움직이는 원통형 물체를 잡는 매니퓰레이터를 위한 레이저 거리계 기반의 서보시스템

LRF-Based Servo System for a Manipulator Grasping Moving Cylinders

천 홍 석*, 김 병 국
(Hongseok Cheon and Byung Kook Kim)

Abstract : We implemented a real-time servo system for a manipulator based on Laser Range Finder (LRF). and established algorithms for grasping a moving cylinder. We devised a manipulator mechanism and driving hardware based on a system board equipped with Xscale Processor with real-time operating system RTAI on Linux. The manipulator motor driver is connected to the system board via CAN communication link, and LRF is connected via RS-232C. We implemented real-time software including CAN device driver, RS-232C device driver, manipulator trajectory generator, and LRF control software. A typical application experiment for grasping a cylinder with circle motion demonstrated our system's real-time performance.

Keywords : LRF-based servo system, manipulator control system, real-time control architecture

I. 서론

로봇에 있어 센서는 인간의 시각, 촉각, 청각기관 등과 같은 감각기관이라 할 수 있다. 이중 가장 많은 정보를 획득할 수 있는 시각기관의 역할을 하는 센서를 이용한 로봇 개발이 활발한데, 비전(vision) 센서와 레이저 거리계(LRF, Laser Range Finder)가 대표적인 시각기관 역할을 하는 센서 기술이다. 비전은 카메라를 통해 얻은 이미지를 영상처리하여 사용하는 것이다. 이미지를 통해 색에 대한 정보를 얻을 수 있고, 카메라 두 대가 있으면 3차원 정보, 즉 사물의 거리와 3차원의 모양도 알 수 있는 인간의 눈과 흡사한 형태의 시각적 기능을 가능하게 한다. 하지만 비전을 이용하여 정보를 얻기 위해서는, 이미지를 처리하고 분석하는 연산이 필요한데, 연산에 필요한 데이터 양이 많고 계산하는 알고리즘도 복잡하여, 프로세서와 메모리의 부담이 커지는 단점이 있다. 레이저 거리계는 정해진 각도 범위 내에서 일정 각도 간격으로 물체와의 거리를 측정하여 출력하는 장치로써 2차원 평면상의 거리정보를 얻을 수 있다. 레이저 거리계는 물체의 위치 및 형태인식에 필요한 부담이 비전에 비해 훨씬 적고 알고리즘도 간단하며 비전센서가 정상 동작할 수 없는 어두운 곳에서도 레이저 거리계는 문제 없이 동작하는 등의 장점이 있다.

움직이는 물체를 인식하여 매니퓰레이터로 잡는 것에 관한 논문은 대부분 비전센서를 이용하였다. Peter 등은 원형 컨베이어 시스템에서 등속으로 움직이는 물체를 매니퓰레이터를 이용하여 파지 하는 시스템을 구축하였다[1]. 시스템에 포함된 컴퓨터는 영상처리를 위한 컴퓨터, 물체의 위치를 파악하는 등의 알고리즘 구현을 위한 호스트 컴퓨터, 매니퓰레이터 제어를 위한 컴퓨터 등 총 세 개로 구성되어 있다. 영상처리의 복잡한 알고리즘 구현을 위해 하드웨어 비용이 많이 들어가게 된 것이라 할 수 있다. 오승욱 등은 매니퓰레이터의 end-effector 부분에 스테레오 카메라를 장착하여 움직이는 물체를 잡는 방법을 소개하고 시뮬레이션을 하였다[2]. Houshangi 역시 움직이는 물체를 잡기 위해 스테레오 카메라를 장착하여, 물체의 움직임을 예측하여 물체를 잡는 방법에 대해 소개하고, 시뮬레이션과 원통형 물체를 잡는 실험을 수행하였다[3]. 이와 같이 비전 센서를 이용하면 물체의 거리를 알기 위해 복잡한 알고리즘으로 계산해야 된다. 로봇이 움직이는 물체를 감지하여 특정 시간에 따른 물체의 위치를 예측하여 파지하려 한다면, 물체의 움직임에 대한 정보를 빠르게 실시간으로 계산해야 한다. 이런 실시간성을 필요로 하는 로봇에 대하여, 3차원 물체의 형태를 고려할 필요가 없는 경우, 레이저 거리계가 비전에 비해 정확한 계산이 가능하고 자원도 효율적으로 활용할 수 있는 장점이 있다.

레이저 거리계를 이용하여 움직이는 물체를 잡는 논문은 아직 없다. 레이저 거리계는 모바일 로봇이 장애물을 인식하여 회피하기 위해 널리 사용되어 왔다. Mendes는 차량에서 외부의 여러 장애물(차량, 보행자)을 인식하기 위해 레이저 거리계를 사용하였다[4]. 움직이는 물체를 인식하고 추적하기 위해 DATMO(Detection And Tracking Moving Object) 시스템을 제안하였고 물체의 움직임에 대한 인식을 통해 충돌 시간도 예측하였다. 차량의 속도가 빠를수록 가까운 물체의 위치를 더욱 정확하고 빠르게 인식할 필요성이 있는데, 비전 센서만을 이용해서는 이런 점을 충족시킬 수 없다. Goncalo Monteiro는 레이저 거리계를 이용하여 위치를 인식하고, 비전 센서를 사용하여 인식된 장애물에 대한 자세한 정보를 얻어내는 방법을 연구했다[5]. 레이저 거리계를 이용하여 장애물의 위치와 크기를 예측하여, 카메라에서 얻은 이미지 정보를 이용해서 그 관심영역(region of interest)에 대해서만 정보를 분석하여 장애물에 대한 구체적인 모양에 대한 정보를 얻어서 어떤 장애물인지 구분하였다. 레이저 거리계와 비전 센서의 장점을 동시에 이용하여 센서 정보량은 많지만 연산 시간이 과대하다는 단점이 있다.

대부분의 움직이는 물체는 운동 방향과 속력에 대한 규칙성을 가진다. 즉, 등속운동, 등가속운동 등의 운동을 하는 물

* 책임저자(Corresponding Author)
논문접수 : 2007. 5. 18., 채택확정 : 2008. 1. 17.
천홍석, 김병국 : KAIST 전자전산학과 전기전자전공
(hscheon@rtcl.kaist.ac.kr/bkkim@ee.kaist.ac.kr)

체가 많다. 전체 운동시간 동안 특정한 규칙을 갖지 않고 무작위 운동을 하는 물체도 짧은 주기로 속도, 가속도 등을 측정하여 주시하면 물체의 움직임을 예측 가능하다. 특정한 규칙으로 운동하는 물체를 파지하는 것을 성공하면 비슷한 과정을 통해 다른 패턴으로 동작하는 물체를 인식하는 것으로 확장하기 어렵지 않을 것이고, 무작위 운동을 하는 물체로 확장하는 것 또한 가능할 것이다. 따라서 무작위 운동을 하는 물체를 매니퓰레이터로 파지하기 위해서는 특정한 패턴을 가지고 움직이는 물체를 대상으로 하는 것을 우선적으로 검증해야 된다. 본 논문에서는 대표적인 반복운동의 하나이고 실험실 환경에서 소형 로봇으로 시간에 따른 물체의 운동을 인식하고 파지하는 것을 실험하기 적합한, 원운동을 하는 물체를 대상으로 하였다. 레이저 거리계를 이용하여 물체를 판별하기 위해서는 물체의 2차원 형태를 이용하여 인식할 수 있기 때문에 높이에 따라 모양이 달라지지 않고 파지하기가 용이한 원통형 물체를 인식하는 것으로 제한하였다. 캔음료와 같은 원통형 물체를 대량생산하는 공장의 생산라인에서 컨베이어벨트를 통해 운반되는 음료수를 포장과 같은 다른 작업을 위해, 잡아서 이동시키는 작업에 적용이 가능할 것이다.

본 논문에서는 레이저 거리계를 이용하여 움직이는 원통형 물체의 위치를 인식하는 방법과, 이를 이용하여 물체를 파지하는 로봇 매니퓰레이터 개발에 대해 다룬다. II장에서 로봇이 어떻게 설계되었는지에 대해 다루고, III장에서는 정지된 원통형 물체에 대한 인식과 원운동 하는 물체 추적에 대해 다룬다. 그리고 IV장은 매니퓰레이터를 이용하여 원운동하는 물체를 파지하는 방법을 다룬다. V장은 실험 결과를 보이고, VI장에서는 결론을 맺는다.

II. 시스템 구조

1. 하드웨어 구조

하드웨어는 그림 1과 같은 구조로 되어 있다. 시스템 보드를 기준으로 레이저 거리계와의 통신은 RS-232C 통신을 하고, 매니퓰레이터 각 관절 구동기인 CX-28과는 CAN 통신으로 연결하였다. 시스템 보드에는 CAN컨트롤러가 없기 때문에 CAN 컨트롤러인 SJA1000를 포함한 인터페이스 보드를 설계 구현 하였다.

시스템의 전자적인 제어를 위한 보드로써, PXA255 Xscale 프로세서를 기반으로 한 Falinux사의 EZ-X5보드(그림 2)를 사용하였다[6]. MCU 성능이 좋고 저전력의 특징을 가지며 RTAI 설치가 가능한 PXA255 프로세서를 내장하고 있는 작지만 강력한 성능을 가진 보드이다.

2. 소프트웨어 구조

움직이는 물체의 시간에 따른 위치를 예측하여 정해진 시점에 매니퓰레이터를 정해진 위치로 이동 시키기 위해서는 실시간 제어가 필요하다. 센서값을 읽고, 매니퓰레이터를 제어하는 등의 다양한 작업이 서로 다른 주기로 이루어져야 된다. 이를 위해 실시간 운영체제 기반으로 소프트웨어를 설계하였다. RTOS는 실시간 리눅스의 하나인 RTAI[8]를 사용하였다.

그림 3은 소프트웨어의 구조를 나타낸 것이다. 소프트웨어는 크게 두 부분으로 나누어져 있다. 매니퓰레이터와 레이저 거리계를 실시간으로 제어하기 위한 실시간 제어 프로그램

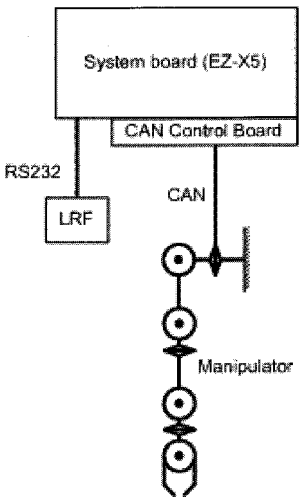
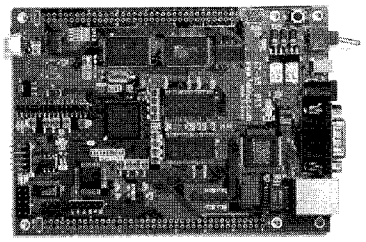


그림 1. 하드웨어 구조.
Fig. 1. Hardware structure.



Size	100mmX140mm
MCU	400MHz PXA255 ARM RISC chip
RAM	64Mbytes SDRAM
ROM1	512Kbytes Boot Flash
ROM2	64 Mbytes NAND Flash
Ethernet	CS8900 10Mbps
Serial	RS-232C 3 Port 지원
USB	USB Client

그림 2. EZ-X5.
Fig. 2. EZ-X5.

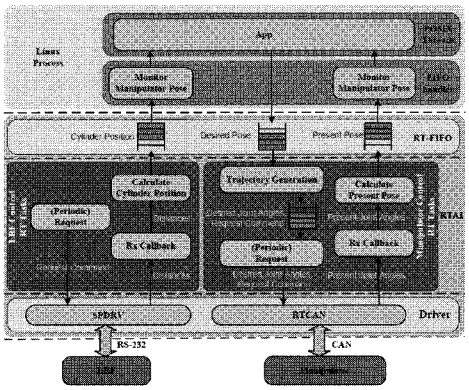


그림 3. 소프트웨어 구조.
Fig. 3. Software structure.

과 이를 이용하여 정지된 물체나 움직이는 물체를 잡는 응용프로그램이 있다. 실시간 제어 프로그램은 RTAI 상에서 모듈로 구현되었으며 응용프로그램은 리눅스의 사용자 영역에서 pthread 기반으로 구현되었다.

레이저 거리계 제어를 위한 소프트웨어 구조에 대해 먼저 보겠다. 본 논문에서 레이저 거리계를 제어하기 위해 RTAI단에서 할 일은 주기적으로 거리 정보를 레이저 거리계로부터 받아서, 그 정보를 이용하여 원통형 물체의 위치를 알아내고 RT-FIFO를 통하여 리눅스 사용자 프로그램으로 보내는 것이다. 이를 위해 실행되는 세 개의 실시간 태스크는 다음과 같은 일을 수행한다.

(1) Request task (periodic task): 주기적으로 200ms 마다 레이저 거리계에 데이터를 요청한다.

(2) Rx callback task: 레이저거리계 데이터 패킷이 도착하면 실시간 시리얼 드라이버인 SPDRV가 깨우는 태스크로써, 도착한 데이터를 'calculate cylinder position task'로 전달하는 일을 한다. 데이터가 도착하는 것은 비동기적으로 일어나기 때문에 이런 callback task에서는 데이터를 전달하는 일만 하는 것이 바람직하다.

(3) Calculate cylinder position task: 'rx callback task'로부터 받은 거리 정보들을 이용하여 원통형 물체의 현재 위치를 계산하고, 이를 RT-FIFO를 통하여 리눅스 프로세스로 보낸다.

리눅스 응용 프로그램에서는 RT-FIFO에 레이저 거리계 스캔 데이터가 도착하면 데이터를 받는 일을 하는 FIFO 핸들러가 있다. 받은 정보를 pthread로 제작된 쓰레드에서 사용하게 된다. 그 쓰레드는 물체가 원운동을 하는 경우에는 물체의 위치를 모니터링 하여 물체의 운동에 대한 데이터를 계산하는 일을 한다.

다음으로 매니퓰레이터 제어를 위한 소프트웨어 구조를 보겠다. 리눅스 응용프로그램의 쓰레드에서 RT-FIFO를 통해 매니퓰레이터 궤적 생성에 관한 요청을 보내면 RTAI의 실시간 태스크가 궤적 생성을 하여 매니퓰레이터를 동작하는 일을 하게 된다. 매니퓰레이터 제어를 위한 실시간 태스크는 다음과 같다.

(1) Trajectory generation task: 리눅스 프로세스로부터 RT-FIFO를 통하여 매니퓰레이터 자세 변화에 대한 정보를 받고, 거기에 맞게 궤적을 생성하고 그 궤적을 따라 자세 변화를 하기 위해, 시간에 따른 각 관절의 각도를 RT-FIFO를 통해 'request task'로 보내게 된다. 그리고 주기적으로 각 관절의 각도 정보를 요청하는 명령을 생성하여 RT-FIFO를 통해 request task로 보낸다.

(2) Request task (periodic task): 'trajectory generation task'로부터 받은 데이터를 주기적으로 CAN으로 보낸다.

(3) Rx callback task: CAN으로 외부에서 데이터가 도착하면 실시간 CAN 드라이버인 RTCAN이 깨우는 태스크로써, 도착한 데이터를 'calculate present pose task'로 전달한다.

(4) Calculate present pose rask: 각 관절의 각도를 이용하여 현재 매니퓰레이터의 자세를 계산하여 리눅스 프로세스로 보낸다.

CAN 통신 속도는 1 Mbps로 설정 하였고, 매니퓰레이터 제어주기는 10 ms로 설정하였다.

3. 응답 시간 분석

3.1 레이저 거리계 응답 시간 보정

그림 4는 레이저 거리계가 센싱한 시간과 시스템보드에서 데이터를 얻는 시간의 차이를 그린 것이다. 실험에서 사용한

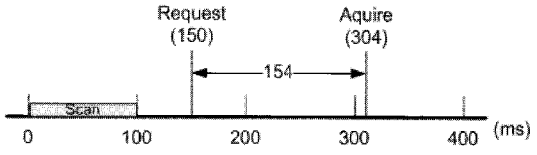


그림 4. 레이저 거리계 시간 보정.
Fig. 4. LRF timing correction.

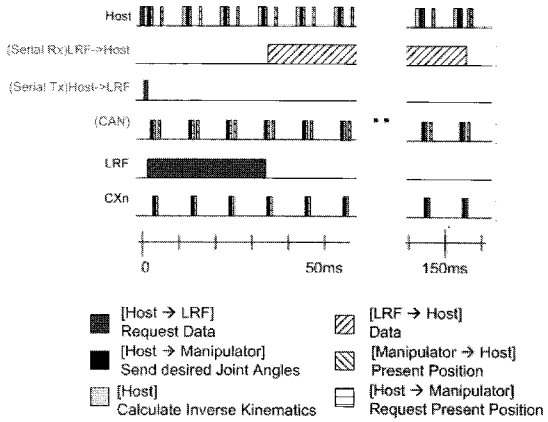


그림 5. 요소 별 시간 분석.
Fig. 5. System timing diagram.

프로세서인 PXA255의 UART와 레이저 거리계인 URG-04LX 간의 속도는 115200bps가 최대 이다. 레이저 거리계로 데이터를 요청하여 받는 시간을 실험적으로 측정한 결과 약 154ms이었다. 그리고 레이저 거리계의 동작 주기는 100ms 이기 때문에 데이터 요청 주기를 200ms으로 설정하였다. 그림 4와 같이 실제 스캔 하여 거리를 측정한 시각과 데이터를 얻는 시간에는 차이가 있기 때문에 이를 분석하여 보정해줄 필요가 있다. 레이저 거리계는 100ms를 주기로 스캔 한다. 호스트에서 200ms마다 데이터를 요청하는데, 기대치는 데이터가 스캔 완료 50ms이후라고 할 수 있다. 요청한 데이터가 호스트로 도착하는 데 걸리는 시간은 약 154ms이므로 스캔 시간과 데이터를 얻은 시각은 데이터에 따라서 204ms에서 304ms인 것을 알 수 있다. 따라서 전체 데이터를 평균적으로 254ms 이전의 데이터인 것으로 보정하였다.

3.2 시스템의 응답 시간 분석

전체 시스템의 응답시간의 분석을 수행하였다. 호스트에서 레이저 거리계로 데이터를 요청하여 요청이 레이저 거리계로 도착하는데 걸리는 시간은 약 868us이고, 호스트에서 매니퓰레이터의 모든 관절 구동기에 위치 명령 또는 데이터 요청 명령을 보내는 데 걸리는 시간은 약 864us이다. 이는 RS-232C와 CAN의 통신속도와 관절 구동기, 레이저 거리계의 데이터 패킷 사이즈에 의해 계산된 시간 값이다. 소프트웨어의 주기 결정에 따라 전체적인 시간은 그림 5와 같이 된다. 시스템의 구성요소들이 소프트웨어적으로 문제 없이 제어주기 이내에 동작하는 것을 확인할 수 있다.

III. 원통형 물체 인식

1. 정지된 원통형 물체 인식

움직이는 물체를 인식하기 위한 기반으로써 물체가 정지

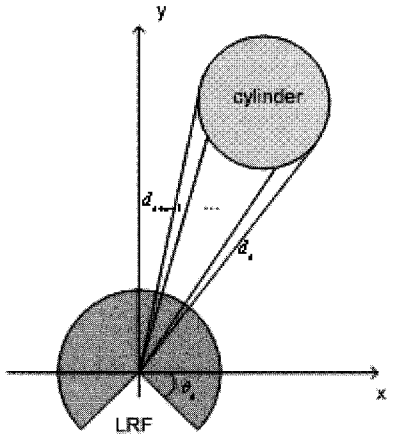


그림 6. 레이저 거리계를 이용한 원통형 물체 인식.
Fig. 6. Cylinder detection using LRF.

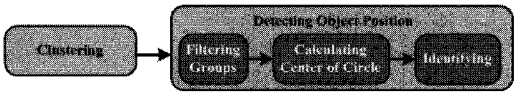


그림 7. 정지 물체 인식.
Fig. 7. Fixed object detection.

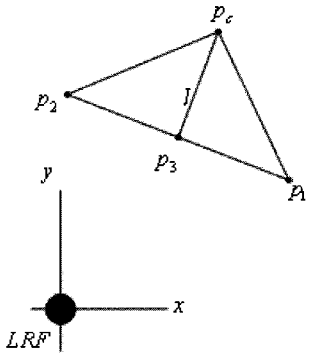


그림 8. 원통형 물체 중점 인식.
Fig. 8. Center of cylinder detection.

되어 있을 때의 인식이 필요하다.

레이저 거리계를 이용하면, 레이저 거리계의 각도 단위(angular resolution, α) 마다 레이저 거리계 중심으로부터의 거리를 알 수 있다. 레이저 거리계의 k 번째 데이터 거리 d_k 와 (1)과 같이 구할 수 있는 k 번째 데이터 각도를 이용하여 2차원 좌표로 변환할 수 있다(2). 실험에서 사용한 레이저 거리계는 그림 6과 같이 θ_s 는 -30°, 스캔 각도 증가 단위 α 는 약 0.35°, k 는 0이상 681이하의 정수이다.

$$\theta_k = k\alpha - \theta_s \tag{1}$$

$$x_k = d_k \cos\theta_k \qquad y_k = d_k \sin\theta_k \tag{2}$$

그림 7은 레이저 거리계에서 받은 거리 정보를 이용하여 정지되어 있는 반지름 r 이 주어진 원통형 물체의 위치를 인식하는 방법이다.

각 단계에서는 다음과 같은 일을 한다.

1.1 Clustering

레이저 거리계로 얻은 인접한 거리 정보의 차가 특정 거리(threshold) 이상으로 크면 물체 사이의 경계선으로 판단하여, 레이저 거리계로 얻은 거리 정보를 선분, 호 등을 포함하는 여러 그룹으로 나눈다.

2.2 Detecting object position

원통형 물체를 찾고 그 물체의 위치를 알아낸다.

2.2.1 Filtering groups

각 그룹의 가장 멀리 있는 점의 거리가 원 내부의 최장거리인 지름보다 작을 때만 반지름이 r 인 원통일 가능성이 있다고 판단한다. 즉, 각 그룹의 k 가 s 에서 $s+n-1$ 이고, LRF의 최대 오차가 r_e 라면, 다음 식이 성립되는 그룹만을 다음 단계인 'Calculating Center of Circle'로 넘어간다.

$$\sqrt{(x_{s+n-1}-x_s)^2+(y_{s+n-1}-y_s)^2} < 2(r+r_e) \tag{3}$$

2.2.2 Calculating center of circle

그림 6에서, 두 개의 거리 정보를 이용해서 그 원통형 물체의 중점을 구할 수 있다.

그림 8과 같이 인식된 두 점 p_1 과 p_2 의 중점을 p_3 이라 하고, p_3 과 p_c 를 지나는 직선 기울기는 s_1, 그리고 p_3 과 p_c 사이 거리를 l 이라고 하면, 원의 중심은 다음과 같다.

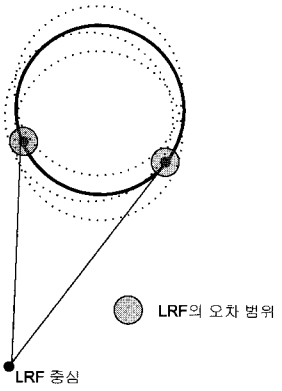
$$p_c = (x_3 + l\cos(\tan^{-1}s_1),\ y_3 + l\sin(\tan^{-1}s_1)) \tag{4}$$

그림 6과 같이 물체에 속해 있는 거리 정보가 n 개 일 때, $\frac{n(n-1)}{2}$ 개의 다른 두 데이터를 이용하여 원의 중점을 구할 수 있다. 하지만 시간이 그만큼 많이 소요된다. 그래서 그룹 내의 모든 데이터를 이용하고, 오차를 줄이기 위해 두 데이터 간의 최단 거리를 최소화할 수 있도록 $0 \le k \le \left\lfloor \frac{n-1}{2} \right\rfloor$ 일 때, 해당 그룹 내의 k 번째 데이터와 $\left\lceil \frac{s+n-1}{2} \right\rceil + k$ 번째 데이터를 이용하여 원의 중점 p_{c_k} 를 구하여 평균한다. 즉, 본 연구에서는 $\lceil n/2 \rceil$ 번의 물체의 중심을 구하여 다음 식과 같이 평균을 내도록 하였다.

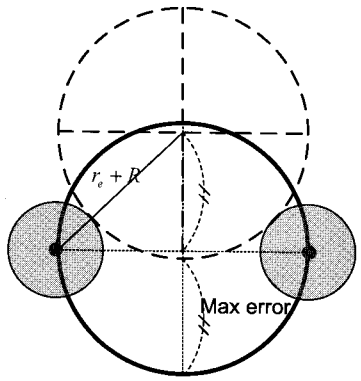
$$p_c = \sum_{k=0}^{\left\lfloor \frac{n-1}{2} \right\rfloor} p_{c\_k} \tag{5}$$

2.2.3 Identifying

다음으로, 원의 중심을 구한 그룹이 레이저 거리계로 인식하고자 하는 원통형 물체가 맞는지를 확인을 해야 된다. 확인 방법은 그림 6에서 얻은 물체까지의 거리 정보 d_s 에서 d_{s+n-1} 를 이용하여 얻은 좌표들과 전 단계에서 구한 원의 중심과의 거리가 원통형 물체의 반지름 r 과 같은 지를 확인해야 된다. 하지만 레이저 거리계로부터 얻은 거리 데이터는 오차가 있을 수 있는데, 그 오차로 인해 전 단계에서 구한 원의 중심 또한 그림 9(a)의 점선으로 된 원과 같이 오차가 있을 수 있다. 레이저 거리계의 최대 오차를 r_e 라고 하면, 원



(a) 오차



(b) 최대 오차

그림 9. 원통형 물체의 위치 오차.
Fig. 9. Position error of cylinder.

의 중심 오차의 최대값은 그림 9(b)를 통해 다음과 같음을 알 수 있다.

$$e_{\max} = \sqrt{(r_e + r)^2 - r^2} \tag{6}$$

따라서 d_s 에서 d_{s+n-1} 까지의 거리로 얻은 좌표들과 전 단계에서 구한 원의 중심과의 거리 l 이 다음 식을 만족하면 원하는 찾고자 하는 물체인 것으로 판단할 수 있다.

$$r - e_{\max} < l < r + e_{\max} \tag{7}$$

2. 원운동 하는 물체 인식

정지 물체 인식을 기반으로 하여, 원운동 하는 물체의 인식에 대해 기술한다. 물체를 잡기 위해 물체의 원운동을 인식한다는 것은 시간에 따른 물체의 위치를 예측할 수 있다는 것이다. 이를 위해 각속도 ω, 원의 중심 p_c, 원의 반지름 R 을 알아야 된다. 원운동에 대한 이런 정보를 구하기 위해서는 세 번 물체의 위치(p_1, p_2, p_3)를 인식하면 된다.

그림 11과 같이 세 점 사이의 거리가 멀수록 원의 중심 오차가 줄어드는 것을 알 수 있다. 따라서 현재 시점의 위치를 p_3 으로 하고 2/3 주기 이전의 위치를 p_1 으로 정하고, p_1 과 p_3 의 중간 시점의 점을 p_2 로 정한다. 각 점의 좌표는 $p_k = (x_k, y_k)$ 로 표현한다.

그림 10에서 l_1, l_2 의 교점이 원운동의 중심 p_C 이다. p_1

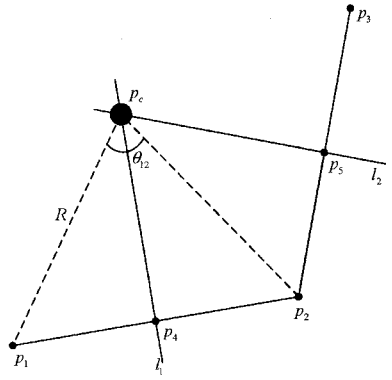
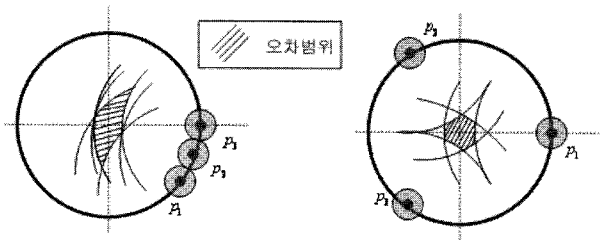


그림 10. 원운동 하는 물체의 위치.
Fig. 10. Position of an object moving a circle.



(a) 세 점이 가까울 때 (b) 세 점이 멀 때

그림 11. 원운동 하는 원의 중심 오차.
Fig. 11. Error of center of the moving circle.

과 p_2 의 중점을 p_4, p_2 와 p_3 의 중점을 p_5 라 하고, p_1 과 p_2 를 지나는 직선의 법선의 기울기를 s_1, p_2 와 p_3 를 지나는 직선의 법선의 기울기를 s_2 라 하면, 원의 중심은 다음과 같다.

$$p_C = (x_C, y_C) = \left(\frac{y_4 - y_5 + s_2 x_4}{s_2 - s_1},\ s_2(x_C - x_5) + y_5\right) \tag{8}$$

그리고 원의 반지름은 다음과 같다.

$$R = \overline{p_1 p_c} = \sqrt{(x_c - x_1)^2 + (y_c - y_1)^2} \tag{9}$$

그리고 레이저 거리계로부터 T 를 주기로 데이터를 얻고, p_1 과 p_2 의 거리가 d_{12} 일 때, 각속도는 (11)과 같다.

$$\theta_{12} = \cos^{-1}\frac{2R^2 - {d_{12}}^2}{2R^2},\ \theta_{12} \geq 0 \tag{10}$$

$$\omega = \frac{\theta_{12}}{T} \tag{11}$$

그런데 레이저 거리계의 오차에 의해 잘못 인식된 원통형 물체에 의해 원의 중심, 각속도, 반지름 등이 잘못 계산될 수 있다. 원운동에 대한 데이터의 오차를 줄이고 신뢰성을 높이기 위해서는 지속적으로 데이터를 구하고 평균을 내는 방법이 있다. 레이저 거리계 제어 주기마다 데이터를 구해서 기존의 데이터와 더해서 평균을 낸다. n 번째로 구한 데이터를 구한 후 더 신뢰성 있는 데이터를 구하는 방법이다.

그림 12에서 피드백 표시는 과거의 데이터가 'object trac-

그림 12. 움직이는 물체 인식.
Fig. 12. Moving object detection.

king'에 사용 된다는 것을 의미한다.

n 번째 데이터로 구한 원의 중심이 $p_{C\_n}=(x_{C\_n}, y_{C\_n})$ 일 때, 원의 중심은 다음과 같다.

$$p_C=\left(\frac{\sum_{k=1}^{n-1}x_{C\_k}+x_{C\_n}}{n}, \frac{\sum_{k=1}^{n-1}y_{C\_k}+y_{C\_n}}{n}\right) \tag{12}$$

각속도를 n 번째 데이터로 구할 때, θ_{12} 는 수정되어야 된다. p_C 와 p_1 의 거리가 R_1, p_C 와 p_2 의 거리를 R_2 라고 하면, θ_{12} 는 다음과 같다.

$$\theta_{12}=\cos^{-1}\frac{R_1^2+R_2^2-d_{12}^2}{2R_1R_2},\ \theta_{12}\geq 0 \tag{13}$$

n 번째 시간에 구한 각속도가 ω_n 일 때, 각속도는 다음과 같다.

$$\omega=\frac{\sum_{k=1}^{n-1}\omega_k+\omega_n}{n} \tag{14}$$

반지름 R_n 또한 각속도와 같은 이유로 p_{C_n} 가 아닌 p_C 와 p_1 의 거리를 이용하여 계산한다는 것에 주의해야 된다. n 번째 시간의 평균 반지름은 다음과 같다.

$$R=\frac{\sum_{k=1}^{n-1}R_k+R_n}{n} \tag{15}$$

주기는 다음과 같다.

$$P=\frac{2\pi}{\omega} \tag{16}$$

IV. 매니퓰레이터 제어와 움직이는 물체 파지

매니퓰레이터는 Hollerbach가 제안한 방법을 참고하여 그림 13과 같은 구조로 설계하였다[7]. 움직이는 물체를 잡기 위해서는 정해진 위치로 이동하여 정해진 시간에 도착해야 된다. 관절 구동기 성능에 따라 매니퓰레이터가 동작할 수 있는 최대 속도, 최대 가속도가 있다. 최대한 이 범위를 벗어 나지 않기 위해, 직선 주행 시 그림 14와 같이 가속력을 최소화하도록 속력 프로파일을 생성하였다. 즉, 가능하다면 삼각형꼴 속력 프로파일(그림 14(a))을 만들고, 삼각형꼴이 불가능하다면 사다리꼴 속력 프로파일(그림 14(b))을 생성한다.

그림 15와 같이 경유점이 있을 때, 경유점에서는 곡선 경로로 이동한다. 그림 15는 P_{k-1} 을 출발하여 P_k 를 경유해서 P_{k+1} 에 도착하는 그림이다. 경유점으로부터 미리 정한 시간

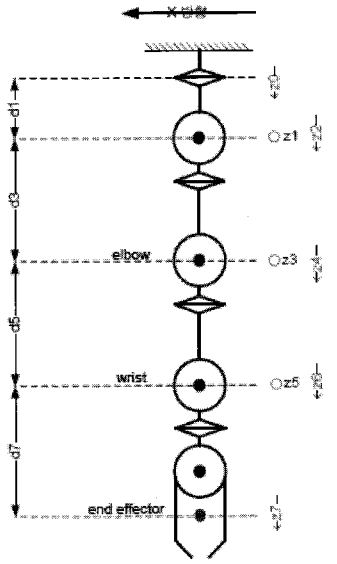
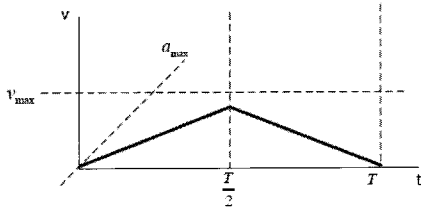
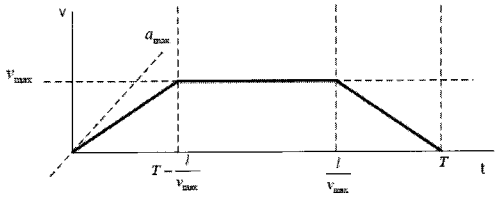


그림 13. 매니퓰레이터 구조.
Fig. 13. Manipulator structure.



(a) 삼각형꼴 속력 프로파일



(b) 사다리꼴 속력 프로파일

그림 14. 속력 프로파일.
Fig. 14. Velocity profile.

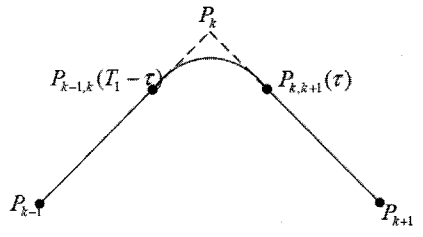


그림 15. 경유점에서의 궤적.
Fig. 15. Trajectory near a via point.

인 τ 이전의 지점에서, τ 이후의 지점까지 이동을 등가속도 운동으로 이동하도록 하였다[9].

움직이는 물체를 파지할 때는 잡는 시점과 접근 방법에 대해 결정해야 된다. 위치가 지정되면, 그 위치로 오는 시간을 예측하여 잡으면 된다. 잡는 위치는 그림 16과 같이 매니퓰레이터에서 가장 가까운 위치(B지점)에 물체가 왔을 때로 하

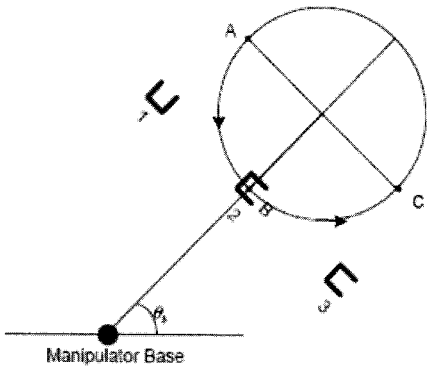


그림 16. 원운동 하는 물체 파지.
Fig. 16. Grasping a moving object.

였다. p_1 으로부터 시간 t 후 물체 위치 $p(t)$ 는 다음과 같다.

$$p(t) = \left(x_c + R\cos(\omega t + \theta_1),\ y_c + R\sin(\omega t + \theta_1)\right) \tag{17}$$

여기서

$$\theta_1 = \tan^{-1}\frac{y_1 - y_c}{x_1 - x_c} \tag{18}$$

따라서 물체를 파지하는 위치 B 는 다음과 같다.

$$B = \left(x_c - R\cos\theta_b,\ y_c - R\sin\theta_b\right) \tag{19}$$

이 때의 시간은 다음과 같다.

$$t_g = \frac{2n_c\pi + 1.5\pi - \theta_1 - (0.5\pi - \theta_b)}{\varpi} = \frac{2n_c\pi + \pi - \theta_1 + \theta_b}{\varpi} \tag{20}$$

(20)에서 n_c 은 지금부터 몇 바퀴 뒤에 잡을 지를 정하는 0 이상의 값 이므로, 최대한 빨리 잡기 위해서는 $n_c = 0$ 으로 한다.

잡는 위치와 시점이 정해지면, 잡기 위해 매니퓰레이터를 접근하는 방법을 정해야 된다. 잡는 순간 end-effector의 운동 방향은 잡는 순간 충격량(impulse)을 최소화하도록 하였다. 즉 그림 16과 같이 물체의 운동방향과 매니퓰레이터 운동 방향을 동일하게 하였다.

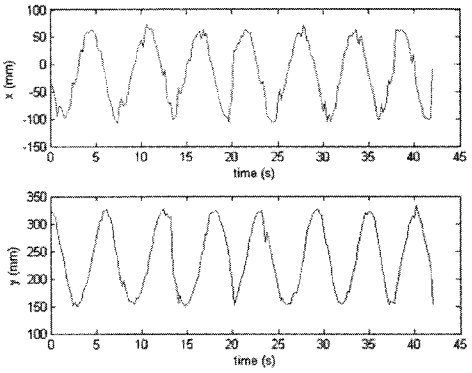
V. 실험 및 결과

1. 레이저 거리계를 이용한 원운동 하는 원통형 물체 인식

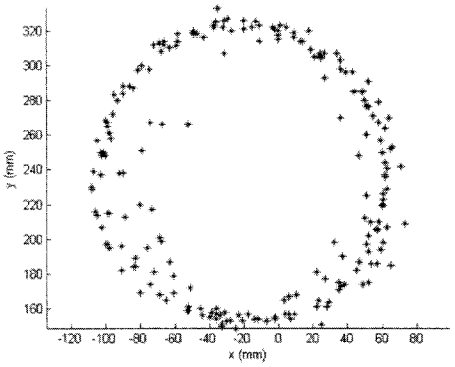
원운동 하는 물체를 레이저 거리계로 인식한 실험 결과를 그래프로 보인다. 200ms마다 레이저 거리계로부터 데이터를 받아서 42초 정도 측정한 데이터이다.

그림 17(a)는 시간에 따라 물체의 중심의 위치를 x, y 각 축에 대해 그래프를 그린 것이다. 원운동이므로 x, y의 그래프가 사인파 형태를 그리며 $\pi/2$ 만큼의 위상차가 나는 것을 확인할 수 있다. 그림 17(b)는 주기마다 인식된 물체 중심의 위치를 데이터를 x, y좌표에 그린 것이다. 원형을 그리는 것을 알 수 있다.

그림 18(a)는 원의 중심을 인식한 것이다. 그리고 그림 18(b)는 시간에 따른 원의 중심과 실제 원의 중심(-20, 240)과의 거리 차다. 9초 이후로는 10mm 이내로 유지된다.

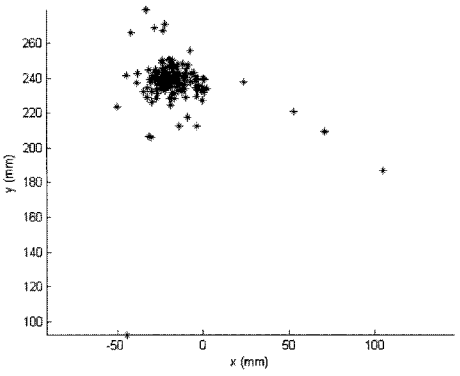


(a) 시간에 따른 물체의 위치 x, y값

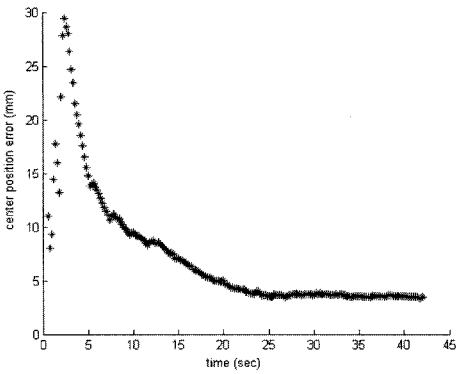


(b) 매 주기 인식된 물체의 위치

그림 17. 시간에 따른 물체의 위치.
Fig. 17. Object position vs. time.

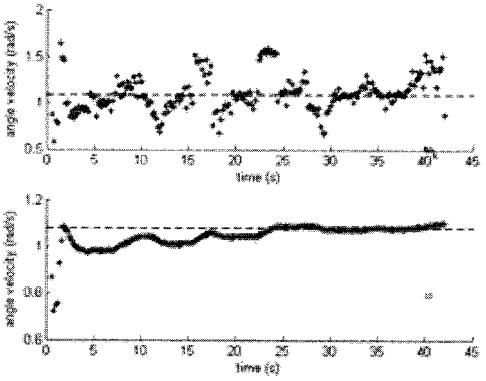


(a) 인식된 원의 중심

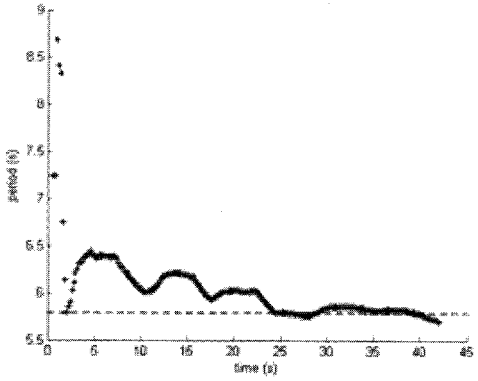


(b) 시간에 따른 오차

그림 18. 원운동의 중심.
Fig. 18. Center of an orbit.



(a) 각속도



(b) 주기

그림 19. 각속도와 주기.

Fig. 19. Angular velocity and period.

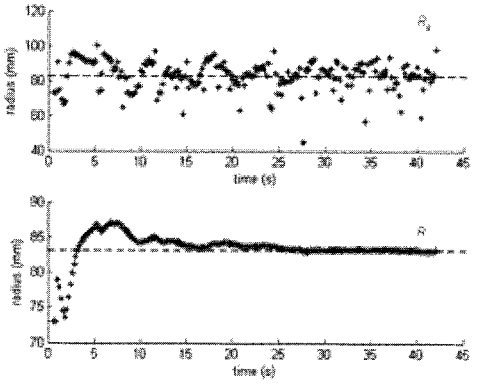


그림 20. 반지름.

Fig. 20. Radius.

그림 19(a)는 각속도를 그래프로 그린 것이다. 위쪽 그림은 주기마다 측정한 각속도 ω_k 를 나타낸 것이고 아래 그림은 누적 값의 평균을 낸 각속도 ω 이다. 그림에서 점선은 실제 각속도를 나타낸 것이다. 1.8초 이후에는 오차가 10% 이내로 줄어든다. 그림 19(b)는 5.8초 주기로 원운동 하는 물체를 인식한 시간에 따른 주기이다. 그림에서 점선은 실제 주기를 나타낸 것이다. 주기는 각속도와 반비례하기 때문에 주기 또한 1.8초 이후에는 오차가 10%이내로 줄어든다.

그림 20은 물체가 실제 반지름이 83mm인 원을 운동할 때 측정된 반지름이다. 점선은 실제반지름을 나타낸 것이다. 전체시간에서 10mm이내의 오차가 나며, 9초 이후에는 2mm이내의 오차로 안정화된다.

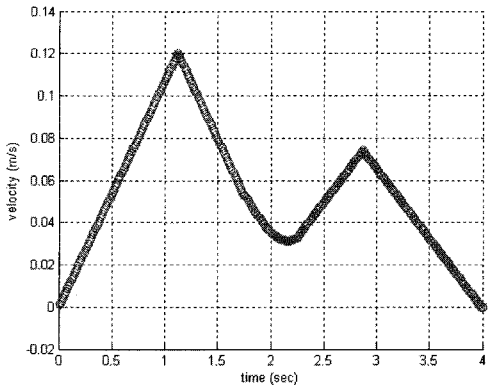
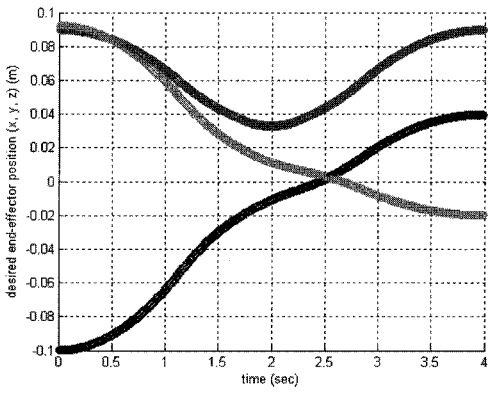
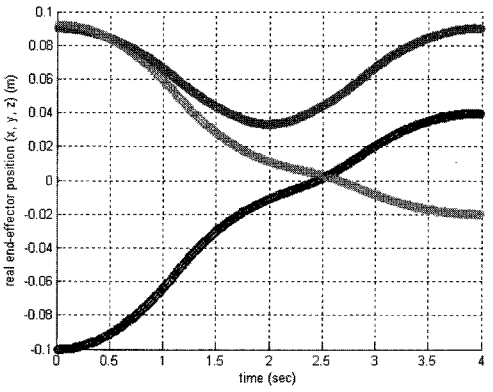


그림 21. End-effector의 시간에 따른 속력.

Fig. 21. End-effector velocity vs. time.



(a) 계획된 end-effector의 시간에 따른 위치(x, y, z)



(b) 실제 end-effector의 시간에 따른 위치(x, y, z)

그림 22. End-effector의 시간에 따른 위치.

Fig. 22. End-effector position vs. time.

반지름, 각속도, 주기 등은 시간이 갈수록 특정 값으로 수렴해 가는 것을 알 수 있다. 즉, 한번 측정하는 것보다 지속적으로 구한 데이터를 사용하여 평균을 내면 더 신뢰성 있는 데이터를 얻을 수 있는 것을 확인할 수 있다.

2. 매니퓰레이터 궤적

주어진 궤적을 따라 움직이는 매니퓰레이터의 위치를 제어 주기 마다 저장하여, 계획된 궤적과 비교하였다.

위치를 (x, y, z) (mm)로 표현할 때, (-100, 90, 92)지점에서 출발하여 2초가 지날 때, 경유점 (-10, 30, 12)를 지나서 출발 후

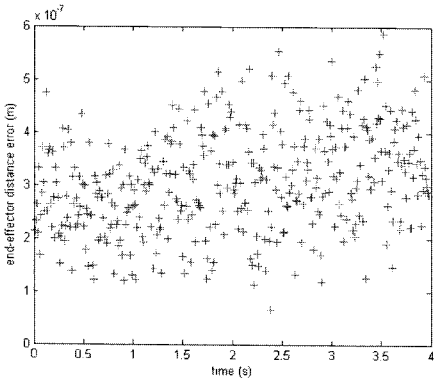


그림 23. 시간에 따른 end-Effector 위치 오차.
Fig. 23. End-effector position error vs. time.

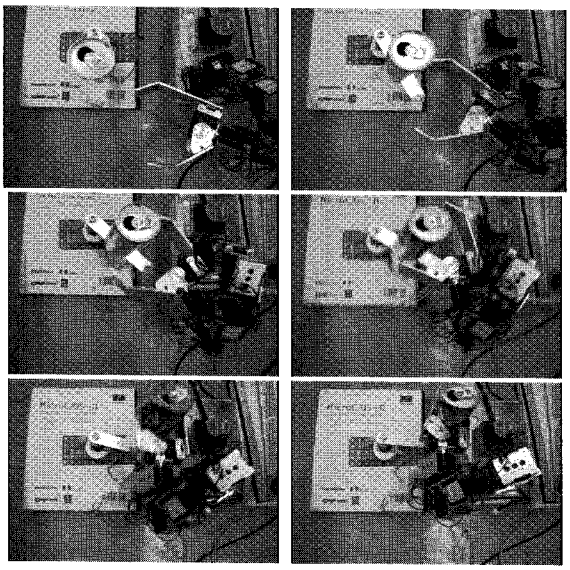


그림 24. 원운동 하는 원통형 물체 파지.
Fig. 24. Grasping a cylinder with circle motion.

4초에 목표점인 (40, 90, -20)에 도착하는 궤적에 대해 실험하였다.

그림 21은 생성된 end-effector의 속력 프로파일이다.

그림 22는 주어진 경로와 속력 프로파일에 따라 생성된 궤적과 실제로 매니퓰레이터가 움직인 궤적을 그린 것이다.

계획된 궤적과 실제 궤적의 시간에 따른 거리 오차는 그림 23과 같고, 오차는 1mm를 벗어나지 않는다. 약간의 오차가 생기는 것은 관절 구동기의 각도 최소 단위에 의한 것이다.

3. 원운동 하는 원통형 물체 파지

레이저 거리계를 이용하여 물체를 인식하고, 매니퓰레이터를 제어하여 그림 24와 같이 원운동 하는 원통형 물체 잡는 실험을 성공하였다.

VI. 결론

본 논문에서는 레이저 거리계를 기반으로 하여 원운동 하는 원통형 물체의 움직임을 감지하는 신뢰성 높은 알고리즘을 제시하였다. 기존의 비전 센서에 비해 거리에 대한 정확성이 뛰어나고 하드웨어와 소프트웨어적 자원효율이 뛰어난 레이저 거리계를 사용한 서보 시스템을 위한 알고리즘을 제시하였다. 임베디드 실시간 시스템 기반으로 제작된 매니퓰레이터 구동 시스템으로 실험을 하여 알고리즘의 타당성을 입증하였다.

물체의 반지름만 알면, 레이저 거리계를 이용하여 원운동의 반지름, 각속도, 원의 중점을 알아낼 수 있다. 레이저 거리계에서 지속적으로 들어오는 정보를 이용하여 시간이 지날수록 더욱 신뢰성 높은 데이터가 계산된다. 레이저 거리계의 실제 측정 시간과 거리 데이터를 습득한 시간의 차이를 보상하여 움직이는 물체의 더욱 정밀한 위치를 예측할 수 있었다. 물체를 잡기 위한 매니퓰레이터를 제작하여, 정해진 위치에 정해진 시간에 도착하는 궤적 생성 알고리즘을 적용하였고 실험결과 계획된 경로를 잘 따라가는 것을 확인할 수 있었다. 시스템의 실시간 제어를 위해 실시간 리눅스의 하나인 RTAI[8]를 기반으로 소프트웨어를 개발하였다.

본 논문에서 개발한 로봇은 레이저 거리계의 빠르고 정확한 거리 측정을 이용하였지만, 2차원의 물체 형태만을 알 수 있는 로봇이다. 이를 보완하기 위해 3차원 정보를 제공하는 비전센서를 추가적으로 사용하여 더욱 지능적인 로봇 개발이 가능할 것이다.

그리고 현재는 원운동만을 감지할 수 있도록 제작되었지만, 랜덤하게 움직이는 물체를 감지하고 잡는 것에 대한 연구가 필요하다. 그리고 본 논문에서는 매니퓰레이터 제어에 있어서 엔드이펙터의 최대 속도와 가속도는 고려하였지만, 각 관절의 최대 속도와 가속도를 고려하지 못하였다. 추후에는 이 부분까지 고려하여 매니퓰레이터를 제어할 필요가 있다.

참고문헌

[1] P. K. Allen, A. Timcenko, B. Yoshimi, and P. Michelman, "Automated tracking and grasping of a moving object with a robotic hand-eye system," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 2, pp. 152-165, 1993.

[2] 오승욱, 심귀보, "움직이는 물체의 파지를 위하여 스테레오 카메라를 사용한 로봇 매니퓰레이터의 위치 및 자세 제어," 제어계측연구회 합동학술연구발표회, 1994.

[3] N. Houshangi, "Control of robotic manipulator to grasp a moving target using vision," Robotics and Automation, 1990, Proceedings., *IEEE International Conference on,* pp. 604-609, 1990.

[4] A. Mendes, L. C. Bento and U. Nunes, "Multi-target detection and tracking with a laser scanner," *IEEE Intelligent Vehicles Symposium*, pp. 796-801, 2004.

[5] G. Monteiro, C. Premebida, P. Peixoto and U. Nunes, "Tracking and classification of dynamic obstacles using laser range finder and vision," in *FACT workshop at IROS 2006 on Intelligent Vehicles*, 2006.

[6] *Intel PXA255 Processor Developer Manual*, Intel(http://www.intel.com).

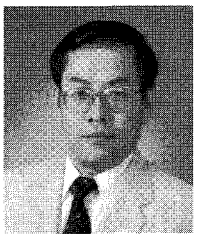
[7] J. M. Hollerbach, "Optimum kinematic design for a seven degree of freedom manipulator," in *Proc. 2nd Intern. Symp. on Robotics Research, Kyoto, Japan*, pp. 215-222, 1984.

[8] *DIAPM RTAI Programming Guide 1.0*, http://www. rtai.org, 2000.

[9] J. J. Craig, *Introduction to Robotics Mechanics and Control*, 3rd edition, Pearson Prentice Hall, 2005.

천 홍 석

1981년 4월 21일생. 2005년 고려대학교 전기전자전파공학부 졸업. 2007년 KAIST 대학원 전기전자전공 석사. 2007년~현재 동 대학원 전기전자전공 박사과정 재학 중. 관심분야는 장애물 회피 로봇.



김 병 국

1952년 10월 5일생. 1975년 서울대학교 전자공학과 졸업. 1975년 KAIST 전기 및 전자 공학과 석사. 1981년 동 대학원 박사. 1981년~1986년 우진계기(주) 연구 실장. 1982년~1984년 University of Michigan 방문연구. 1986년~현재 KAIST 전기 및 전자공학과 교수. 관심분야는 실시간 시스템, 로보틱스, 임베디드 제어.