트랜스퓨터를 이용한 로보트 매니퓰레이터의 실시간 제어

장 용 근*, 홍 석 교
아주대학교 전자공학과

Real Time Control for Robot Manipulator using Transputer

Yong Geun Jang , Suk Kyo Hong
Dept.of E.E. , Ajou University

ABSTRACT

Many dynamic control have been proposed; however, most of them are limited within stage of simulation study. The main reason is that the computations required for inverse dynamics are far beyond the ability of the present commercially available microprocessors. In this paper, in order to achieve real-time processing in robot dynamic control, a parallel processing computer for robot dynamic control is implemented using two transputer. Two transputer compute two degree of freedom robot. The transputer is a special purpose MPU for parallel processing. Transputers are used in networks to build a high performance concurrent system. A network of transputers and peripheral controllers is constructed using point-to-point communication. To gain most benifit from the transputer architecture, the whole system is programmed in OCCAM which is a high level language for concurrent applications. This control algorithm is applied to the RHINO SCARA type manipulator. We could taked about 438.6 microseconds to compute robot dynamic with two-processors.

1. 서론

현재 산업 현장에서 사용되고 있는 산업용 로보트들은 그 사용되는 목적에 따라 여러 종류로 분류되고 있으며 이들 로보트 중 전자회로 기판의 부품 삽입 작업이나 부품 조립 공장에서의 부품 이송 작업등은 비교적 설치공간이 적고 설치 하기가 용이한 SCARA TYPE형의 ROBOT 를 주로 이용하여 제품을 생산하고 있다.

일반적으로 고속으로 움직이거나 정확한 제어를 위해서는 로보트 매니퓰레이터의 동력학 특성을 고려해야 한다. 그러나 로보트 매니퓰레이터의 동력학 제어에 대한 많은 연구가 제안되었지만 대부분이 시뮬레이션 연구에 국한되었고, 실제 로보트 매니퓰레이터에 설치한 것은 극히 적다. 이것의 주요 원인은 동력학 제어를 위해서는 로보트의 기구상태와 운동상태에서의 각종 기구의 정보들 즉, 전체속도, 각 엑츄에이터의 동력, 각 링크의 운동정도, 각 관절의 속도 및 가속도, 중력 관계, 관성 등 많은 정보들이 수시로 계산되어야 하는데, 이 방대한 계산량에 대해 현재 이용되는 마이크로 프로세서들로는 계산 시간이 길어 많은 불편함과 문제점으로 실시간 제어가 어렵다.[1]

역 동력학(inverse dynamic) 의 많은 계산량을 처리하는 어려움을 극복하기 위해 많은 병렬 알고리즘이 제안되고 있는데, 본 연구에서는 로보트의 동력학을 병렬화하고 트랜스퓨터를 이용하여 실제 계산의 실시간 병렬처리를 위해 Newton-Euler 알고리즘을 변형시킨 Resolved Newton-Euler 알고리즘을 적용하기로 한다.[2][3]

매니퓰레이터의 제어방식은 직교 좌표계에서 매니퓰레이터의 end-effector를 직접 제어할 수 있는 PI제어 알고리즘과 CT(Computed Torque) 방법을 적용, 비교해 보고 이들 제어 알고리즘으로 원하는 경로를 따라 위치 추적오차로 제어 성능을 평가한다. 제어 대상은 RHINO SCARA 형 ROBOT를 사용하고 병렬 처리기와 로보트 사이에 I/O 기판을 제작한다.

2. 매니퓰레이터의 동력학과 병렬처리 방법

보다 효율적인 운동 방정식을 유도하기 위해 Newton의 두번째 법칙을 변화시킨 여러형태의 Newton-Euler 방정식을 개발하였지만 많은 마이크로 프로세서 연결로 bus arbitration 문제와 많은 통신시간으로 실시간 제어에 어려움이 따랐다. 그러다가 최근 하나의 칩 속에 지점간 통신 링크에 의해 연결된 트랜스퓨터의 개발로 로보트의 동특성을 각 축당 제어하는 병렬 처리를 쉽게 할 수 있게 되었다. 처음 병렬 N-E 을 보이고 다음에 변형된 N-E 알고리즘을 보인다.

Newton-Euler는 Newton ($F=m\dot{v}^*$)의 공식과 Euler($N=J\dot{W} + W\times JW$)에 의해 유도 되어진다. 또한 N-E 알고리즘은 forward 알고리즘과 backward 알고리즘 두 부분을 포함하는 재귀적인 형태를 가지고 있다.

Forward Algorithm

$$W_i = \begin{bmatrix} W_{i-1} + z_{i-1}\dot{q}_i \\ W_{i-1} \end{bmatrix} \qquad (2.4)$$

$$\dot{W}_i = \begin{cases} \dot{W}_{i-1} + z_{i-1}\ddot{q}_i + W_{i-1} \times (z_{i-1}\dot{q}_i) \\ \dot{W}_{i-1} \end{cases} \quad (2.5)$$

$$\dot{V}_i = \begin{cases} \dot{W}_i \times P_i^* + W_i \times (W_i \times P_i^*) + \dot{V}_{i-1} & (2.6) \\ z_{i-1}\ddot{q}_i + \dot{W}_i \times P_i^* + 2W_i \times (z_{i-1}\dot{q}_i) & (2.7) \\ \quad + W_i \times (W_i \times \dot{P}_i^*) + V_{i-1} \end{cases}$$

$$\bar{a}_i = \dot{W}_i \times \bar{S}_i + W_i \times (W_i \times \bar{S}_i) + \dot{V}_i \quad (2.8)$$

Backward Algorithm

$$F_i = m_i\bar{a}_i \quad (2.9)$$

$$N_i = I_i\dot{W}_i + W_i \times (I_iW_i) \quad (2.10)$$

$$f_i = F_i + f_{i+1} \quad (2.11)$$

$$n_i = n_{i+1} + P_i^* \times f_{i+1} + (P_i^* + \bar{S}_i) \times F_i + N_i \quad (2.12)$$

$$\tau = \begin{cases} n_i^T z_{i-1} + b_i\dot{q}_i \\ f_i^T z_{i-1} + b_i\dot{q}_i \end{cases} \quad (2.13)$$

병렬처리 구조에서 병렬처리 장치 사이의 통신시간이 무시될 수 있을 만큼 작다면 병렬처리기의 성능은 처리장치의 수에 비례하여 증가하지만 통신 시간이 증가하면 병렬처리기의 성능은 오히려 저하되는 수가 있다. 그러므로 이상적인 병렬화 방법은 전체 알고리즘 계산에 필요한 작업을 계산시 필요로 하는 가장 작은 단위 작업으로 나누어 전체 계산 시간이 가장 최소가 되도록 하는 것이 가장 좋다. 이는 통신시간의 지연이 전혀 발생하지 않기 때문에 병렬처리 장치의 수와 그 성능이 선형적으로 비례한다. 그러나 통신량이 매우 많으므로 실제 구현하였을 경우 각 처리 장치는 통신에 많은 시간이 걸리게 되고 계산시간 보다 정보를 얻기 위해 기다려야 하는 시간이 더 많아 실질적으로 그 성능은 저하된다.

본 연구에서는 계산을 위한 단위 작업을 어느 정도 갖도록 하고, 통신량이 되도록 적게 하여 통신보다 계산을 더 많이 하면서도 병렬처리가 가능한 것이어야 한다.

따라서 하나의 축에 하나의 처리 장치를 대응시켜 1축에서의 계산 결과가 2축에서 필요한 정보만을 전송하면 최소 작업 단위를 사용하는 것에 비해 통신량을 상당히 줄일 수 있다. 등력학 알고리즘을 하나의 축에 하나의 처리 장치를 대응시켜 정보 의존성에 의한 정보흐름과 각 처리 장치에서 계산해야 할 내용을 그림 1에 나타내었다.

그림 1에서 보여지는 방법의 특징은 하나의 축에 하나의 처리 장치를 대응시켜 정보의 의존성을 줄이고 일정한 시간 간격을 주기로 등기를 맞추어 정보를 전송한다는 점이다.

다시말하면 정보를 주고 받을 두개의 통신장치가 모두 대기상태에 있을때만 정보전송이 가능하므로 정보의 손실을 막을 수 있다. 하지만 이같은 시스템에서는 일정한 간격으로 통신이 이루어져야 하므로 각 단위 작업 중 가장 계산시간이 긴 단위 작업에 시간 간격을 맞추어야 하므로 상대적으로 처리 장치 성능이 저하

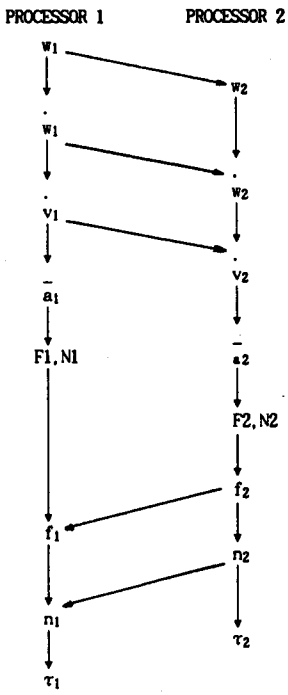

그림 1. 2축의 회전형 조인트의 병렬 등력학

되는 단점이 있다. 다시말하면 처리 장치가 계산을 수행하는 것보다 통신시간이 될때까지 기다려야 하므로 대기 시간이 많아 진다는 것이다. 계산이 끝난 후에 통신을 시작하여 상대방이 준비가 되면 전송이 이루어지는 형태의 방법을 사용하면 앞의 시스템보다는 전체 계산 시간이 더 짧다. 즉 필요없이 기다리는 시간이 줄어들게 되므로 처리 장치 활용지수가 증가한다.

또한, 통신 장치와 계산 장치가 독립적으로 수행되는 시스템에서는 계산 장치가 통신 장치에 정보의 전송을 의뢰한 후에 그 다음 단위 작업의 계산을 수행하도록 하면 통신과 계산의 병렬처리가 가능하며 통신과는 무관하게 계산을 수행할 수 있으므로 더 큰 성능을 발휘할 수 있다.

본 연구에서는 계산에 필요한 단위작업을 정보 의존성이 있는 부분과 없는 부분으로 나누어 실행하고, 미리 계산한 값을 전송하는 알고리즘을 제안하고 거기에 따른 정보흐름과 각 처리 장치에서 계산해야 할 내용을 그림 2에 나타내었다.

Forward Algorithm

$$W_i = z_{i-1}\dot{q}_i \quad (2.14)$$

$$W_i = W_{i-1} + z_{i-1}\dot{q}_i \quad (2.15)$$

$$\dot{W}_i = z_{i-1}\ddot{q}_i + W_{i-1} \times (z_{i-1}\dot{q}_i) \quad (2.16)$$

$$\dot{W}_i = \dot{W}_{i-1} + z_{i-1}\ddot{q}_i + W_{i-1} \times (z_{i-1}\dot{q}_i) \quad (2.17)$$

$$\dot{V}_i = \dot{W}_i \times P_i^* + W_i \times (W_i \times P_i^*) \quad (2.18)$$

$$\dot{V}_i = \dot{W}_i \times P_i^* + W_i \times (W_i \times P_i^*) + \dot{V}_{i-1} \quad (2.19)$$

$$\bar{a}_i = \dot{W}_i \times \bar{S}_i + W_i \times (W_i \times \bar{S}_i) \quad (2.20)$$

$$\bar{a}_i = \dot{W}_i \times \bar{S}_i + W_i \times (W_i \times \bar{S}_i) + \dot{V}_i \quad (2.21)$$

Backward Algorithm

$$F_i = m_i \bar{a}_i \quad (2.22)$$

$$N_i = W_i \times (I_i W_i) \quad (2.23)$$

$$N_i = I_i \dot{W}_i + W_i \times (I_i W_i) \quad (2.24)$$

$$f_i = F_i + f_{i+1} \quad (2.25)$$

$$n_i = P_i^* \times f_{i+1} + (P_i^* + \bar{S}_i) \times F_i \quad (2.26)$$

$$n_i = n_{i+1} + P_i^* \times f_{i+1} + (P_i^* + \bar{S}_i) \times F_i + N_i \quad (2.27)$$

$$\tau = n_i^T z_{i-1} + b_i \dot{q}_i \quad (2.28)$$

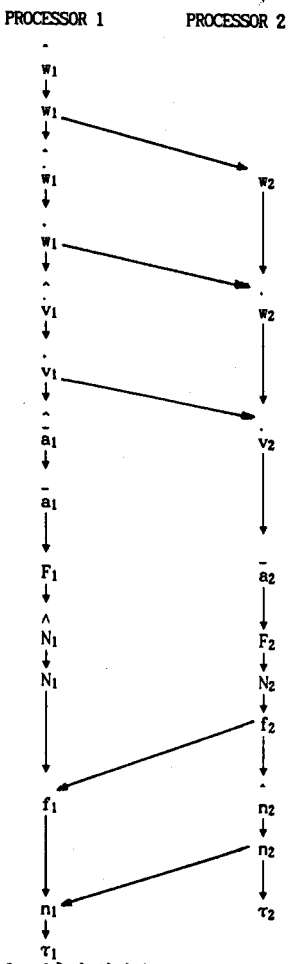

그림 2. 2축의 회전형 조인트의 병렬 동력학

3. 시스템 구성

전체 시스템의 구성은 그림 3과 같다.

본 연구에서 사용된 시스템은 병렬처리를 위한 계산기로 32bit 중앙처리장치, 4Kbyte 의 고속 내장램(On Chip RAM)과 부동점 연산부(Floating Point Unit), 4개의 고속 통신용 링크가 내장된 INMOS사의 T800 트랜스퓨터와 병렬처리에 적합한 프로그래밍 언어인 OCCAM을 사용하고, 호스트 컴퓨터로써 IBM/386 PC가 사용된다. 병렬처리 장치를 구성하기 위해 주기판으로 INMOS의 B008을 사용하며, 로보트와의 인터페이스를 위해 제작한 기판도 이 슬롯 중의 적당한 곳에 꽂아 사용하고, 링크 스위치 C004를 제어하여 링크 어댑터 C011와 연결하여 외부와 인터페이스를 가능하게 한다. 슬롯 0에 있는 트랜스퓨터는 호스터 컴퓨터에서 작성된 프로그램을 컴파일하고 또한 다른 트랜스퓨터 사이의 다리 역활과 프로그램의 수행시 발생하는 상황을 기록하며, 슬롯 1, 2에 있는 트랜스퓨터는 동력학 및 실제 계산값을 링크 어댑터에 보내게 한다.

본 연구의 수행을 위해 B008 보드에 3개의 TRAM을 꽂아 병렬 처리기를 구성하였다. 이것은 통신 장치의 성능이 강력하고 병렬처리 계산기로 아주 적합하다.

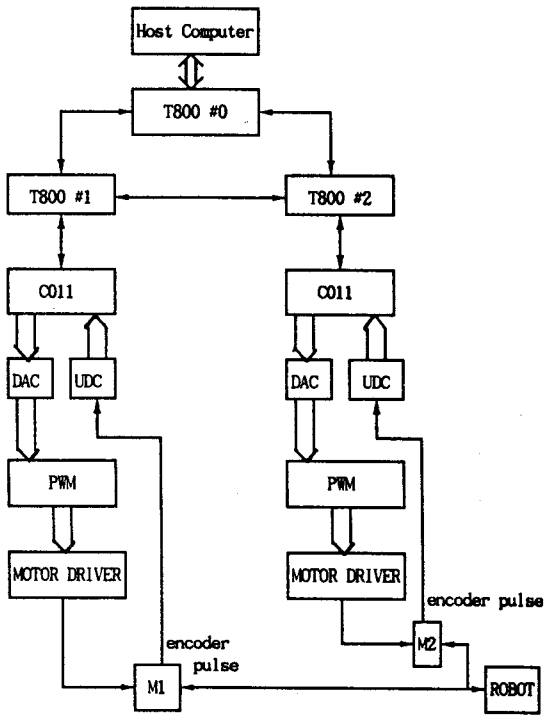

그림 3. 전체 시스템 구성도

표 1에 여러가지 수치 계산시간과 통신을 나타내었다.[6]

Integer Operation Time		Floating point Operation Time		Communication Operation Time	
Add	50	Add	350	Output	1300
Multiply	1950	Multiply	550	Input	1325
Devide	2000	Devidy	850		
PLUS	100	ABS	500	(Byte transfer)	
Procedure	825	SQRT	6300		
		COS	18900		
		EXT	22000		

표 1. T800의 계산시간(n sec)

4. 실험방법 및 결과

이 병렬처리기와 로보트 사이에 I/O를 담당하도록 링크 어댑터 C011를 사용하여 접속기판을 제작하였고 병렬 처리기에서 계산된 토크 명령을 DAC로 출력하여 아날로그 신호로 바꾸어 주고 이를 PWM 입력으로 사용하고 PWM 출력신호로 로보트를 구동하였다. DAC는 SAMSUNG사 KSV3110을 사용하였고, PWM 입력전압인 0 ~ +5 값으로 변환시키고 DAC의 보호를 위해 Op-Amp Max 400를 사용하였다. 로보트의 실제 자유도는 4지만 본 연구에서는 위치 제어만을 위해 2축만 사용하여 실험하였다.

제어 대상 SCARA 형 로보트는 큰 기어비를 가지고 있어 상대적으로 마찰력의 효과가 커진다. 그러므로, 마찰력을 모델링하여 로보트 동력학에 보상하였다.

본 실험에서 엔코드 분해성능 저하로 샘플링시간은 10msec로 하고, 0 ~ 50 도의 위치제어를 CT와 PI 제어를 적용,비교해 보았다. 로보트 매니플레이터와의 인터페이스로 고속 링크어댑터를 사용하여 역동력학 계산시간으로 Hashimotor는 3축 구성으로 464.2 μsec가 걸렸고[2], KAIST Dept. of E.E.에서 실험한 결과는 0.9msec가 걸린 것에 비해 본 연구에서는 2축 구성으로 438.6 μsec가 걸렸다

다음 그림 4,5와 같이 조인터 1, 2의 추종 결과를 보여 준다. 실험 결과는 PI 제어가 CT제어보다 좋은 추종 성능을 보여주는데, 로보트 매니플레이터의 정확한 파라미터 부족으로 CT제어의 많은 오차를 보았다.

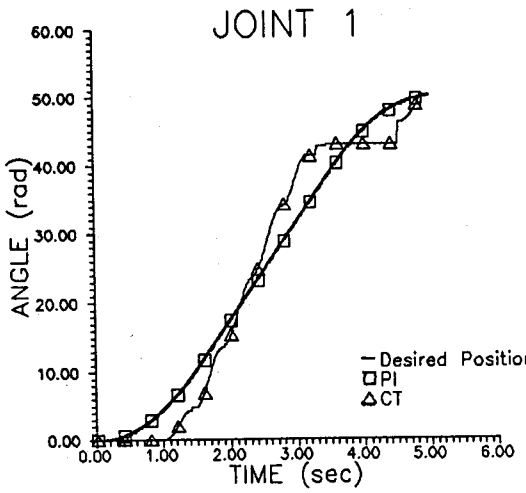

그림 4. 조인트 1의 위치 추종

5. 결 론

T800 트랜스퓨터와 고속 링크어댑터 사용으로 역동력학 계산시간이 433.6 μsec의 빠름을 알 수 있었지만 엔코드 분해능력의 저하 및 정확한 파라미터의 부족으로 많은 오차를 보았고 보다 큰 적용 로보트와 보다 성능이 향상된 로보트에서는 빠른 계산과 통신으로 비선형 특성인 로보트 매니플레이터의 실시간 제어에는 적합하리라 생각된다.

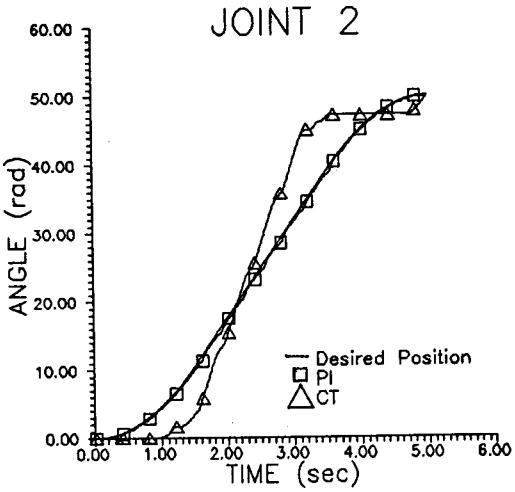

그림 5. 조인트 2의 위치 추종

참고 문헌

[1] Fu, K, S. and Gonzalez, R. C., Lee, C. S. G. 1987 "ROBOTIS : Control, Sensing, Vision, Intelligence" McGraw-Hill Book Co.

[2] Hashimoto, K. and Ohashi, K., Kimura, H. 1990 "An implementation of a parallel algorithm for real-time model-based control on a network of microprocessos", Inter. Jour. of Robotics Research. Vol.6, No.6, pp37-47.

[3] Hashimoto, K. and Kimura, H. 1989. "A parallel algorithm for inverse dynamics", Inter. Jour. of Robotics Research. Vol.8, no.5, pp63-76.

[4] Spong, M. K. and Vidyasagar, M. 1989 "Robot Dynamics and Control", John Wiley & Song, Inc.

[5] Inmos Ltd. 1989. "Transputer databook".

[6] Inmos Ltd. 1988. "Transputer Technical Note", Prentice Hall.

[7] Inmos Ltd. 1988b. "Transputer Development System", Herts, UK : Prentice-Hall.

[8] Inmos Ltd. 1988c. "Transputer Reference Manual", Herts, UK : Prentice-Hall.

[9] Inmos Ltd. 1989. "B008 User Guide and Reference Manual"

[10] J. Galletly, 1990. "OCCAM2", Pitman.