論 文
40~2~9

로보트의 힘 제어를 이용한 윤곽 추적, 삽입및 그라인딩 작업의 구현에 관한 연구

A Study on the Implementation of Edge-Following Insertion and Grinding Tasks Using Robot Force Control

鄭 栽 旭* · 李 範 熙** · 高 明 三***
(Jae-Ook Jeong · Bum-Hee Lee · Myoung-Sam Ko)

요 약

로보트가 정밀한 작업을 수행하기 위해서는 환경의 변화및 불확실성에 적응할 수 있는 센서를 이용한 지능을 갖춘 로보트의 개발이 필수적이다. 본 논문에서는 힘/토크 센서와 2대의 퍼스널 컴퓨터, 그리고 PUMA 560로보트를 사용하여 힘 제어를 구현하고 그를 이용한 여러가지 응용작업들을 실험하였다.

힘 제어 방법으로는 힘 제어축과 위치 제어축을 분리하여 제어할 수있는 복합 위치/힘 제어방식을 사용하였으며 힘/토크 센서 출력값을 컴퓨터에서 읽을 수 있도록 인터페이스 보드를 제작하였다. 또한 두대의 컴퓨터가 많은 양의 정보를 빠르게 교환할 수 있도록 공유기억장치를 제작하였다.

힘 제어를 이용하여 응용작업을 하기 위해서는 먼저 기본적인 힘 명령이 제공되어야 하는데 이를 위하여 이산적인 힘 되먹임을 받는 MOVE-UNTIL명령과 작업도중 계속적인 힘 되먹임을 받는 MOVE-COMPLY명령을 작성및 실험하였으며 위의 두 기본적인 명령들을 이용하여 힘 제어 응용작업으로 윤곽 추적, 삽입및 그라인딩작업의 3가지 작업을 구현및 실험하였다.

Abstract - In the case that the robot manipulator should respond to the variance and uncertainty of the environment in performing precision tasks, it is indispensable that the robot utilizes the various sensors for intelligence. In this paper, the robot force control method is implemented with a force/torque sensor, two personal computers, and a PUMA 560 manipulator for performing the various application tasks. The hybrid position/force control method is used to control the force and position axis separately. An interface board is designed to read the force/torque sensor output into the computer. Since the two computers should exchange the information quickly, a common memory board is designed.

Before the algorithms of application tasks are developed, the basic force commands must be supplied. Thus, the MOVE - UNTIL command is used at the discrete time instant and, the MOVE - COMPLY is used at the continuous time instant for receiving the force feedback information. Using the two basic force commands, three application algorithms are developed and implemented for edge-following, insertion, and grinding tasks.

1. 서 론

공장 자동화에 있어서 로보트의 작업중 윤곽 추적이나 삽입 작업, 조립 작업등과 같은 최근의 작업들은 외부 제한조건(constraint)에 의한 힘과 환경에서의 변화 및 불확실성(uncertainty)에 로보트로 하여금 대응할 것을 요구한다. 로보트에 이러한 능력을 부여하기 위해서 여러가지 센서를 이용한 감지장치를 지닌 로보트의 개발이 활발하게 연구되었다. 본 논문에서는 그중 힘/토크 센서를 사용한 힘 제어 알고리즘을 개발하고 그를 이용한 응용작업들을 구현한다.

로보트 힘 제어는 작업 목적과 경로 계획(trajectory generation), 힘과 위치 되먹임(force and position feedback), 환경에 따른 경로의 수정 등을 포함하고 있다. 그중 로보트가 어떤 방법으로 감지한 힘을 이용하여 위치 명령을 수정하는가 또는 어떻게 로보트가 행할 작업과 요구되는 힘 제어 방법(force control strategy)을 연관시키느냐에 따라 분류해 보면[12], logic branching feedback방법과 continuous feedback방법으로 나눌 수 있다.

Logic branching방법은 접촉과 같은 이산 사건(discrete event)에서 쓰여지는 방법으로 이것을 컴퓨터 알고리즘으로 다음과 같이 나타낼 수 있다.

MOVE ALONG X-AXIS UNTIL FORCE EXCEEDS F

즉 로보트가 움직이는 도중 감지된 힘이 힘 명령보다 크면 멈추게 되는 방법으로 간단한 물체 검출과 작업전의 물체와의 접촉을 위하여 사용한다.

Continuous feedback방법은 6축 힘/토크센서를 이용하여 연속 동작 도중에 로보트 팔과 환경사이의 계속적인 접촉을 유지하도록 하는 방법으로 이 방법을 사용하면 로보트로 하여금 삽입이나 윤곽 추적, 조립작업과 같이 로보트가 물체와 접촉해야만 하는 작업, 즉 움직임의 제한(motion constraint)이 있는 작업을 효과적으로 수행시킬 수 있다.

현재까지 연구된 힘 제어 시스템은 그림 1에 보인 구조로 대부분 설명될 수 있다[12]. 즉 로보트

*正 會 員 : 서울大 大學院 制御計測工學科 博士課程
**正 會 員 : 서울大 工大 制御計測工學科 副教授 · 工博
***正 會 員 : 서울大 工大 制御計測工學科 教授 · 工博
接受日字 : 1990年 5月 24日
1次修正 : 1990年 12月 24日

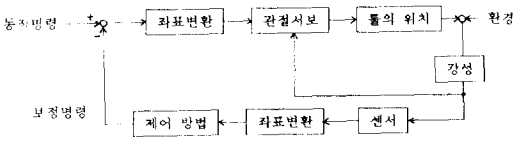


그림 1 힘 제어 시스템의 구조
Fig. 1 The structure of a force control system

가 계획된 위치로 명령을 받고 있을 때 힘 되먹임 원칙(force feedback strategy)에 의하여 위치나 속도가 수정된다. 만일 환경과의 접촉이 일어나면 힘 센서를 통한 힘 되먹임(force feedback)이 생기게 되고 힘 되먹임 원칙(force feedback strategy)에 의하여 현재 위치나 경로를 수정하게 된다.

이를 구현하기 위하여 본 논문에서는 6축의 힘/토크를 이용하여 힘 제어축과 위치제어축을 직교좌표계에서 분리, 제어할 수 있는 복합 위치/힘 제어 방식을 PC에 구현하고 그 알고리즘을 기초로 윤곽 추적, 삽입, 그라인딩의 응용 작업들을 프로그램화하며, 각각의 작업에 따른 실험결과를 보인다.

2. 힘 제어 시스템의 구성 및 힘 제어 명령의 구조

2.1 힘 제어 시스템의 구성

그림 2는 힘 제어를 위한 전체 시스템의 개요이다.

전체 시스템은 두 대의 PC와 공유기억장치 보드 및 힘/토크 센서 인터페이스 보드 등으로 이루어져 있다. PC 1은 PUMA로보트 제어기의 LSI-11을 대치하여[7], 직선경로 계산, 월드 모델링 및 PUMA 위치 서보 제어기에 위치 명령을 보내는 등 전체적인 동작을 담당하고 있으며 힘 제어가 필요할 때에는 PC 2에 명령을 내려 PC 2로 하여금 힘 제어를 실행시키고, 힘 제어의 결과를 다시 받아 로보트의 움직임을 보정한다. 이때 PC 1과 PC 2는 힘 제어 알고리즘을 위하여 많은 양의 데이타 교환을 필요로 하므로 이를 위하여 한 기억장소를 두 PC가 참조할 수 있는 공유기억장치 보드를 제작하여 이를 통한 데이타 교환을 할 수 있도록 하며, 힘/토크 센서 인터페이스 보드에서는 힘/토크 센서 전자박스에서 처리된 6축의 힘/토크 값을 병렬 포트를 통하여 PC2에서 읽을 수

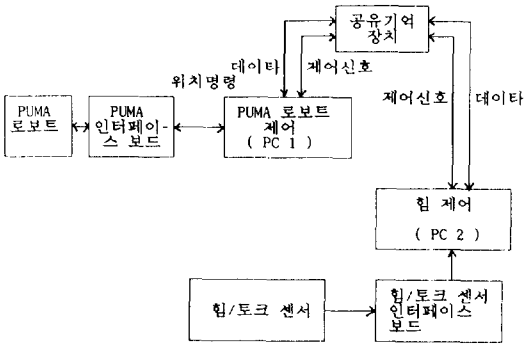
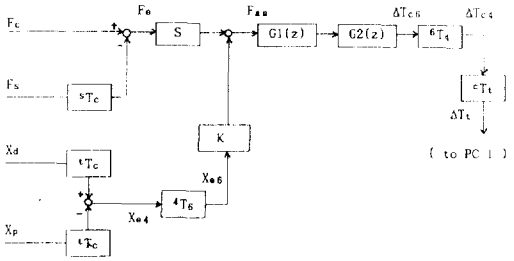


그림 2 전체 시스템의 블럭 다이어그램
Fig. 2 The block diagram of total system



F_c : Force command (6×1 vector)
F_s : Sensed force (6×1 vector)
X_d : Desired position (4×4 matrix)
X_p : Present position (4×4 matrix)
${}^tT_c$: Transform from tool frame to compliant frame
${}^cT_t$: Transform from compliant frame to tool frame
${}^sT_c$: Transform from sensor frame to compliant frame
${}^4T_6$: Transform 4×4 matrix to 6×1 vector
${}^6T_4$: Transform 6×1 vector to 4×4 matrix
S : Selection matrix
K : Stiffness matrix
$G_1(z)$, $G_2(z)$: S/W compensator

그림 3(a) 힘 제어 알고리즘
Fig. 3(a) Force control algorithm

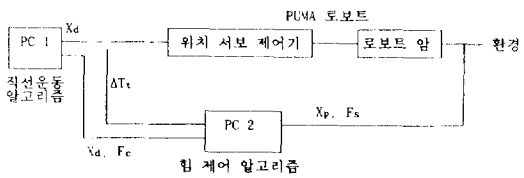


그림 3(b) 힘 제어 알고리즘
Fig. 3(b) Force control algorithm

있도록 하고 있다.

힘 제어 알고리즘의 개요를 그림 3에 나타냈다. 본 연구에서 사용한 힘 제어 알고리즘은 위치 서보 제어를 이용한 직교 좌표계에서의 복합 위치/힘 제어(hybrid position/force control) 방식으로 Maples와 Becker가 제시한 방식[1]의 변형이라고 할 수 있다. 이 방법은 Raibert와 Craig[2]의 복합 위치/힘 제어 방식이나 Salisbury[14]의 강성 제어 방식에 비해 다음과 같은 장점을 갖고 있다.

· 복합 위치/힘 제어 방식은 강성 제어 방식과 달리 힘 제어 축과 위치 제어 축을 분리할 수 있기 때문에 강성(stiffness)과 유연성(compliance)을 각 축에 적절하게 조합하여 다양한 응용이 가능하다.

· 직교 좌표계 시스템에서는 사용자가 임의로 보정운동이 행해지는 좌표계(compliant frame)를 지정해 줄 수 있고 힘 명령을 보정 좌표계에서 줄 수 있어, 관절 좌표계 시스템보다 힘 제어 명령을 지정하는 것이 쉽다.

· Raibert와 Craig방식과는 달리 직선 운동 알고리즘에 의해 계산된 위치 명령을 힘 제어 알고리즘으로 하여금 보정하는 방식이므로 힘 제어 알고리즘을 S/W로 구현하여 PUMA로보트의 PI위치 서보 제어기를 개조하지 않아도 된다.

본 논문에서 지금부터 얘기하는 4×4행렬은 월드 좌표계에서의 변환 행렬이고 6×1벡터는 위치에 관계되는 경우에는 x, y, z좌표에서의 이동값과 변환각을, 힘에 관계되는 경우에는 x, y, z좌표에서의 힘과 토크값을 갖는다. PC 2에서 행하는 힘 제어 알고리즘은 다음과 같이 수행된다. PC 1으로부터 읽은 현재 위치(X_p)와 목적 위치(X_d)를 식 (1)에 의해 보정 좌표계로 변환시킨 다음 위치 오차(X_{e4})를 식 (2)에 의하여 계산한다.

$$X_{dc}=X_d\cdot{}^tT_c$$
$$X_{pc}=X_p\cdot{}^tT_c \tag{1}$$
$$X_{d4}=(X_{pc})^{-1}\cdot X_{dc} \tag{2}$$

계산된 오차(X_{e4})는 4×4행렬이나, 감지된 힘과 명령 힘은 6×1벡터이므로 4×4행렬을 6×1벡터로 변환한다. 6×1벡터로 변환된 위치 오차 X_{e6}는 강성(stiffness) 행렬을 거쳐 예상 힘으로 계산된다. 이 강성값은 사용자가 임의로 정해줄 수 있는 값으로 예를 들면 오차가 10cm일때 강성이 0.1N/cm이라면 예상 힘은 $F=kx$라는 식에 따라 1N이 된다. 로보트는 예상 힘을 감하는 방향으로 제어되기 때문에 만일 강성이 0이라면 로보트는 최대

한의 유연성(compliance)를 갖고 작업하게 된다. 즉 강성행렬은 로보트의 유연성을 조절하게 되며, 강성이 크면 클수록 로보트의 움직임은 예정된 경로를 따라가려 하므로 디버링(deburring), 그라인딩(grinding)등의 작업에 적합하며, 강성이 작을수록 로보트는 보정동작을 민감하게 하므로 삽입이나 윤곽 추적 등에 적합하다.

명령 힘과 감지된 힘의 오차를 계산하기 위해 센서 좌표계에서의 감지된 힘(F_s)을 식 (3)에 의해 보정 좌표계로 변환시키고, 힘 오차 F_e를 식 (4)에 의해 계산한다.

$$F_{sc}=F_s\cdot {}^sT_c \tag{3}$$

$$F_e=F_c-F_{sc} \tag{4}$$

힘 오차 F_e는 선택(selection) 행렬을 통과함으로써 특정 축만 힘 제어를 할 수 있게 된다. 선택행렬은 0과 1값만을 가지는 행렬로 힘 제어를 하는 축은 1, 위치 제어를 하는 축은 0이 된다.

예상 힘과 힘 오차를 더하여 실제 힘 오차(actual force error, F_{ae})를 계산하고 그 값은 제어기(compensator)로 들어가게 된다.

$$F_{ae}=F_e\cdot S+K\cdot X_{e6} \tag{5}$$

제어기는 $(z-b)/(z-a)$로 표시되는 소프트웨어 제어기이다. 이때 제어기는 힘 제어 루프의 차수를 결정하게 되는데 제어기의 이득(gain)이 크면 로보트는 작은 힘에도 민감한 반응을 나타내게 된다.

사용한 제어기의 식은 $z/(z-1)$로 적분기(integrator)로써 사용되고 있으며, 이 제어기가 힘 오차값을 적분함에 따라 로보트는 실제 힘 오차에 비례하는 속도로 움직이게 된다. 제어기를 이중적분기로 쓰게 되면 로보트는 가해진 힘에 비례하는 가속도로 움직이게 되며, 이 구조는 사람이 손으로 로보트를 교시(teach)하는 등의 빠른 응답을 필요로 할 때에 유용하다.

제어기를 거쳐 계산된 제어 명령은 6×1벡터, (ΔT_{c6})이므로 이 값은 다시 4×4행렬(ΔT_{c4})로 바꿔주고 식 (6)에 의하여 로보트의 툴 좌표계로 변환시킨 다음 이 값을 PC 1에 위치 보정 명령(ΔT_t)으로 보내 주게 된다.

$$\Delta T_t={}^tT_c\cdot\Delta T_{c4}\cdot {}^tT_c^{-1} \tag{6}$$

보정 동작은 다음의 목표 위치를 X_d, 보정값을 ΔT_t라 할 때 식 (7)에 의해서 이루어지며 이 과정은 PUMA로보트의 매 샘플링 시간(28ms)마다 반복된다.

$$X_d\Leftarrow X_d\cdot\Delta T_t \tag{7}$$

2.2 힘 명령의 작성 및 실험 결과

힘 제어 응용 프로그램을 위하여 가장 기초적으로 제공되어야 하는 것이 1장에서 얘기한 continuous feedback방법과 logic branching방법의 실현을 위한 move-until과 move-comply이다.

로보트가 움직이는 도중 물체와 접촉시 로보트의 동작을 중단하기 위해 쓰여지는 것이 move-until이다. 이 명령은 사용자가 정한 힘 제어축과 힘 명령을 받아 움직이다가 감지된 힘이 힘 명령보다 커지면 멈추게 된다. 예를 들어 힘 제어축을 X로, 힘명령을 10uf로 지정하고 move-until을 수행하게 되면 로보트가 현재 위치에서 X축방향으로 움직이다가 감지된 힘(F_x)이 힘 명령(F_c)보다 커지게 되면 끝나게 된다. 이때 움직이는 속도는 힘 명령과 힘 이득을 곱한 값에 비례하게 된다.

로보트가 물체와 접촉을 한 이후에 여러가지 작업을 하기 위해서는 힘 센서의 계속적인 되먹임(continuous force feedback)을 받으면서 동작을 하게 되는 데 이것을 위한 함수가 move-comply이다. Move-comply는 목표위치를 입력하면 현재 위치에서 목표위치까지의 직선 경로를 생성하게 된다(직선 보간 운동 알고리즘). 그리고 힘 제어축과 힘 명령을 지정함에 따라 현재 위치에서 목표 위치까지 움직이는 동안 힘 제어 축에서 감지된 힘을 힘 명령으로 유지하기 위하여 보정 동작을 하게 된다. 보정 동작은 사용자가 정한 힘 제어 축과 힘 명령 및 보정 좌표계의 위치를 받아 미리 계산된 경로를 따라 움직이면서 보정좌표계의 힘 제어축의 힘을 힘 명령에 따라 조정하게 된다.

응용 작업의 프로그램은 위 두 명령(move-until, move-comply)을 이용하여 작성하게 되며, 작업의 특성에 따라 동작이 끝나는 조건에 변형을 가하거나 힘 제어 알고리즘에 약간의 추가를 하게

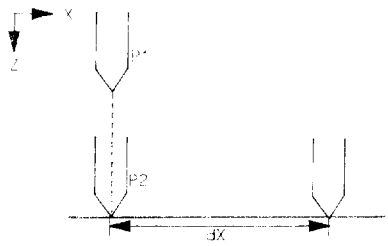


그림 4 힘 명령의 실험 구성
Fig. 4 The experimental layout of force commands

된다. 위의 명령들을 실험하기 위한 구성을 그림 4에 나타내었다.

로보트를 준비위치(P1)에 이동시켜 놓고 move -until을 Z축에 대하여 수행하여 툴의 끝이 물체 표면에 접촉하도록 하고 접촉위치에서 X축으로 dX만큼의 위치까지 힘제어축을 Z축으로, 힘 명령을 −30uf로 move-comply를 수행한 실험결과를 다음 그림에 보인다. 이때 그림의 X축의 단위 1은 힘 제어 루프를 한번 실행하는 시간으로 실제로는 약 10msec정도이며 Y축은 감지된 힘의 양으로 단위 1(1 unit force)당 약 0.03Kg-f이다.

Move-comply명령의 실험 결과에서 보면 로보트가 물체와 접촉하면서 움직이는 도중 물체 표면에 힘 명령에 따른 일정한 힘을 가하는 것을 볼 수 있다. 특히 힘제어 루프 이득이 커짐에 따라 더욱 빨리 힘 명령을 따라가는 것을 볼 수 있다.

3. 응용 작업의 구현 및 실험

3.1 윤곽 추적

윤곽 추적의 목적은 임의의 물체의 형태를 알아내고, 또 추적도중 물체에 일정한 힘을 가하는 것이다. 이를 위하여 힘 제어 알고리즘내에 보정 좌표계에서의 위치 제어 부분과 윤곽 추적이 끝났음을 감지하는 부분을 추가하여 윤곽 추적 알고리즘을 구현한다.

윤관 추적시에는 추적도중 도구(tool)의 방향(orientation)이 일정하게 유지될 수도 있고[3][6] 물체 표면에 일정한 방향을 유지하기 위해

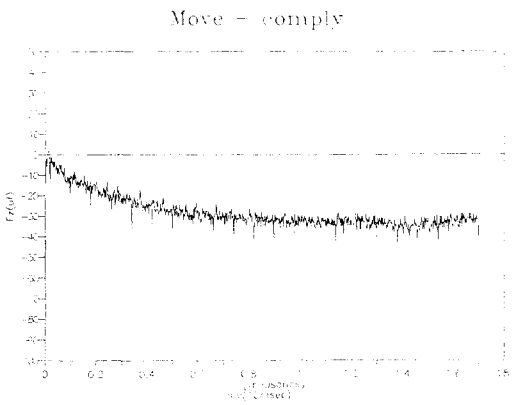


그림 5 힘 이득이 0.002일때의 move-comply
Fig. 5 Move-comply with force gain=0.002

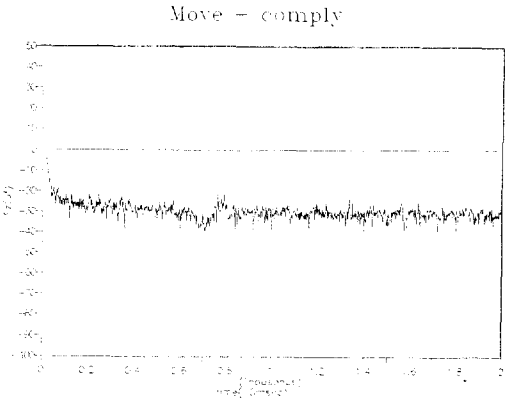


그림 6 힘 이득이 0.005일때의 move-comply
Fig. 6 Move-comply with force gain=0.005

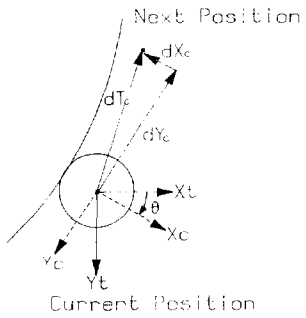


그림 7 윤곽 추적 작업의 구성
Fig. 7 The layout of edge-following task

도구(tool)를 돌릴 수도 있다. 본 논문에서는 전자의 방법을 사용하였으며 이때 물체 표면에 일정한 힘을 가하기 위해 가상의 보정 좌표계(compliant frame)를 지정하여 이 좌표계에서 물체 표면에 접한 방향으로 위치 제어를 하고, 직각 방향으로 힘 제어를 함으로써, 물체 표면에 가해지는 힘과 추적 속도를 일정하게 유지하도록 하고 있다. 그림 7에 윤곽 추적을 할 때의 보정 좌표계와 툴 좌표계와의 관계를 나타내었다.

그림 7에서 보면 보정 좌표계의 X축에서 힘 제어, Y축 방향으로 위치 제어를 하게되는데, 보정 좌표계는 툴 좌표계와 보정 좌표계사이의 변환 행렬과 보정 좌표계와 센서 좌표계 사이의 변환 행렬을 계산함으로써 이루어진다. 이때 2가지의 가정을 한다.

1. 물체의 추적은 X-Y plane, 즉 2차원 평면에서만 한다.

2. 툴 좌표계와 센서 좌표계의 방향(orientation)은 같다.

$$\theta = \arctan\left(\frac{F_y}{F_x}\right) \tag{8}$$

그러면, 툴 좌표계와 보정 좌표계와의 각도는 식 (8)과 같다.
이 각을 이용하여 툴 좌표계와 보정 좌표계와의 변환 행렬(^tT_c)및 보정 좌표계와 센서 좌표계와의 변환 행렬(^cT_s)을 계산한다. 즉,

$$^tT_c \Leftarrow {}^tT_c \cdot \mathrm{Rot}(Z, \theta) \tag{9}$$

$$^cT_s \Leftarrow \mathrm{Rot}(Z, \theta) \cdot {}^cT_s \tag{10}$$

이 변환 행렬들을 이용하여 감지된 힘(F_s)과 위치 오차(X_e)를 보정 좌표계로 변환시키고 3장에서의 힘 제어 알고리즘을 적용하면 다음 위치의 보정값(ΔT_c)을 계산할 수 있다. Y축에서의 위치제어는 정해진 속도(sp)를 다음 식에 의해 ΔT_c에 더해줌으로써 이루어진다.

$$\Delta T_c \Leftarrow \Delta T_c \cdot \mathrm{Trans}(Y, sp) \tag{11}$$

위의 계산된 보정량(ΔT_c)를 식 (6)에 의하여 툴 좌표계로 변환하여 최종적인 툴 좌표계에서의 보정값(ΔT_t)을 위치 제어 PC로 넘겨주게 된다.

윤곽 추적의 동작은 현재 위치(X_p)가 초기 위치(X_{ini})에 돌아왔을 때 끝나게 되며 그 조건은 다음과 같다.

$$|X_p - X_{ini}| < d \tag{12}$$

이때 d는 초기 위치와 동작이 끝날 위치의 최대 오차 허용치이며, 이는 힘이득과 추적 속도 및 힘 오차에 관계된다. 그러나, 실제 힘 오차가 어떤

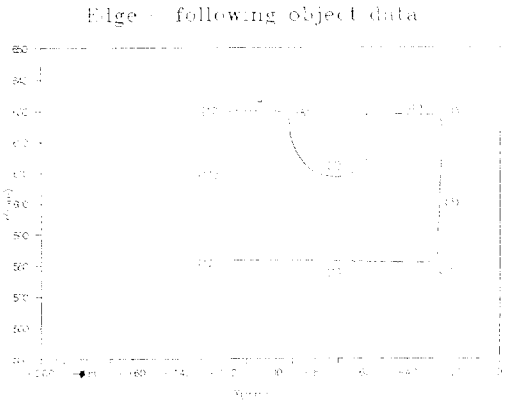


그림 8 추적 물체의 윤곽
Fig. 8 The edge of tracking object

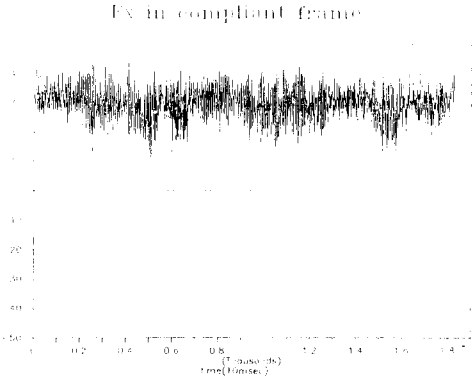


그림 9 물체 추적시 보정 좌표계의 F_x
Fig. 9 The F_x in compliant frame during edge-following

값을 가질지는 예측이 불가능하므로 본 논문에서는 d를 PUMA로보트가 위치제어기의 샘플링시간(28msec)이내에 갈 수 있는 최대거리로 지정하여 추적도중 현재 위치와 초기 위치와의 거리차가 d보다 작으면 동작을 끝내도록 한다.

위의 실험결과를 다음에 보인다. 추적 물체의 모양은 그림 8과 같고 시작점(P1)에서 시작하여 물체의 표면을 따라가다가 다시 시작위치에 오게 되면 추적을 끝낸다. 그림 9는 그림 8의 물체를 추적도중 보정 좌표계에서의 F_x를 나타내고 있다. 이때의힘 명령은 30uf이고 추적 도중의 힘 응답은 30±10uf로 실제 오차가 300gf-cm를 넘지 않음을 알 수 있다. 전체적으로 볼 때 본 논문의 윤곽 추적 알고리즘은 물체의 모양에 관계없이 윤곽 추적을 행할 수 있었으며, 또한 물체와의 접촉면에서의 힘(contact force)도 힘 명령에 따른 일정한 값을 유지하였다.

3.2 삽입(Insertion)

삽입 작업은 힘 제어의 응용으로 가장 많이 연구되는 예지만, 펙과 홀의 모양, 챔퍼의 유무, 그리고 펙과 홀사이의 정밀도에 따라 삽입 알고리즘이 크게 달라질 수 있으므로, 어떤 일반적인 알고리즘을 작성한다는 것은 어려움이 있다.

삽입 작업의 상태는 그림 10과 같이 taper-crossing상태와 side-contact상태로 나눌 수 있는데[4], 펙을 홀에 삽입을 시작하면 위치 오차가 없는 경우를 제외하고는 taper-crossing상태에서 시작하게 된다. Taper-crossing상태의 목적은 위치 오차의 보정으로 펙이 홀의 중심에 나아가는

것이다. Taper-crossing상태가 끝나게 되면 side-contact상태가 시작되며, 이 상태에서의 목적은 펙과 홀을 정렬시키는, 즉 각 오차를 보정하는 것이다.

본 논문에서의 삽입 알고리즘은 위의 두 상태를 구분하지 않고 move-comply명령을 이용하여 해결하고 있다. 이로 인하여 처음 taper-crossing상태에서 각 오차가 없어도 챔퍼에 의한 토크로 인하여 각 보정을 할 수 있으나, 이는 곧 side-contact상태로 넘어감에 따라 없어지게 된다.

본 논문에서 삽입 작업의 실험시 사용한 펙은 끝이 둥글게 깎여 있으며 펙과 홀의지름은 약 1 mm의 차이를 갖고 있다. 펙을 둥글게 깎은 것은 홀에 챔퍼가 있는 것과 같은 효과를 갖고 있다고 볼 수 있다. 본 논문에서 사용한 알고리즘은 원통형의 펙에 적용될 수도 있으며 단 펙이나 홀의 한 쪽에는 챔퍼가 있어야 한다. 즉 본 논문에서 사용한 삽입 알고리즘이 성공적으로 수행되기 위해서는 다음과 같은 조건을 만족해야 한다.

1. 펙이나 홀의 한 쪽에는 챔퍼가 있어야 한다. 만일 챔퍼가 없는 경우에는 반드시 side-contact상태부터 시작해야 한다.
2. 챔퍼가 있는 경우에는 삽입 작업이 taper-crossing상태부터 시작되어야 하므로, 로보트의 위치 오차가 챔퍼의 허용범위내에 있어야 한다.

$$|X_e| < d_c \tag{13}$$

X_e : 위치오차

d_c : 챔퍼의 허용범위

삽입을 실험하기 위해서는 보정 좌표계를 지정하고, 힘 제어 축과 힘 제어 명령을 지정해 주어야 한다. 보정 좌표계는 펙의 끝에 설정하는데 이는 툴 좌표계에서 Z축방향으로 펙의 길이(d_p)만큼을 더해주면 된다.

$$^tT_c = \mathrm{Trans}(Z,\ d_p) \tag{14}$$

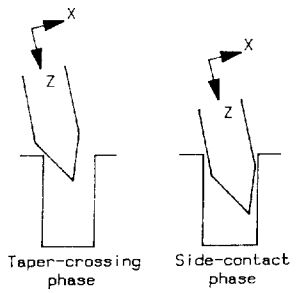


그림 10 삽입 작업의 구성
Fig. 10 The layout of insertion task

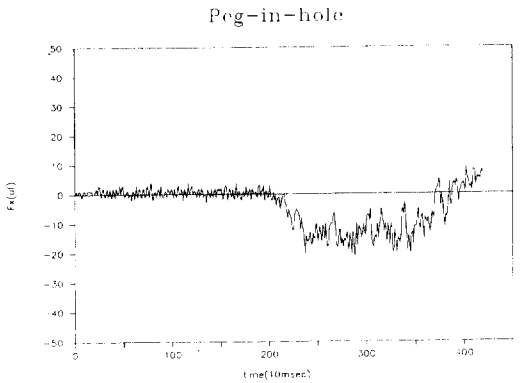


그림 11 삽입 작업시 F_x
Fig. 11 The F_x during insertion

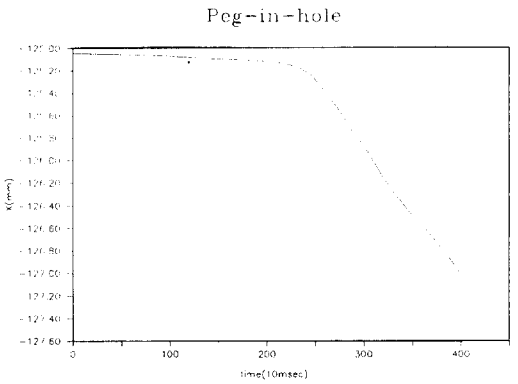


그림 12 삽입 작업시 X축의 좌표
Fig. 12 The X-axis coordinate during insertion

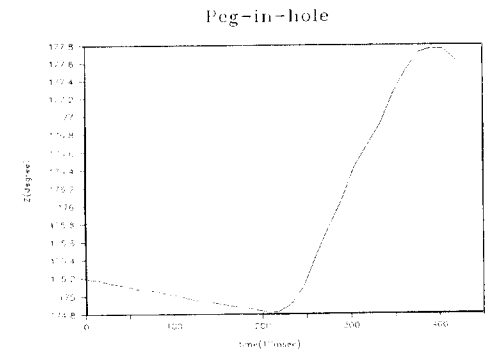


그림 13 툴 좌표계와 월드 좌표계사이의 각도
Fig. 13 The dergree between tool-frame and world-frame

힘 제어 축은 Z축을 제외한 모든 축으로 하며 힘 명령은 모두 0으로 지정한다. 삽입 작업이 끝나는 조건은 Z축의 힘을 계속 감지하면서 Z축의 힘(F_z)이 한계값(Flimit)을 넘는 것으로 이때 펙이 홀의 바닥에 닿은 것으로 인식하게 되어 동작을 멈추게 된다.

정지 조건 : $F_z > F_{limit}$ (15)

이때 한계값은 작업의 구성, 펙과 홀의 재질에 따라 크게 달라지는데 본 논문에서는 센서가 견딜 수 있는 최대값의 반으로 설정하여 대부분의 경우에 무리없이 작업을 끝낼 수 있도록 한다.

실험은 X축과 Y축방향으로 약 3mm정도의 위치 오차와 5도 정도의 각 오차를 두고 한다. 그림 11은 F_x의 변화를 나타내고 있는데, 펙이 홀에 접촉하면서 생기는 힘을 위치 보정을 함으로써 상쇄해나가는 것을 볼 수 있으며 그림 12는 펙의 X축 방향으로 펙이 위치를 보정해 나가고 있는 것을 볼 수 있다. 그림 13은 툴의 Z축과 월드 좌표계의 Z축사이의 각도를 나타내고 있으며, 펙이 홀에 삽입되면서 각 오차를 보정해 나가는 것을 볼 수 있다.

3.3 그라인딩

그라인딩 작업은 물체의 가공을 위하여 사용되며, 주로 물체의 표면을 다듬거나 깎아낼 때에 쓰여진다[20]. 그라인딩 작업의 구성은 크게 두 가지로 나눌 수 있는데 첫번째는 로보트가 툴을 쥐고 바닥에 고정되어 있는 물체를 가공하는 것이며, 두번째는 로보트가 물체를 쥐고 바닥에 고정되어 있는 툴을 이용하여 물체를 가공하는 것이다.

본 논문에서는 후자의 방법을 택하여 로보트에 분필을 쥐게 하고 고정된 툴을 이용하여 이 분필의 끝을 둥글게 깎아본다. 이를 위한 작업의 구성이 그림 14에 나타나있다.

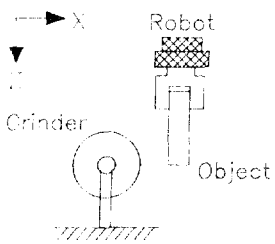


그림 14 그라인딩 작업의 구성
Fig. 14 The layout of grinding task

위 그림에서 X축으로 그라인더에 다가가서 분필을 깎게 되는데, 이를 위하여 Move-until의 알고리즘을 약간 고쳐 그라인딩 알고리즘으로 이용한다. Move-until알고리즘은 특정 축에서 힘 제어를 하며 물체에 접근하다가 접촉힘이 한계값을 넘으면 멈추게 되는데, 이때 멈추는 조건을 그라인딩이 완료되었을 때를 감지하는 것으로 바꾼다. 그라인딩이 완료되었을 조건은 분필을 깎기 시작했을 때의 위치(X_i)와 현재 위치(X_p)의 차가 분필의 직경의 반이 되었을 때로 한다.

$$|X_i - X_p| > \frac{d}{2} \tag{16}$$

분필을 깎기 시작하는 순간은 X축으로 감지된 힘(F_x)이 힘 명령(F_c)과 같은 순간, 즉 $F_x = F_c$의 조건이 만족되는 순간이다. 이 때의 위치를 X_i로 기억시키고 식 (16)을 이용하면 한쪽면에서의 그라인딩이 완료되었는지를 알 수 있다.

분필을 둥글게 깎기 위해선 분필을 회전시켜야 하지만 그라인더와 분필이 닿아있는 상태에서 분필을 회전시킬 수 없으므로 로보트를 현재위치(X_p)에서 월드 좌표계의 X축 방향으로 -10mm 떨어진 위치로 이동시킨다음 분필을 회전시킨다. 식 (17)이 이상의 동작이며, 그에 따라서 보정좌표계의 변환행렬(^tT_c, ^cT_s)도 식 (18)(19)에 의해 회전시킨다.

$$X_c = \mathrm{Trans}(Y, -10) \cdot X_p \cdot \mathrm{Rot}(Z, \theta) \tag{17}$$

Move to X_c in cartesian frame

$$^tT_c \Leftarrow {}^tT_c \cdot \mathrm{Rot}(Z, \theta) \tag{18}$$

$$^cT_s \Leftarrow \mathrm{Rot}(Z, \theta) \cdot {}^cT_s \tag{19}$$

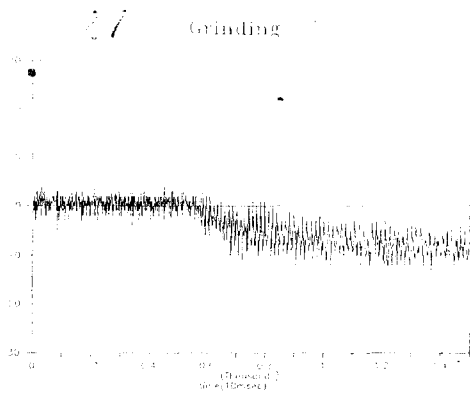


그림 15 그라인딩 작업시 F_x($F_g = 0.005$)
Fig. 15 The F_x during grinding($F_g = 0.005$)

한 번에 회전하는 각(식 (17) (18) (19)의 θ)은 사용자가 임의로 정해줄 수 있으나 작게 할 수록 분필의 깎는 정도가 세밀해진다. 본 논문에서는 힘 이득을 0.005로 하여 실험하며 그에 대한 결과를 그림 15에 나타냈다. 힘 명령은 −10uf로 지정하였는데 접촉힘이 −30uf정도만 되어도 분필은 부러지므로 정확한 힘 제어를 하고 있다고 볼수 있다.

4. 결 론

현재 산업용 로보트는 그 응용에 있어서 미리 프로그램되어 있거나 사람의 가르침에 의한 매우 단순한 작업들만 처리하고 있다. 그것은 센서를 이용한 로보트의 환경에 대한 반응을 할 수 있는 능력이 주어져 있지 않기 때문이며 이는 로보트의 다양한 응용을 불가능하게 하고 있다.

본 논문에서는 힘/토크 센서를 사용한 힘 제어 알고리즘을 PC와 PUMA로보트에 구현하고 힘 제어의 기본적인 명령및 응용작업 프로그램을 작성 및 실험하였다. 힘 제어 알고리즘은 복합 위치/힘 제어 방식을 사용하였으며 이를 운용하기 위하여 작성한 명령은 logic branching방법을 구현한 move-until과 continuous feedback방법을 구현한 move-comply로, 이의 알고리즘과 실험 결과를 보였다. 이 기본적인 명령들을 세가지의 힘 제어 응용작업을 실현하는데 이용함으로써 본 논문에서 구현한 힘 제어 알고리즘과 힘 제어 명령이 다양한 응용이 가능하고 쉽게 재구성(reconfiguration)이 가능하며 거의 모든 힘 제어 응용 작업에 필요한 기초적인 명령들임을 알 수 있었다.

결론적으로 본 연구에서 다룬 힘—토크 센서를 이용한 힘 제어 실현과 그 응용 프로그램의 작성 및 실험은 로보트로 하여금 위치오차와 작업 환경의 부정확성을 자체적으로 극복하여 물체의 검출 및 그 밖의 정밀을 요하는 복잡한 작업들의 수행이 가능함을 확인하였다.

참 고 문 헌

[1] J.A. Maples & J.J. Becker, "Experiments In Force Control of Robotic Manipulator", IEEE Int. Conf. on Robotics and Automation, 1986.

[2] M.H. Raibert, J.J. Craig, "Hybrid Position/Force Control of Manipulators," Tran. ASME, Vol. 102, June 1981.

[3] G.P. Starr. "Edge Following With A PUMA 560 Manipulator Using VAL-Ⅱ", 1986 IEEE Int. Conf. on Robotics and Automation.

[4] J.S. Lee, "A Shared Position/Force Control Methodology For Teleoperation, RCA corporation, Advanced Technology Laboratories.

[5] S. Lee & J.M. Lee, "The Control of Surface Contact and Slide Using Wrist Force/Torque Sensor", Int. Symposium on Robotics and Automationh, 1987.

[6] M.H. Choi, "A Study On The Two Dimensional Object Controur Tracking With A Sensor Equipped Manipulator", M.S. Thesis, Dept of Control & Insatrumentation, Seoul Nat'l Univ., 1988.

[7] 이원석, "컨베이어 벨트를 포함하는 로보트 작업 시스템의 구성에 관한 연구", 서울대학교 제어계측과 석사 학위 논문, 1989.

[8] H.V. Brussel, J. Simons, & J. Deschutter, "An Intelligent Force Controlled Robot," Annals of the CIRP, Vol. 31, Jan. 1982.

[9] B.E. Shimano, C.C. Geschke, C.H. Spalding Ⅲ, R. Goldman, D.W. Scarborough, "AIM : a task level control system for assembly," Proc. 8th Int. Conf. Assembly Automation, March 1987.

[10] T. Lozano-Perez, "Robot Programming," Proc. IEEE, Vol. 17, July 1983.

[11] R.P. Paul & B.E. Shimano, "Compliance & Control," Proc of Joint Automatic Control Conf., pp. 649~699, 1976.

[12] D.E. Whitney, "Force Feedback of Manipulator Fine Motions", Joint Automatic Control Conference, San Francisco, 1976.

[13] J. De Schutter, "Improved Force Control Laws For Advanced Tracking Applications," IEEE Int. Conf. on Robotics and Automation, 1988.

[14] J.K. Salisbury, "Active Stiffness Control of A Manipulator In Cartesian Coordinates", Proc. 19th IEEE Conf. on Decision and Contr, Dec 1980.

[15] R.P. Paul, "Manipulator Cartesian Path Control", IEEE Trans. on Systems, Man, and Cybernetics, Vol. SMC-9, No. 11, Nov 1979.

[16] C. Reboulet, A. Robert, H. Poilve, & A. Gailet, "Hybrid Position Force Control-

Application to Assembly", 15th ISIR.

[17] H. Asada & K. Ogawa, "On The Dynamic Analysis of A Manipulator And Its End Effector Interacting With The Environment", IEEE Int. Conf. on Robotics and Automation, 1987.

[18] H. Kazerooni & J. I. Baecker, "Hybrid Force/Position Control In Robotic Deburring", Conf. on Applied Motion Control, 1987.