'92 韓國自動制御學術會議論文集 1992. 10. 19~21

로봇 머니퓰레이터를 위한 적응학습제어 알고리즘의 구현

* 이 형기 최 한호 정 명진

한국과학기술원 전기및 전자공학과

Implementation of an adaptive learning control algorithm for robot manipulators

Hyung Kyi Yi, Han Ho Choi, and Myung Jin Chung

Dept. of Electrical Eng. KAIST

ABSTRACT

Recently many dynamic control algorithms using robot dynamic equation have been proposed. One of them, Kawato's feedback error learning scheme requires neither an accurate model nor parameter estimation and makes the robot motion closer to the desired trajectory by repeating operation. In this paper, the feedback error learning algorithm is implemented to control a robot system, 5 DOF revolute type Movemaster. For this purpose, an actuator dynamic model is constructed considering equivalent robot dynamic model with respect to actuator as well as friction model. The command input acquired from the actuator dynamic model is the sum of products of unknown parameters and known functions. To compute the control algorithm, a parallel processing computer, Transputer, is used and real-time computing is achieved. The experiment is done for the three major link of Movemaster and its result is presented.

I. 서론

산업이 발달함에따라 인간은 로봇이 다양한 작업을 좀더 신속, 정확하게 수행할 것을 요구하고 있다. 그러나 로봇의 동력학적 특성때문에 전통적인 PID 알고리즘으로는 고속 정밀 제어가 어렵다. 그래서 로봇의 동력학을 이용하는 많은 다이나믹 제어방식들이 제안되었다. 이 방식들중 적응제어는 플랜트의 정확한 모델링이 불필요하여, 로봇과같이 모델링이 어려운 제어대상에 적용하기 쉽다. 초기의 MRAC(Model Reference Adaptive Control), self-tuning[3]등의 적응제어이론은 선형 플랜트를 대상으로 이론이 전개되었기 때문에 로봇같은 비선형 플랜트에 직접 적용하기가 어려웠다.

Craig는 로봇 동력학식을 링크 길이, 질량, 관성등의 로봇 파라미터에대해 선형으로 표현하여, 로봇 파라미터를 적응하여 찾아내고, 이 파라미터를 이용하여 로봇 동력학식을 푸는 적응제어 기법을 소개하였다.[5] 또 Slotine와 Li는 앞의 방법이 각가속도를 측정해야 하는 단점을 보완하여 실제 각가속도 대신에 원하는 경로의 각가속도를 이용하는 방법을 제안하였다. 각가속도는 잡음에 매우 민감하여 일반적으로 측정이 어렵기 때문이다.[6] Sadegh과 Horowitz는 각 축의 실제 측정된 각도를 사용하지 않고 추종하고자 하는 각도를 이용하여 로봇의 역동력학을 계산하는 기법을 제안하였다. 이 방법은 원하는 경로의 각도를 사용하기 때문에, 로봇의 역동력학을 off line으로 미리 계산할 수 있다는 장점과 잡음에 강인하다는 장점이 있다.[2] 또 신경회로와 PD 되먹임 제어기를 병렬로 사용하여, 운전 초기에는 PD 제어기가 주가되고, 작업을 행하는 과정에서 신경회로가 PD 제어기의 출력을 줄여 나가게 학습을 하여, 학습이 진행됨에 따라 신경회로가 PD 제어기를 대체하게하는 feedback error learning[1][7]이 있다. 이 방법은 신경 회로 제어가 갖는 일반적인 특성을 그대로 가지게 되어,제어대상이나 그 환경 모델이 불필요하고, 불확실한 플랜트혹은 부하등의 환경 변화에 대한 적응이 가능하고, 엔코더로부터의 측정 잡음에 강인성이크며, 병렬계산에 의한 고속 실시간제어 가능, 정보의 fault tolerance등의 장점이 있다. 이방법은 근본적으로 Sadegh와 Horowitz의 방법과 유사하다.

본 논문에서는 모터의 동력학과 기계적인 마찰력을 고려하여 로봇의 동적모델을 확장하고, 확장된 모델에 위의 feedback error learning 방식을 적용하고, 실제 실험을 통하여 알고리즘의 유용성을 검증하였다. 알고리즘을 계산하기 위하여 병렬형 CPU인 Transputer 5개를 사용하였고, 병렬연산을 수행하여, 실시간 제어가 가능하도록 하였다.[8] 로봇은 Mitsubishi사의 Movemaster 5축형을 사용하여 3축 까지 제어하였다.

II. 적응학습제어 알고리즘

전형적인 적응제어이론은 선형플랜트를 대상으로 전개되어 로봇과 같은 비선형 플랜트에 직접 적용하기가 힘들다. 로봇의 동력학식을 로봇 파라미터에대해 선형화시켜 이를 해결할 수 있다.

1. Reparametrization[4]

로봇을 일렬로 연결된 강체로 가정하고, 마찰력을 무시한 모델링은 다음과 같다.

$$\tau = H(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q)$$

q : $n\times 1$ joint displacement vector

$H(q)$: $n\times n$ inertia matrix

$C(q,\dot{q})\dot{q}$: $n\times 1$ centripetal and Coriolis torque

$G(q)$: $n\times 1$ gravitational torque

τ : $n\times 1$ joint torque or force

이 식을 질량, 관성모멘트, 링크길이 등의 로봇 파라미터에대해 선형으로 표현할 수 있다.

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) \triangleq Y_1(q,\dot{q},\ddot{q})p_1$$

Y_1은 아는 함수의 행렬($n\times r$)이고 p_1은 로봇파라미터의 열벡터($r\times 1$)이다.

2. 마찰력 모델링

단순화한 모델에서 마찰력은 종종 무시되지만 실제 실험시에 마찰력은 모델링 오차로 작용하여 로봇의 성능에 큰 영향을 준다. 마찰력은 보통 각속도의 함수로 표현된다. 다음은 마찰력의 여러 모델을 소개한다.[9]

1) Coulomb/stiction : Coulomb 마찰력은 각속도가 영이 아닌 경우 작용하는 상수값이다. stiction 마찰력은 모터가 정지상태에서 출발하기 위하여 극복해야 할 마찰력이다.

$$f(\dot{q}) = \alpha\, sgn(\dot{q})$$

2) Viscous 마찰력 : 속도에 비례하여 작용하는 마찰력으로 모터의 회전 방향에 따라 다른 값일 수 있다.

$$f(\dot{q}) = \alpha_j\, sgn(\dot{q}) + \beta_j \dot{q}$$

j 는 모터의 회전 방향에 따른 비대칭성을 나타낸다.

3) 각회전위치 의존성 : 마찰력은 속도의 함수일 뿐만아니라 각 회전위치에 의존한다.

$$f(\dot{q}) = K_f \sin(\omega_0 q + \phi)$$

4) Downward bend : 낮은 속도에서 stiction 마찰력값에서 지수함수적으로 감소한 후 다시 증가한다.

$$f(\dot{q}) = [\alpha_0 + \alpha_1 e^{-\beta|\dot{q}|}]\, sgn(\dot{q})$$

$\alpha_0 + \alpha_1$은 stiction 마찰력이고 β는 감쇠 상수이다.

이상을 종합하면 다음과 같이 모델링이 된다.

$$f(\dot{q}) = [\alpha_0 + \alpha_1 e^{-\beta_1|\dot{q}|} + \alpha_2(1 - e^{-\beta_2|\dot{q}|})]\, sgn(\dot{q})$$

앞장에서 설명한 파라미터에대한 선형화를 시키기 위하여 위의 모델을 약간 수정하자. 윗 식은 다음식으로 근사할 수 있다.

$$f(\dot{q}) = [\alpha_0 + \alpha_1|\dot{q}|^{0.5} + \alpha_2|\dot{q}|]\, sgn(\dot{q})$$

$$\triangleq Y_2(\dot{q})p_2$$

$$Y_2(\dot{q}) = [sgn(\dot{q}) \quad |\dot{q}|^{0.5} sgn(\dot{q}) \quad |\dot{q}| sgn(\dot{q})]$$

$$p_2 = [\alpha_0 \quad \alpha_1 \quad \alpha_2]^T$$

마찰력도 위와같이 파라미터에대해 선형으로 표현할 수 있다.

3 모터의 동력학

대부분의 산업용 로봇은 구동기로 영구자석형 DC모터을 사용하고 있으며 모터축과 로봇조인트 사이에 기어박스를 설치하여 로봇 동력학을 모터 축으로 전달한다. 영구 자석형 DC 모터의 동력학을 다음과같이 전기적인 동력학과 기계적인 동력학으로 표현한다.

전기적 동력학

$$E = L\frac{di_a}{dt} + R i_a + K_b \frac{\dot{q}}{n}$$

$$T = K_t i_a$$

기계적동력학

$$T = J_m \frac{\ddot{q}}{n} + f(\dot{q}) + n\tau \qquad (2.3.1)$$

J_m : 모터관성

f : 마찰력

τ : 로봇 토크

K_t : 토오크 상수

L : 모터 인덕턴스

R : 모터 저항

K_b : 역기전력 상수

i_a : 모터 아마추어 전류

E : 모터 입력 전압

T : 모터 토크

n : 기어비 (로봇 각도/모터 각도)

제어기의 출력은 모터의 입력 전압으로 가해지기때문에 실제 제어할 플랜트는 모터를 포함한 로봇이다. 모터 입력 전압에대해 동력학식을 표현하면 다음과 같다.

$$E_i = \frac{R_i}{K_t} T_i + K_b \frac{\dot{q}}{n}$$

모터의 인덕턴스는 매우 작으므로 무시했다. 식(2.3.1)을 T에 대입하면,

$$E = \frac{R}{K_t n} J_m \ddot{q} + \frac{R}{K_t} f(\dot{q}) + \frac{R}{K_t} n\tau + K_b \frac{\dot{q}}{n}$$
$$\triangleq Y(q, \dot{q}, \ddot{q})p$$

로봇의 동력학, 모터의 동력학, 마찰력을 모두 고려한 위의 동력학식도 파라미터에대한 선형화될 수 있다.

4. 적응학습제어

로봇 파라미터를 적응학습하여 로봇 동력학을 보상하고 PD 제어기와 합하여 토크 명령을 만들어 내는 적응학습기법이 있다. 제어입력과 학습법칙은 다음과 같다.

$$E = Y(q_d, \dot{q}_d, \ddot{q}_d)\hat{p} + K_p e + K_d \dot{e}$$
$$\frac{d\hat{p}}{dt} = \Gamma Y(q_d, \dot{q}_d, \ddot{q}_d)^T e_r, \qquad \Gamma > 0$$
$$e = q_d - q_p$$
$$e_r = \dot{e} + \lambda e, \qquad \lambda > 0$$

학습이 시작되기 전에는 PD 되먹임제어기만으로 제어를 하고, 파라미터의 학습이 진행되면서 controller는 로봇의 역동력학을 배우게 된다. 학습이 완료되면 controller는 로봇의 역동력학을 완전히 보상한다. 이 방식은 원하는 경로의 각도를 사용하여 off line 계산이 가능하고, 학습할때 잡음에 강인하다. 이방법은 feedback error learning[1][7]으로 불리어지는데, 그림 1은 그 구조도이다. 실험에 사용한 Y는 다음과 같다.

$$Y = \begin{bmatrix} f^T & 0 & 0 \\ 0 & g^T & 0 \\ 0 & 0 & h^T \end{bmatrix}$$

$$C_i = \cos(q_i), S_i = \sin(q_i), C_{ij} = \cos(q_i + q_j), S_{ij} = \sin(q_i + q_j),$$
$$f_0 = \ddot{q}_1,\ f_1 = S_2\ddot{q}_3,\ f_2 = C_{23}\ddot{q}_2,\ f_3 = C_{23}\ddot{q}_3,\ f_4 = C_2\dot{q}_2^2,$$
$$f_5 = S_{23}\dot{q}_2^2,\ f_6 = S_{23}\dot{q}_3^2,\ f_7 = S_2C_2\dot{q}_1\dot{q}_2,\ f_8 = S_{23}C_{23}\dot{q}_1\dot{q}_2,$$
$$f_9 = S_2S_{23}\dot{q}_1\dot{q}_2,\ f_{10} = C_2C_{23}\dot{q}_1\dot{q}_2,\ f_{11} = S_{23}C_{23}\dot{q}_1\dot{q}_3,$$
$$f_{12} = C_2C_{23}\dot{q}_1\dot{q}_3,\ f_{13} = S_{23}\dot{q}_2\dot{q}_3,\ f_{14} = C_2^2\ddot{q}_1,$$
$$f_{15} = C_{23}^2\ddot{q}_1,\ f_{16} = C_2S_{23}\ddot{q}_1,\ f_{17} = S_{23}^2\ddot{q}_1,$$
$$f_{18} = |\dot{q}_1|^{0.5} sgn(\dot{q}_1),\ f_{19} = sgn(\dot{q}_1),\ f_{20} = |\dot{q}_1| sgn(\dot{q}_1)$$

$$g_0 = \ddot{q}_2,\ g_1 = S_2\ddot{q}_1,\ g_2 = S_2\ddot{q}_1,\ g_3 = C_{23}\ddot{q}_1,\ g_4 = \ddot{q}_3,\ g_5 = S_3\ddot{q}_3,$$
$$g_6 = S_2C_2\dot{q}_1^2,\ g_7 = S_{23}C_{23}\dot{q}_1^2,\ g_8 = S_2S_{23}\dot{q}_1^2,\ g_9 = C_2C_{23}\dot{q}_1^2,$$
$$g_{10} = C_3\dot{q}_3^2,\ g_{11} = C_3\dot{q}_2\dot{q}_3,\ g_{12} = S_{23},\ g_{13} = C_2,$$
$$g_{14} = sgn(\dot{q}_2),\ g_{15} = |\dot{q}_2|^{0.5} sgn(\dot{q}_2),\ g_{16} = |\dot{q}_2| sgn(\dot{q}_2)$$

$$h_0 = \ddot{q}_3,\ h_1 = C_{23}\ddot{q}_1,\ h_2 = \ddot{q}_2,\ h_3 = S_3\ddot{q}_2,$$
$$h_4 = S_{23}C_{23}\dot{q}_1^2,\ h_5 = C_2C_{23}\dot{q}_1^2,\ h_6 = C_3\dot{q}_2^2,\ h_7 = S_{23},$$
$$h_8 = sgn(\dot{q}_3), h_9 = |\dot{q}_3|^{0.5} sgn(\dot{q}_3),\ h_{10} = |\dot{q}_3| sgn(\dot{q}_3)$$

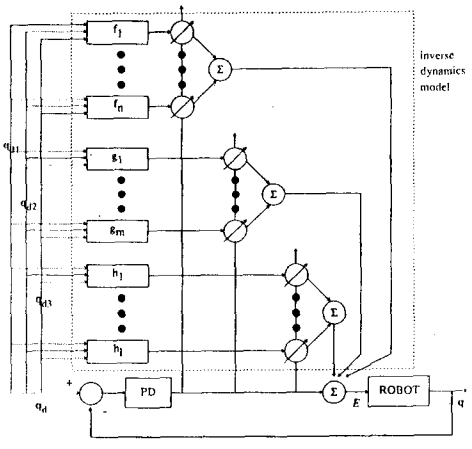

그림 1. feedback error learning의 구조도

III. 시스템 구성

로봇은 복잡한 비선형 미분방정식으로 모델링되기 때문에 로봇동력학을 고려한 제어 알고리즘은 계산하는데 많은 시간이 소요된다. 본 논문에서는 영국 INMOS사의 Transputer T800-20 4개와 T222-20 1개를 사용하여 병렬처리 계산기를 구성하여 계산시간을 단축시켰다. I/O 인터페이스 보드는 encoder 펄스로부터 로봇 조인트의 각 위치를 계산하고 Transputer에서 계산된 토그 명령을 아날로그 신호로 바꾸어 서보 앰프로 전달한다. 서보는 한국 모트로닉스사의 HSC-100R 속도형 서보를 사용하였고 제어대상 로봇은 미쯔비시 중공업사의 Movemaster를 사용하였다.

1. Transputer의 특징[10]

Transputer는 영국의 INMOS사에서 개발한 병렬 처리형 CPU이다. Transputer에는 4개의 직렬 통신로가 있으며 이를 적절히 연결하여 원하는 구조의 병렬 처리기를 구성할 수 있다. 실험에 사용한 IMS T800은 32 bit microprocessor이고 중앙처리장치(CPU)는 최대 30 MIPS로 동작하며 내부에 수치계산전용처리장치(FPU)가 있어 1.5 MFLOPS의 계산속도를 가진다. 20 Mbps의 속도를 갖는 4개의 직렬 통신로(communcation link)가 있어 이를 사용하여 Transputer끼리 정보를 주고 받는다. 내부에 4 Kbyte의 내부 정적기억장치(SRAM)를 가지고 있으며 외부기억장치는 4 Gbyte까지 직접 사용할 수 있다. 본 논문에서는 Occam language[11]를 이용하여 프로그램을 개발하였는데, Occam은 Transputer와 함께 개발된 언어이고 각 명령들은 어셈블리 언어처럼 Transputer의 내부 구성과 밀접한 관계를 가지고 있다. Occam은 매우 간단한 명령들로 이루어져 있으며 명령의 형태가 정형화 되어 있어 이해하기 쉽다는 장점이 있다. 다른 언어들과 구분되는 Occam의 가장 큰 특징은 SEQ, PAR, ALT의 약속어가 있다

는 것이다. SEQ는 SEQuence에서 나온 말로서 보통의 다른 언어와 마찬가지로 순차적으로 일을 처리한다. PAR은 PARallel, 즉 병렬로 일을 하는 것이고, ALT는 alternative, 선택적으로 일을 수행하는 것을 말한다. 또한 통신로로의 데이타 전송을 위해 ? , ! 의 약속어가 있다. ?는 통신로를 통한 정보의 수신을 나타내고, ! 는 전송을 나타낸다.

2. 병렬처리 장치의 설명

본 논문에서는 PC의 슬롯에 꽂아 쓸수 있는 INMOS의 B008 보드를 구입하여 사용하였다. B008에는 10개의 slot이 있어 이곳에 Transputer와 각종 외부장치들로 구성된 Transputer module(TRAM)이라고 불리는 장치를 꽂아 사용한다. 이 논문에서 사용한 TRAM은 앞에서 설명한 T800-20와 외부 기억장치로 구성되어 있다. TRAM B404는 2MByte의 외부 기억장치를 갖고 있고 slot 0에 꽂아 사용한다. B404는 Occam 프로그램을 컴파일하고 PC와 다른 Transputer와의 통신을 중개해 주는 역할을 한다. 다른 3개의 TRAM은 B411이며 1MByte의 외부기억장치를 갖고있다. B411은 임의의 슬롯에 꽂아 서로 통신로를 연결하여 사용자가 원하는 병렬처리 구조를 만들 수 있다. B008 보드는 slot에 꽂혀 있는 TRAM끼리 두가지 방법으로 통신로를 연결할 수 있도록 설계되어 있다. 4개의 통신로 중 2개는 하드웨어적으로 직접 선을 연결 하여 사용하고 나머지 2개는 소프트웨어로 프로그래밍 할 수 있다. 동적으로 각 통신로 사이의 연결은 불가능하여 프로그램 수행 중 병렬처리구조를 바꿀 수는 없다. 미리 병렬구조를 setting하고 PC에서 컴파일한 프로그램을 통신로를 통하여 각 transputer에 download한다.

3. I/O 인터페이스 보드의 제작

I/O 인터페이스 보드는 로봇과 transputer 보드와의 다리 역할을 하며 모터의 encoder 출력을 각 위치로 바꿔주고 Transputer에서 계산된 결과를 아날로그 신호로 변환한다. 사용한 encoder는 incremental 형으로, A상과 B상의 펄스를 내보낸다. 이 두 신호를 조합하여 up/down 신호를 생성한다. 이 up/down 신호를 16 bit 카운터에 기록하여 로봇의 현재 위치를 알아낸다. DAC(digital to analog converter)로는 아날로그 디바이스사의 AD7247을 사용했다. 이는 12bit의 해상도를 가지며 내부에 2개의 DAC가 들어있고 latch를 포함하고 있다.

4. 전체 시스템의 구성

IBM PC AT의 확장 슬롯에 IMS B008 보드를 꽂고, flat cable을 이용하여 I/O 인터페이스 보드와 연결하였다. PC에서 프로그램을 작성하고 이를 Occam compiler를 이용하여 컴파일한 후 B008에 다운로드시킨다. 실험결과는 Transputer의 메모리에 저장되었다가, 실험이 다 끝난후 PC로 가져간다. 그림 2는 전체 시스템 구성도이다.

IV. 실험방법및 실험결과

대상 시스템으로 미쯔비시 중공업사의 5축 revolute type의 로봇, movemaster를 사용하였다. 로봇의 각 관절은 직류 전동기에 의해 구동되며 광학적 incremental encoder에 의해 위치를 feedback 받는다. Encoder의 해상도는 모터에서 0.9 deg/pulse 이다. 작업대상으로는 로봇의 베이스좌표에서 xy평면상의 중심이 (0.347 , 0) 반지름이 7 cm 인 원으로 하였다. 로봇축의 속도 한계로 인하여 4초에 원을 한 번 돌도록 하였고, 2 msec 샘플링을 하였다.

$$K_p = \begin{bmatrix} 20 & 0 & 0 \\ 0 & 20 & 0 \\ 0 & 0 & 20 \end{bmatrix}, \quad K_d = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 30 \end{bmatrix}$$

일때 PD 되먹임 제어기만을 사용하였을때의 실험결과을 그림 3에 보였다. 적응학습의 효과를 관찰하기 위해서 대략적인 PD 게인을 사용하였다.

$\lambda = 0.4$, $\Gamma = 0.005\mathrm{I}$ (I : 3×3 unit matrix) = 0.4일때 원을 100번 학습한 후의 결과를 그림 4에 보였다. II장의 Y에 원하는 경로의 q를 사용하였고, $\dot{q}$와 $\ddot{q}$는 수치적 미분에의한 값이다. 반복적으로 원을 100번 그리게하여 학습이 진행됨에 따라 rms error가 줄어드는 것을 그림 5에서 확인할 수 있다. 1 축은 초기 error가 1.089도에서 0.046도로, 2 축은 6.42도에서 0.0697도로, 3축은 2.06도에서 0.387도로, 학습이 진행되면서 error가 줄어들었다. 학습이 진행된 후 1 축의 토크 입력을 그림 6에 보였다. 각속도의 부호에 따라 결정되는 Coulomb 마찰력에 의해 토크의 전체모양이 결정되었다. 이와같은 현상이 나타난 이유는 기어에 의해 로봇의 동력학이 기어비만큼 감소하고 마찰력이 상대적으로 커진 원인인 것 같다. 그림에서 friction은 학습에의해 배운 토크중 마찰력성분을 나타내고, dynamic은 마찰력을 포함한 로봇의 동력학을 학습한 토크이고, total torque는 dynamic과 PD 제어기를 합하여, 로봇의 제어 입력으로 실제 가해진 토크이다. 그림 7은 7 cm 원에서 학습한 후 4 cm 원을 그리게 한 결과이다. 적응 학습에 의하여 로봇의 역동력학을 배우기때문에 작업대상을 바꾸어도 잘 추종함을 볼수있다.

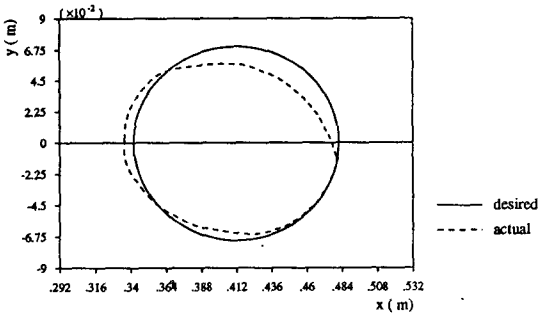

그림 3. PD 제어기만에 의한 원 작업 수행 결과(직교 좌표계)

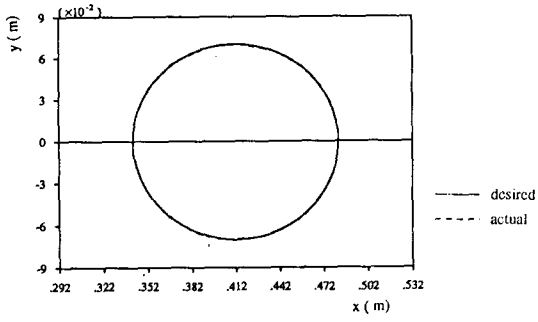

그림 4. 적응학습후 원 작업 수행 결과(직교 좌표계)

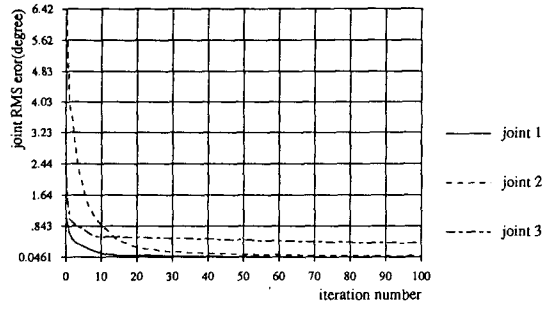

그림 5. 적응학습진행시 오차의 감소 궤적

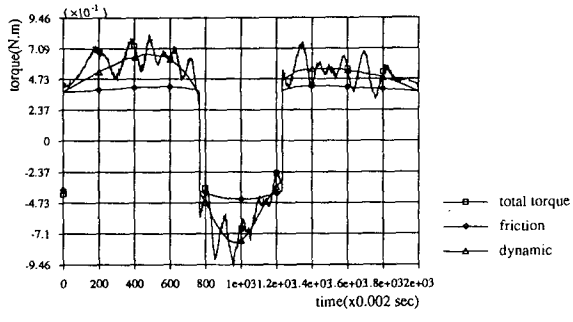

그림 6. 토크 입력

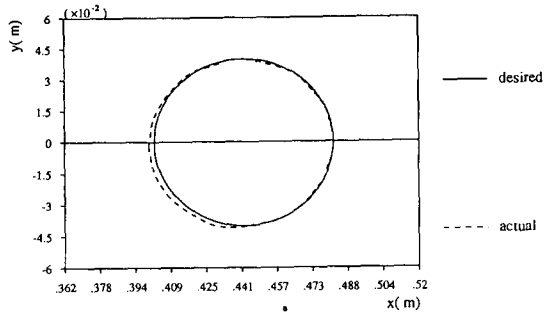

그림 7 적응학습후 다른 작업 수행 결과

V. 결론

이론전개단계에서 무시되는 마찰력 모델링과 구동기의 동력학을 고려하여 로봇의 동적모델을 확장하고 확장된 모델에 feedback error learning 알고리즘을 적용하였다. Transputer 5개를 사용하여 병렬연산을 수행하여 real time control이 가능하였다. 실험에서 마찰력이 입력토크에 크게 작용함을 알았고, 알고리즘이 매우 강인하고, 성능이 뛰어남을 확인하였다.

REFERENCE

[1] 최 한호, "적응학습에 의한 매니퓰레이터 제어", KAIST 석사 학위 논문, 1990.

[2] N.Sadegh, R.Horowitz, "Stability and Robustness Analysis of a Class of Adaptive Controllers for Robotic Manipulators" , Inter. Jour. of Robotics Research, Vol.9, No.3, pp.74-92, June 1990.

[3] K.J.Astrom, ADAPTIVE CONTROL, ADDISON WESLEY, 1989.

[4] P.Khosla, and T.Kanade, "Parameter identification of Robot dynamics", IEEE Conf. Decision an Control. 1985.

[5] J.J.Craig, Adaptive Control of Mechanical Manipulators, ADDISON WESLEY, 1988.

[6] J.E.Slotine and W.Li, "On the Adaptive Control of Robot Manipulators." Inter. Jour. of Robotics Research, Vol.6, No.3, pp.49-59, Fall 1987.

[7] M.Kawato, Y.Uno, M.Isobe, R.Suzuki, "Hierarchical Neural Network Model for Voluntary Movement with Application to Robotics", IEEE Control Systems Magazine, pp 8-16, April 1988.

[8] Occam2 toolset User manual, INMOS, 1989.

[9] C.C.Wit, P.Noel, A.Aubin, and B.Brogliato, "Adaptive Friction Compensation in Robot Manipulators: Low Velocities, " Inter. J. Robotics Research, Vol.10, No.3, pp.189-199, June 1991.

[10] TARANSPUTER DEVELOPMENT SYSTEM, Prentice Hall, 1988.

[11] J.Galletly, OCCAM2, Pitman, 1990.

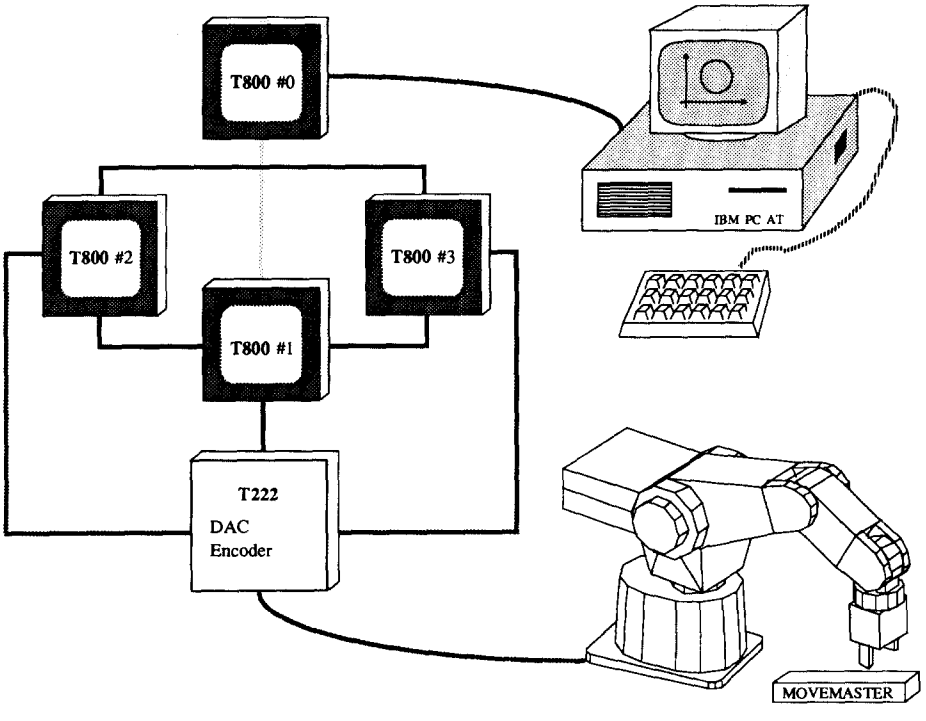

그림 2. 전체 시스템 구성도