래밍되었다.

(2) 새로운 작업교시를 할 경우에 이 시스템을 통해 시뮬레이션을 수행하고 작업에 대해 프로그래밍을 한다면, 온-라인상태에서는 단지 교시만 함으로써 자동화 생산라인에 적용하는 시간과 비용을 최소화하는 데 이용될 수 있다.

(3) 화면상에서 실시간 시뮬레이션을 통해 여러 작업에 대한 준비시간을 최적화할 수 있으며, 로보트의 일부 구성요소를 변경할 경우에 대해서도 오프-라인 상태로 여러가지 성능실험을 할 수 있다.

(4) 고속운전에 따르는 생산성 향상의 효과와 허용된 오차를 고려하여 궤적계획시 경로에 대한 시간의 최적화를 이루는 데 이용될 수 있다.

(5) 인간이 로보트와 작업환경을 직접 볼 수 없는 우주공간이나 원자력발전소와 같은 극한 작업환경에서 화면만을 이용하여 로보트를 원격조정할 경우에도 활용될 수 있다.

개발된 OLP시스템을 4축 스카라 로보트에 적용하기 위해서는 사전에 실험을 통한 검증이 필요하며, 이를 위해 현재 로보트를 구입하여 준비하고 있는 중이다. 실제 로보트에서는 개개의 관절에 대한 제어기를 주로 사용하고 있어, 본 OLP시스템에 슬라이딩모드 제어기를 비롯한 개별 관절제어기를 추가할 계획으로 있다. 은선 및 은면의 제거와 음영처리가 가능한 그래픽, 충돌 검색 및 회피를 위한 알고리즘, 사용자가 대화적으로 도움을 받을 수 있는 편의성, 여러 제어기를 사용해 시뮬레이션한 결과를 비교 및 분석할 수 있는 방법 등에 대한 추가적인 연구가 수행된다면 보다 효율적인 OLP 시스템이 구현될 수 있을 것이다.

참고문헌

(1) 전향식, 최영규, 1992, "개인용 컴퓨터 그래픽스를 사용한 로보트 시뮬레이터의 연구," 부산대학교 공과대학 논문집, 제43집, pp. 119~126.

(2) Craig, J. J., 1988, "Issues in the Design of Off-Line Programming Systems," *International Symposium of Robotics Research,* Cambridge, Massachusetts, pp. 379~389.

(3) Craig, J. J., 1989, *Introduction to Robotics Mechanics and Control,* Addison-Wesley, New York.

(4) Dillmann, R. and Huck, M., 1986, "A Software System for the Simulation of Robot Based Manufacturing Processes," *International Journal of Robotics Research,* Vol. 2, No. 1, pp. 3~18.

(5) Hornick, M. L. and Ravani, B., 1986, "Computer-Aided Off-Line Planning and Programming of Robot Motion," *International Journal of Robotics Research,* Vol. 4, No. 4, pp. 18~31.

(6) SILMA Inc., 1989, *The CimStation User's Manual, Ver. 4.0,* Available from SILMA Inc., 1601 Saratoga-Sunnyvale Rd., Cupertino, California.

(7) Shumaker, G. G., 1980, "*Robotics-Air Force* Project," *Computer World*, March, pp. 32~45.

(8) Yoffa, N. A., 1988, "Off-Line Programming for Automative Spot Welding,"*Robotics World*: Flexible Automation & Intelligent Machines, April, pp. 24~25.

(9) Park, C. S., 1985, *Interactive Microcomputer Graphics*, Addison-Wesley, New York.

(10) Fu, K. S., Gonzalez, R. C. and Lee, C. S. G., 1987, *Robotics : Control, Sensing, Vision and Intelligence, McGraw-Hill, New York.*

(11) *Asada, H. and Slotine, J. J. E., 1986, Robot Analysis and Control*, John Wiley & Sons, New York.

(12) James, M. L., Smith, G. M. and Wolford, J. C., 1993, *Applied Numerical Methods for Digital Computation*, Harper Collins, New York.

(13) Craig, J. J., Hsu, P. and Sastry, S. S., 1987, "Adaptive Control of Mechanical Manipulators," *International Journal of Robotics Research*, Vol. 6, No. 2, pp. 16~28.

(14) Slotine, J. J. E. and Li, W., 1987, "On the Adaptive Control of Robot Manipulator," *International Journal of Robotics Research*, Vol. 6, No. 3, pp. 49~59.

(15) 박민조, 1994, "3차원 그래픽을 이용한 오프-라인 프로그래밍의 개발," 부산대학교 석사학위 논문 대학원.

부 록

스카라 로보트의 운동방정식 (8)을 각 링크에 대해 기술하면 다음과 같다.

$$H_{11}\ddot{\theta}_1+H_{12}\ddot{\theta}_2+H_{14}\ddot{\theta}_4+h_{112}\dot{\theta}_1\dot{\theta}_2+h_{122}\dot{\theta}_2^2=\tau_1$$
$$H_{21}\ddot{\theta}_1+H_{22}\ddot{\theta}_2+H_{24}\ddot{\theta}_4+h_{211}\dot{\theta}_1^2=\tau_2$$
$$H_{33}\ddot{\theta}_3+G_3=\tau_3$$
$$H_{41}\ddot{\theta}_1+H_{42}\ddot{\theta}_2+H_{44}\ddot{\theta}_4=\tau_4$$

여기서,

$$H_{11}=m_1L_{c1}^2+I_1+m_2(L_1^2+L_{c2}^2+2L_1L_{c2}\cos\theta_2)+I_2$$
$$+(m_3+m_4)(L_1^2+L_2^2+2L_1L_2\cos\theta_2)+I_4$$
$$H_{12}=H_{21}$$
$$=m_2(L_{c2}^2+L_1L_{c2}\cos\theta_2)+I_2+(m_3+m_4)$$
$$(L_2^2+2L_1L_2\cos\theta_2)+I_4$$
$$H_{22}=m_2L_{c2}^2+I_2+m_3L_2^2+m_4L_2^2+I_4$$
$$H_{14}=H_{41}=H_{24}=H_{42}=-I_4$$
$$H_{33}=m_3+m_4$$
$$H_{44}=I_4$$
$$h_{112}=-2m_2L_1L_{c2}\sin\theta_2-2(m_3+m_4)L_1L_2\sin\theta_2$$
$$h_{122}=-m_2L_1L_{c2}\sin\theta_2-(m_3+m_4)L_1L_2\sin\theta_2$$
$$h_{211}=m_2L_1L_{c2}\sin\theta_2+(m_3+m_4)L_1L_2\sin\theta_2$$
$$h_{212}=-0.5\{m_2L_1L_{c2}\sin\theta_2+(m_3+m_4)L_1L_2\sin\theta_2\}$$
$$=-h_{221}$$
$$G_3=-g(m_3+m_4)$$